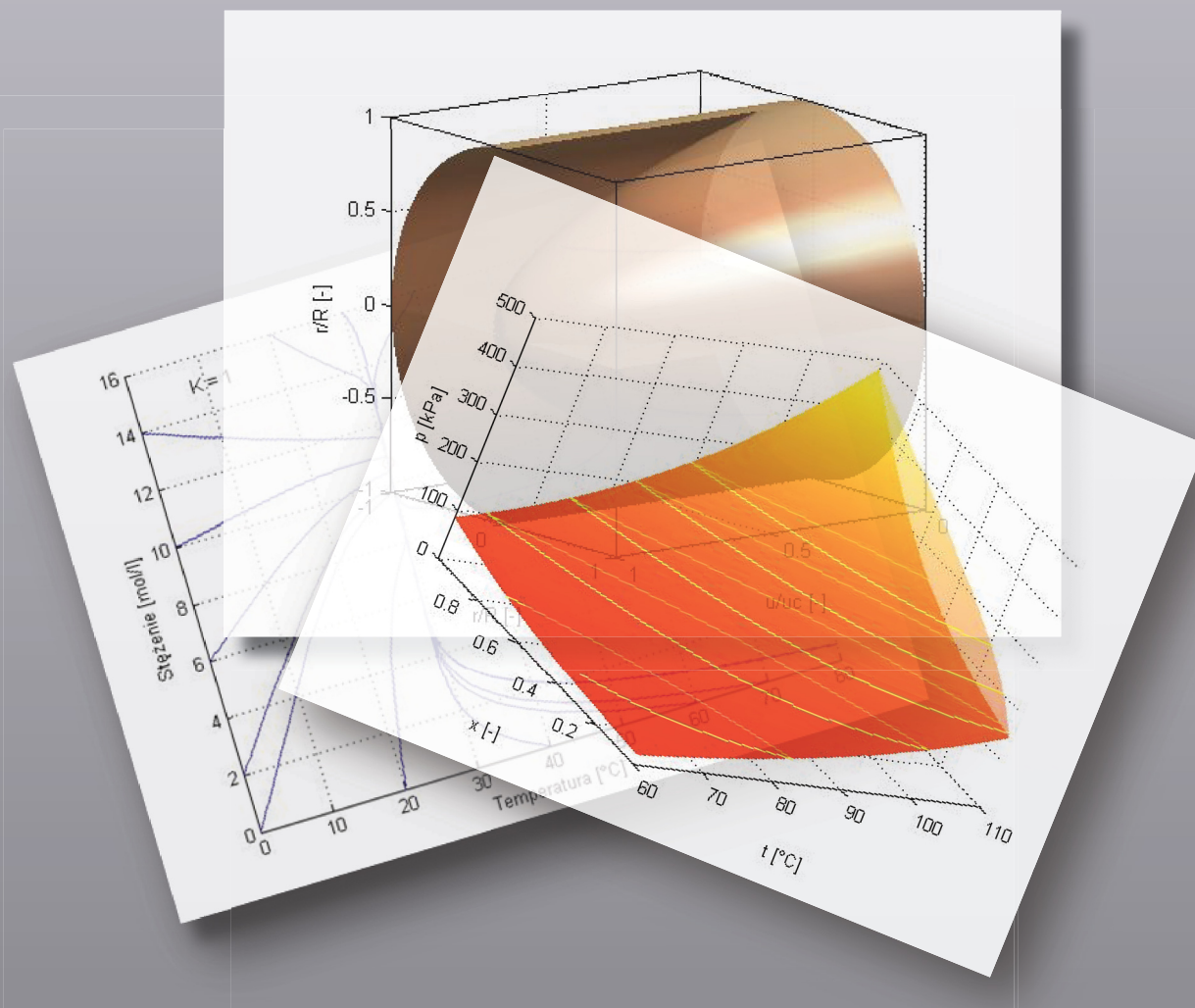


Zdzisław Pakowski
Robert Adamski

Podstawy MATLABa w inżynierii procesowej



„... książka jest pozycją, której do tej pory brakowało na rynku wydawniczym. Należy podkreślić, że jest to pierwsza [w Polsce] książka wiążąca zastosowanie MATLABa z inżynierią chemiczną i procesową. Wszechstronne zastosowanie tego programu w szeroko pojętych badaniach naukowych, w tworzeniu aplikacji badawczych bądź komercyjnych oraz w systemie nauczania studentów powoduje, iż taka pozycja jest bardzo potrzebna. Autorzy w swojej książce nie skupili się jedynie, jak to często się zdarza, na opisie programu, ale zamieścili cały zbiór napisanych przez siebie skryptów rozwiązujących praktyczne problemy z inżynierii chemicznej. Umożliwia to zarówno naukę MATLABa na praktycznych problemach, jak i rozwiązywanie innych problemów na bazie zamieszczonych projektów.”

*Dr hab. inż. Robert Grzywacz
Politechnika Krakowska*

Zdzisław Pakowski
Robert Adamski

Podstawy MATLABa w inżynierii procesowej

Recenzenci:
prof.dr hab. inż. Ireneusz Zbiciński
dr hab. inż. Robert Grzywacz

Redaktor Naukowy Wydziału
Inżynierii Procesowej i Ochrony Środowiska
dr hab. inż. Tadeusz Gluba

© Copyright by Politechnika Łódzka 2014

WYDAWNICTWO POLITECHNIKI ŁÓDZKIEJ

90-924 Łódź, ul. Wólczańska 223
tel. 42 631 20 87, fax 42 631 25 38
e-mail: zmowienia@info.p.lodz.pl
www.wydawnictwa.p.lodz.pl

ISBN 978-83-7283-630-4

Nakład 250 egz. Ark druk 30,0. Papier offset. 80 g 70 x 100
Druk ukończono w listopadzie 2014 r.
Wykonano w Drukarni Quick-Druk, 90-562 Łódź, ul. Łąkowa 11
Nr 2115

SPIS TREŚCI

1. WSTĘP	7
2. ŚRODOWISKO MATLABA	10
2.1. Charakterystyka	10
2.2. Budowa pakietu	11
2.2.1. Okno poleceń, katalogów i pamięci roboczej	12
2.2.2. Podstawowe zasady poruszania się w obrębie środowiska MATLAB	14
2.3. Liczby	15
2.4. Zmienne	15
2.4.1. Wprowadzanie zmiennych	16
2.4.2. Formaty liczbowe	18
2.4.3. Operatory arytmetyczne i boolowskie	18
2.5. Specjalne znaki, nazwy i funkcje	19
2.6. Pliki w MATLABie	21
2.7. Zarządzanie pamięcią	22
2.8. Tworzenie skryptów i funkcji	23
2.8.1. Skrypt a funkcja	23
2.8.2. Budowa funkcji	24
2.8.3. Wywołanie funkcji	25
2.8.4. Umieszczanie wielu funkcji w jednym pliku	27
2.8.5. Przekazywanie nazwy funkcji jako parametru	29
2.8.6. Funkcje anonimowe	31
2.8.7. Przekazywanie dodatkowych parametrów do funkcji wbudowanych	33
2.8.8. Funkcje matematyczne MATLABa przydatne w obliczeniach inżynierskich	36
2.9. Instrukcje	37
2.9.1. Instrukcje warunkowe	37
2.9.2. Instrukcje iteracyjne	38
2.9.3. Instrukcje wyboru	39
2.9.4. Instrukcje wejścia/wyjścia w oknie Command Window	39
2.9.5. Uruchamianie programów	40
2.9.6. Publikowanie raportów	41
3. OPERACJE NA MACIERZACH	49
3.1. Macierze pełne	49

3.1.1.	Tworzenie macierzy pełnych	49
3.1.2.	Generowanie macierzy pełnych	51
3.1.3.	Funkcje przekształcające macierze pełne	52
3.1.4.	Macierze wielowymiarowe	53
3.2.	Macierze rzadkie	55
3.2.1.	Charakterystyka macierzy rzadkich	55
3.2.2.	Generowanie macierzy rzadkich	57
3.2.3.	Operacje na macierzach rzadkich	57
3.3.	Łańcuchy i tablice znakowe	59
3.3.1.	Funkcje wykonujące operacje na łańcuchach	61
3.4.	Tablice strukturalne	61
3.5.	Macierze komórkowe	64
3.5.1.	Funkcje działające na macierzach komórkowych	64
3.6.	Odwołanie do elementów macierzy	67
3.7.	Operacje na macierzach i tablicach	68
3.8.	Funkcje macierzowe	73
3.9.	Właściwości i normy macierzy	77
4.	WYKRESY I GRAFIKA	97
4.1.	Wykresy 2D	97
4.1.1.	Tworzenie wykresu 2D	97
4.1.2.	Typy wykresów 2D	101
4.1.3.	Inne funkcje wykorzystywane do pracy z wykresami	104
4.1.4.	Style linii i punktów	106
4.2.	Wykresy 3D	107
4.2.1.	Przygotowanie danych	107
4.2.2.	Wykres funkcji dwóch zmiennych	107
4.2.3.	Typy wykresów 3D	108
4.2.3.1.	Powierzchnie w przestrzeni	108
4.2.3.2.	Krzywe przestrzenne	112
4.2.3.3.	Wizualizacja danych statystycznych	113
4.2.4.	Polecenia przedstawiające funkcje wielu zmiennych	115
4.2.4.1.	Pola skalowne	115
4.2.4.2.	Pola wektorowe	118
4.3.	Grafika za pomocą uchwytów obiektów graficznych (handle graphics)	122
4.4.	Wykorzystanie polecenia patch	137
4.5.	Animacja	143

5. ANALIZA DANYCH DOŚWIADCZALNYCH	149
5.1. Zapisywanie i wczytywanie danych	149
5.1.1. Eksport danych	149
5.1.2. Import danych	156
5.2. Obróbka wstępna danych	160
5.3. Aproksymacja	162
5.3.1. Metoda najmniejszych kwadratów	163
5.3.2. Ocena statystyczna jakości parametrów funkcji aproksymującej	164
5.3.3. Aproksymacja liniowa	165
5.3.4. Aproksymacja wielomianem	166
5.3.5. Aproksymacja dowolną funkcją	171
5.3.6. Aproksymacja funkcji wielu zmiennych	184
5.3.7. Wykorzystanie narzędzia Basic Fitting	200
5.3.8. Korzystanie z narzędzia Curve Fitting Tool	203
5.4. Interpolacja danych	208
5.5. Wygładzanie danych	213
6. ROZWIĄZYWANIE RÓWNAŃ	219
6.1. Równania liniowe	219
6.1.1. Układ równań dobrze i źle uwarunkowany	220
6.2. Równania i układy równań nieliniowych	228
6.2.1. Rozwiązywanie równań i układów równań algebraicznych z pomocą solverów MATLABa	228
6.2.1.1. Rozwiązanie pojedynczego równania algebraicznego	229
6.2.1.2. Rozwiązywanie układów równań	233
6.2.2. Rozwiązywanie nadokreślonych układów równań	242
6.2.3. Rozwiązywanie nieoznaczonych układów równań	244
7. ROZWIĄZYWANIE RÓWNAŃ RÓŻNICZKOWYCH ZWYCZAJNYCH	247
7.1. Typy równań i warunków brzegowych	247
7.1.1. Sztywność równań różniczkowych	248
7.1.1.1. Pojedyncze sztywne ODE	248
7.1.1.2. Sztywność układów równań	248
7.1.2. Numeryczne rozwiązywanie ODE	249
7.2. Integratory numeryczne równań różniczkowych MATLABa	249
7.3. Rozwiązywanie zagadnień początkowych	255
7.3.1. Procesy nieustalone	255
7.3.1.1. Teoretyczne podstawy dynamiki liniowej	257

7.3.1.2. Dynamika nieliniowa	263
7.3.2. Modele procesów o postaci ODE w inżynierii procesowej	271
7.3.2.1. Procesy nieustalone	272
7.3.2.2. Procesy ustalone	296
7.3.3. Wywoływanie integratorów z opcjami	307
7.3.3.1. Rozwiązanie zagadnienia początkowego z obsługą wydarzeń	310
7.3.3.2. Rozwiązanie zagadnienia początkowego z przekazywaniem dodatkowych parametrów	322
7.4. Rozwiązanie dwupunktowych zagadnień brzegowych	324
7.5. Rozwiązanie uwikłanych równań różniczkowych	328
8. ROZWIĄZYWANIE RÓWNAŃ RÓŻNICZKOWYCH CZĄSTKOWYCH	331
8.1. Klasyfikacja równań różniczkowych cząstkowych	331
8.1.1. Równania rzędu pierwszego	331
8.1.2. Równania rzędu drugiego	332
8.2. Zagadnienia początkowe i brzegowe	333
8.3. Rozwiązanie równań cząstkowych pierwszego rzędu	334
8.3.1. Stan nieustalony	334
8.3.2. Stan ustalony	340
8.4. Rozwiązanie równań typu parabolicznego	345
8.4.1. Metoda linii	345
8.4.2. Wykorzystanie integratora pdepe	357
8.5. Rozwiązanie równań typu eliptycznego	367
8.6. Problemy obliczeń numerycznych w rozwiązywaniu PDE	378
8.7. Przykłady obliczeniowe	382
9. IDENTYFIKACJA PARAMETRÓW RÓWNAŃ RÓŻNICZKOWYCH – ROZWIĄZYWANIE ZAGADNIEŃ ODWROTNYCH	419
9.1. Sformułowanie problemu identyfikacji parametrów	419
9.2. Sformułowanie zagadnienia odwrotnego	419
9.3. Algorytmy optymalizacyjne	420
9.4. Identyfikacja współczynników modelu określonego równaniem różniczkowym zwyczajnym	422
9.5. Identyfikacja współczynników modelu określonego równaniem różniczkowym cząstkowym	442
10. SPIS LITERATURY	469

1. WSTĘP

Potrzeby obliczeniowe inżynierii chemicznej i procesowej są coraz większe. Jest to związane zarówno z rozwojem stopnia poznania podstaw teoretycznych procesów, a co za tym idzie coraz bardziej skomplikowanymi ich modelami, jak i z rozwojem metod obliczeniowych, techniki komputerowej oraz oprogramowania, w tym wyspecjalizowanych programów do obliczeń numerycznych. W książce „Symulacja procesów inżynierii chemicznej” (Pakowski i Głębowski, 2001) opartej w całości o program Mathcad® zostały przedstawione podstawy zastosowania tego narzędzia do rozwiązywania problemów z dziedziny symulacji procesów inżynierii chemicznej.

Jednakże wiele problemów symulacyjnych przekracza często możliwości metod numerycznych Mathcada (Pakowski i in., 2003). Jest to podstawowa przyczyna wymagająca zastosowania innego narzędzia obliczeniowego. Drugą jest potrzeba rozszerzenia spektrum specjalistycznych funkcji i metod poza uniwersalne metody dostępne w Mathcadzie. Narzędziem spełniającym oba te warunki jest program MATLAB® firmy MathWorks. MATLAB pojawił się jako narzędzie obliczeniowe już w r. 1984 i początkowo służył jedynie do celów rachunku macierzowego. Obecnie jest uniwersalnym programem do obliczeń numerycznych stosowanym powszechnie, zarówno w nauce jak i dyscyplinach technicznych m.in. w inżynierii chemicznej i procesowej (Ramirez, 1997). Ma on znaczną przewagę w szybkości pisania, uruchamiania i wykonywania programów do obliczeń numerycznych nad zwykłymi językami programowania takimi jak Fortran, C+ itp.

Jednak MATLAB, w odróżnieniu od Mathcada, jest językiem programowania i wymaga znajomości takich pojęć z zakresu informatyki, jak: typ zmiennej, struktury danych, funkcja i procedura, sposoby przekazywania parametru, operacje dyskowe itd., a ponadto wymaga również znajomości zasad algebry liniowej. Natomiast program w Mathcadzie ma strukturę dokumentu i zasadniczo nie wymaga od użytkownika umiejętności programowania. Wysiłek włożony w naukę MATLABa procentuje dopiero po pewnym czasie (wg nas przeciętnie po pół roku pracy) i opłaca się wtedy, gdy zamierza się wykorzystywać program do rozwiązywania poważnych problemów obliczeniowych. Mathcada zaś można nauczyć się w ciągu dwóch tygodni i prawie natychmiast używać go do rozwiązywania stosunkowo prostych zadań.

Czytelnicy, którzy znają wcześniejszą pracę „Symulacja procesów inżynierii chemicznej” spodziewają się tu, być może, podobnej struktury, a więc podziału modeli procesów na klasy i prezentacji problemów dla każdej z klas. Zastosowano tu jednak inną strukturę, odpowiada ona rodzajom problemów matematycznych występujących w praktyce obliczeniowej inżynierii procesowej. Ograniczono

omówienie teorii zagadnień matematycznych a skupiono się narzędziach programistycznych do ich praktycznej realizacji, gdyż książka jest, przede wszystkim, prezentacją MATLABa. Dla ilustracji w tekście zawarto 59 przykładów obliczeniowych o charakterze ogólnym. Problematykę inżynierii chemicznej obejmują natomiast 43 projekty – zadania o charakterze praktycznym. Rozwiązują one konkretne problemy inżynierskie. Można je modyfikować w celu dostosowania do konkretnego zadania.

Szczególnie dużo uwagi poświęcono rozwiązywaniu równań różniczkowych zwyczajnych i cząstkowych, te bowiem sprawiają początkującym najwięcej problemów.

Przygotowane przez nas problemy obliczeniowe posługują się stosunkowo prostymi rozwiązaniami programistycznymi i prowadzą prosto do celu. Korzystając z wielu dostępnych w MATLABie opcji dodatkowych, można znacznie wzbogacić i udoskonalić te programy. Zachęcamy do intensywnego korzystania z pomocy programu MATLAB, szczególnie na etapie nauki na poziomie zaawansowanym, gdyż pozwala to znacznie rozwinąć umiejętności posługiwania się tym narzędziem. Liczymy na to, że trud włożony w naukę MATLABa zapoczątkuje podczas przygotowywania zarówno prac naukowych jak i w praktyce zawodowej. Książka jest przeznaczona dla wszystkich, którzy interesują się praktyką obliczeń numerycznych w dziedzinie inżynierii chemicznej i procesowej, studentów, doktorantów, pracowników naukowych i inżynierów.

Większość przykładów i projektów powstała w czasie wielu lat zajęć prowadzonych przez autorów na studium doktoranckim. W tym miejscu dziękujemy doktorantom, którzy wnieśli swój wkład w ich powstanie.

Autorzy

LITERATURA

Pakowski Z., Głębowski M., 2001, Symulacja procesów inżynierii chemicznej, Wyd. PŁ.

Pakowski Z., Głębowski M., Adamski R., 2003, Mathcad vs. MATLAB in the curriculum of chemical engineering at the Technical University of Lodz, Process Integration and Modeling of Chromatography Processes Conf., Rzeszów.

Ramirez W.F., Computational Methods for Process Simulation, 2nd ed., Butterworth-Heinemann, 1997.

PRZYKŁADOWE PROGRAMY

Książka zawiera 102 przykłady i projekty napisane w celu ilustracji omawianych w niej zagadnień. Większość z nich zamieszczona jest w tekście. Niektóre spośród tych programów używają dodatkowych plików pomocniczych, których w tekście nie zamieszczono. Wszystkie te programy są dostępne w katalogach PRZYKŁADY i PROJEKTY na stronie <http://books.wipos.p.lodz.pl/matlab/>. Podkatalogi są oznaczone numerem rozdziału, a pliki główne numerem przykładu lub projektu. Wszystkie programy oznaczone znakiem  w tekście znajdują się w tych katalogach. Pozostałe programy zawarte w książce należy wpisywać z klawiatury. Programy zostały napisane w MATLABie wersja R2011b (7.10).

2. ŚRODOWISKO MATLABa

W tym rozdziale zostanie przedstawiona krótka charakterystyka pakietu obliczeniowego MATLAB oraz podstawowe zasady edycji programów i wykorzystania wbudowanych weń funkcji. Opis dotyczy wersji R2011b. Ponieważ MATLAB nie posiada polskiej wersji językowej wszystkie nagłówki, komunikaty i nazwy będą podawane w oryginale, a tylko niektóre tłumaczone na polski. Od Czytelnika wymagamy podstawowej znajomości pojęć z dziedziny informatyki i choćby jednego języka programowania. Zalecana jest znajomość podstaw algebry liniowej i analizy numerycznej. Literaturą wprowadzającą mogą być prace *Brzózka i Dobczyński, 1998, Pratap, 2007, Stachurski 2003, Zalewski i Cegiela, 2003*. W notacji liczbowej w całej książce przyjęto kropkę dziesiętną tak jak w MATLABie.

2.1. Charakterystyka

MATLAB jest uniwersalnym środowiskiem obliczeń numerycznych, z zastosowaniem zasad języka programowania wysokiego poziomu. MATLAB przeznaczony jest do wykonywania obliczeń naukowo-technicznych i inżynierskich. Pakiet ten pozwala na wykonanie zarówno obliczeń numerycznych jak i symbolicznych, wizualizowanie ich na wykresach dwuwymiarowych (2D) i trójwymiarowych (3D) oraz tworzenie uproszczonego interfejsu komunikacji z użytkownikiem. Pierwotnie MATLAB, stworzony w końcu lat 70. XX w. przez Cleve'a Molera z Uniwersytetu New Mexico, udostępniał napisane w Fortranie pakiety do obliczeń z dziedziny algebry liniowej EISPACK i LINPACK. W roku 1983 Jack Little, Cleve Moler i Steve Bangert utworzyli firmę The MathWorks i napisali MATLABa w języku C. Nazwa MATLAB jest skrótem od **Matrix Laboratory**, czyli laboratorium macierzowe. Pakiet MATLAB może być wykorzystany w wielu różnych dziedzinach, jak: matematyka, fizyka, chemia, elektronika, automatyka, telekomunikacja, inżynieria chemiczna, meteorologia, medycyna, biologia, biomedycyna, aeronautyka, budownictwo, mechanika oraz wiele innych. Na większości uniwersytetów jest uważany za standardowy program do obliczeń numerycznych.

Korzystając z MATLABa należy pamiętać, że podstawowym typem zmiennej liczbowej jest w nim macierz i wektor (macierz jednokolumnowa), a skalar jest macierzą o wymiarze $[1 \times 1]$. Dlatego obliczenia najlepiej przedstawiać w formie operacji macierzowych i wektorowych, gdyż pakiet MATLAB został zoptymalizowany pod ich kątem. Z reguły MATLAB działa szybciej, gdy dana operacja jest zapisana w postaci operacji macierzowej a nie w postaci pętli `for` lub `while`,

tak jak w innych językach programowania. W celu przyspieszenia działania programów w MATLABie należy stosować dwie podstawowe zasady:

- wektoryzować pętle i algorytmy.

Obliczenie 1001 wartości funkcji $\sin(x)$ dla x w przedziale od 0 do 10 można wykonać pętlą `for` (nie zalecane):

```
i=0
for t=0:.01:10
    i=i+1
    y(i)=sin(t)
end
```

lub poprzez wektoryzację (zalecane):

```
t=0:0.001:10
y=sin(t)
```

- rezerwować pamięć dla wyników obliczeń.

Jeśli podczas obliczeń wymiar macierzy musi być ciągle modyfikowany, to spowalnia obliczenia i przyczynia się do fragmentacji pamięci. Można tego uniknąć, rezerwując przestrzeń na macierz wynikową np. przez wypełnienie jej zerami np. `y=zeros(1,100)` przed obliczeniami.

Dobrego stylu programowania można się nauczyć, śledząc przykłady zawarte w podręcznikach i katalogu `matlab\toolbox\matlab\demos` na płycie MATLABa.

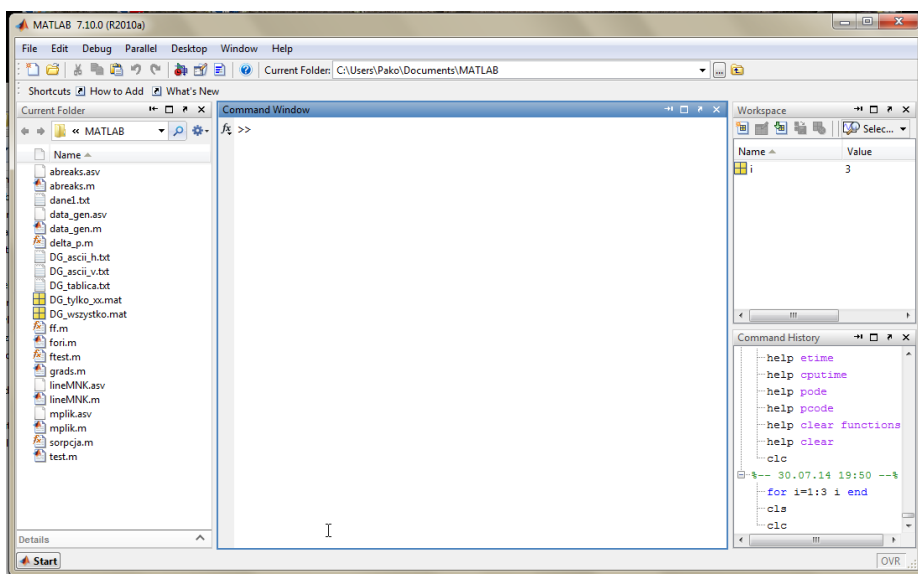
2.2. Budowa pakietu

Pakiet MATLAB jest produktem modułowym tzn., kupując MATLABa, otrzymujemy produkt podstawowy, w skład którego wchodzi podstawowe narzędzia matematyczne, algebra liniowa, analiza danych, integratory równań różniczkowych zwyczajnych, narzędzia graficznej prezentacji wyników i in. Jeśli jednak chcemy korzystać z bardziej zaawansowanych funkcji np. statystycznych, optymalizacyjnych, narzędzi do rozwiązywania równań różniczkowych cząstkowych jesteśmy zmuszeni do zakupu dodatkowych skrzynek narzędziowych tzw. Toolboxów. Dla omawianej tu wersji istnieje 27 takich skrzynek. Z istotnych dla inżyniera procesów można wymienić Optimization, Statistics, Data Acquisition, Curve Fitting czy Fuzzy Logic Toolbox.

Możliwe jest tworzenie przez użytkownika własnych skrzynek narzędziowych, dostępne są również skrzynki narzędziowe tworzone przez innych użytkowników. Zarówno sposobu dodawania własnych skrzynek jak i skrzynek innych niż fabryczne należy szukać w Internecie. Głównym adresem, pod którym można szukać pomocnych informacji jest oficjalna strona producenta MATLABa www.mathworks.com/matlabcentral/. Istnieją również strony w języku polskim.

2.2.1.Okno poleceń, katalogów i pamięci roboczej

Po uruchomieniu pakietu MATLAB widzimy pulpit Rys. 2.1, który domyślnie składa się z trzech górnych listewek – listewka Menu, listewka Toolbars (narzędziowa) listewka Shortcuts (skrótów), czterech okien i u dołu ekranu listewki statusu zawierającej przycisk **Start**.



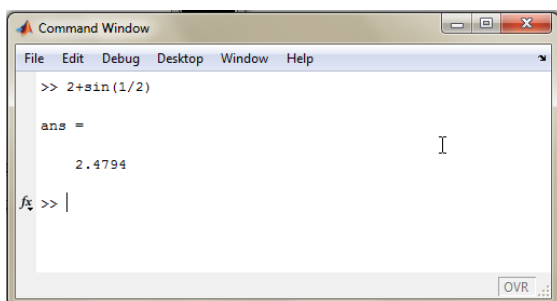
Rys. 2.1. Zrzut pulpitu programu MATLAB

Okna widoczne na pulpicie to:

- Okno **Current Folder** – bieżący katalog – jest panelem, w którym mamy podgląd wszystkich plików zawartych w bieżącym katalogu.
- Okno **Workspace** – przestrzeń robocza – jest panelem, w którym widać jakie zmienne znajdują się aktualnie w pamięci komputera. Wyświetlane są nazwy zmiennych i ich rozmiar. Klikając dwa razy lewym klawiszem myszy na nazwie zmiennej otwieramy okno z jej zawartością: **Array**

editor – podgląd macierzy. Jeśli jest to tablica lub wektor, to zobaczymy poszczególne jego elementy.

- Okno **Command Window** – okno poleceń jest panelem, z którego wywołujemy napisany program, zwany skrypcem lub funkcją (pliki te mają rozszerzenie m, więc nazywa się je również m-file – zob. punkt 2.6 i 2.8.1), poprzez wpisanie jego nazwy. Możemy tu również wprowadzać zmienne, wklejać programy z prostego edytora tekstu np. notatnika, itd. W oknie poleceń można bezpośrednio wykonywać polecenia, w tym obliczać wartości wyrażeń jak pokazano na Rys. 2.2. Uwaga okno to zostało odczepione od głównego okna programu Rys. 2.1 poleceniem wybranym z Menu Desktop/Undock Command Window. Do ponownego przyłączania służy przycisk ze strzałką w prawym rogu paska menu.



Rys. 2.2. Wykonanie obliczenia w oknie poleceń. Odpowiedź **ans** jest domyślną zmienną, pod którą jest podstawiana wartość obliczonego wyrażenia

Bardzo pomocne są komunikaty błędów, które są wyświetlane w tym oknie. Okno **Command Window** można w każdej chwili oczyścić poleceniem **clc**.

- Okno **Command History** zawiera kolejne, wydane w oknie **Command Window** polecenia, które są zapisywane w pamięci pod datą ich użycia. Poprzednie wydane polecenia można też przywołać klawiszem $\uparrow$ (kursor w górę) w oknie **Command Window**.

Przycisk **Start** znajdujący się z lewej strony u dołu ekranu umożliwia szybki dostęp do narzędzi, programów demo, skrótów i dokumentacji. Nie służy jednak do uruchamiania pisanych przez użytkownika programów. Pisanie programów i ich uruchamianie odbywa się w oknie **Editor**, otwierającym się po wybraniu z Menu opcji File/New lub ikony New Script z paska narzędzi.

Aktualny wygląd pulpitu, jak również wiele innych ustawień, użytkownik może dostosować do swoich potrzeb w opcjach Desktop i File/Preferences. W File/Preferences można np. ustawić wielkość i krój czcionki ekranu.

2.2.2. Podstawowe zasady poruszania się w obrębie środowiska MATLAB

Oto kilka wskazówek dla początkujących:

- MATLAB rozróżnia wielkie i małe litery.
- Nazwy zmiennych i funkcji muszą zaczynać się od litery, następne znaki mogą być cyframi.
- Nazwy zmiennych mogą mieć dowolną długość, ale tylko 19 znaków jest rozpoznawanych.
- Tekst (łańcuchy) zapisuje się w apostrofach.
- Postawienie znaku % (procent) powoduje, że wyrażenie za nim staje się komentarzem i nie jest wykonywane.
- Indeksy macierzy zaczynają się od 1.
- Zmienne są tworzone w pamięci po przypisaniu im wartości.
- Jeśli nazwa funkcji jest inna niż nazwa pliku, w którym funkcja ta jest zapisana, funkcja będzie znana pod tą drugą nazwą, lepiej więc używać tych samych nazw.
- Zmienna **ans** przyjmuje (domyślnie) wynik operacji, którego nie podstawiono pod żadną inną zmienną.
- W każdej chwili można podejrzeć aktualną listę zmiennych, albo w oknie **Workspace** lub za pomocą polecenia **who**.
- Postawienie znaku ; (średnik) na końcu polecenia powoduje pominięcie na ekranie w oknie **Command Window** wydruku wyniku jego działania.
- Pisząc nowy program, najlepiej umieścić go w nowym katalogu o tej samej nazwie, znajdą się tam wszystkie pliki wytworzone podczas budowy programu. Ponieważ funkcje pisane przez użytkownika umieszcza się w osobnych plikach stąd jeden program generuje z reguły wiele plików.
- W oknie **Command Window** możemy wykonywać operacje na plikach, tak jak w systemie operacyjnym DOS.
- Natychmiastową pomoc na temat polecenia, funkcji czy symbolu można wyświetlić wpisując polecenie **help**, po którym wpisuje się odpowiednią nazwę.
- Pracę z MATLABem kończy się przez wpisanie polecenia **exit** lub **quit**.

2.3. Liczby

Liczby rzeczywiste zapisuje się, stosując kropkę dziesiętną. Mogą one być poprzedzane znakiem + lub -. Tradycyjnie dozwolony jest zapis liczb z cechą i mantysą rozdzielonymi literą e. Do zapisu liczb zespolonych stosuje się jednostkę urojoną i lub j, zapisując przed nią część urojoną bez znaku mnożenia. Poniżej pokazano przykłady z komentarzem.

```
2           %liczba całkowita 2
+2          %liczba całkowita 2
-2          %liczba całkowita -2
2.2         %liczba zmiennoprzecinkowa (rzeczywista) 2,2
2e3         %liczba zmiennoprzecinkowa 2000
2.2e-02     %liczba zmiennoprzecinkowa 0,022
2+2i        %liczba zespolona 2+2√-1
```

Uwaga: lepiej nie pisać `2+2*i`, gdyż wtedy MATLAB rozumie `i` jako jednostkę urojoną, jedynie gdy zmienna ta nie została użyta wcześniej do innego celu.

2.4. Zmienne

Z punktu widzenia programisty zmienna jest elementem, którego wartość może ulegać zmianie podczas wykonywania programu. Z punktu widzenia komputera zmienna jest zdefiniowanym obszarem w pamięci, którego zawartość może zmieniać się podczas wykonywania programu. O rodzaju i sposobie prezentacji danych decyduje typ zmiennej. Zmienna może mieć nazwę o dowolnej długości, przy czym jedynie pierwsze 19 znaków jest rozróżniane. Nazwa może zawierać litery alfabetu łacińskiego (w tym podkreślenie) i cyfry, ale powinna rozpoczynać się od litery. Pamiętajmy też, że duże i małe litery są rozróżniane.

Jak w każdym języku programowania również w MATLABie występują zmienne globalne i zmienne lokalne. Zmienna lokalna to zmienna widoczna tylko w obrębie danej funkcji, inicjowana w momencie interpretacji skryptu i kasowana w momencie kończenia działania funkcji lub skryptu. Kolejne, ponowne wywołanie tej samej sekwencji kodu zapisanego w funkcji tworzy instancję (wystąpienie) zmiennej.

Zmienna globalna to zmienna widoczna w obrębie całego programu. W celu udostępnienia danej zmiennej lokalnej w funkcji dla innej funkcji lub skryptu trzeba ją zadeklarować jako globalną we wszystkich funkcjach lub skryptach, które jej używają. W tym celu stosuje się polecenie `global`, a następnie wymienienia się kolejne zmienne rozdzielone spacjami np. `global A B C D`.

Nadużywanie zmiennych globalnych może prowadzić do poważnych problemów, w przypadku skryptów wielowątkowych zmienne globalne mogą kolidować

ze zmiennymi lokalnymi, na działanie danej części skryptu może mieć wpływ funkcja zupełnie niezwiązana, o ile oba elementy używają tej samej zmiennej globalnej, i któryś z nich ją modyfikuje. Lepiej więc przekazywać zmienne przez nagłówki funkcji.

W typowych językach programowania zmienną trzeba zadeklarować na początku programu oraz podać jej typ. W MATLABie podawanie typu zmiennej jest niepotrzebne. Program sam rozpoznaje daną numeryczną, natomiast jeśli ma być to zmienna typu znakowego (łańcuch), umieszczamy wartość zmiennej w apostrofach. Udogodnieniem jest również to, że zmienną możemy deklarować w dowolnej części kodu, ale przed wyrażeniem z jej użyciem, pamiętajmy również o możliwości nadpisania innej wartości zmiennej po jej ponownym zadeklarowaniu. Program, interpretując kod, uzna, że właściwą wartością jest wartość ponownie podana i jej będzie używał do obliczeń. Trzeba zwrócić na to uwagę, ponieważ może to doprowadzić do błędnych wyników obliczeń.

Elementem wyróżniającym MATLAB jest przyjęty format zmiennych, a są nim wyłącznie macierze. W MATLABie wszystkie typy zmiennych są zapisywane w macierzach: zmienne numeryczne, logiczne czy tekstowe. W formie macierzy 1x1 jest zapisywany również skalar. W obliczeniach numerycznych bardzo często są wykorzystywane wektory. Wektory są zapisywane jak macierze jednowierszowe lub jednokolumnowe. Należy pamiętać, że w równaniach macierzowych wektor jest zawsze wektorem kolumnowym.

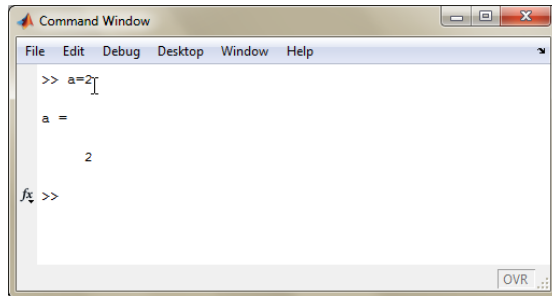
W łatwy sposób można deklarować macierze wielowymiarowe, przez użycie więcej niż dwu indeksów macierzowych lub macierze blokowe, albowiem MATLAB daje możliwość deklarowania macierzy lub wektora w danej komórce innej macierzy lub wektora.

MATLAB przydziela pamięć zmiennym w sposób dynamiczny w trakcie ich tworzenia. Zmienne dynamiczne – są to dynamiczne macierze, które bardzo ułatwiają pisanie skryptów, ponieważ niektóre procedury wbudowane w MATLAB dają wyniki, dla których użytkownik nie jest w stanie przewidzieć rozmiaru macierzy i zadeklarować odpowiedniej liczby komórek. Aby jednak przyspieszyć wykonywanie skryptu, lepiej jeśli znamy docelowy rozmiar macierzy, predefiniować macierz statycznie na początku skryptu, wypełniając ją np. zerami.

2.4.1. Wprowadzanie zmiennych

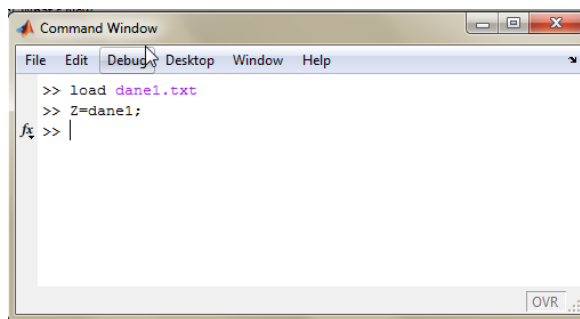
Zmienne możemy wprowadzać do pamięci roboczej w różny sposób np.:

- Bezpośrednio z linii komend oznaczonej znakiem >> w oknie poleceń. Wystarczy wpisać nazwę zmiennej, znak = oraz wyrażenie określające wartość zmiennej. Przykład pokazany jest na Rys. 2.3.



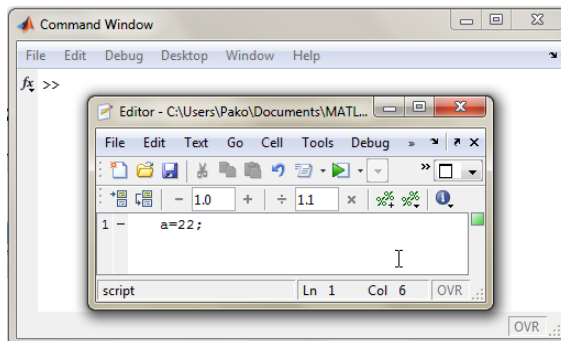
Rys. 2.3. Podstawienie wartości pod zmienną

- Wczytanie z pliku *.mat lub *.txt za pomocą polecenia load.



Rys. 2.4. Wczytanie danych z pliku i podstawienie ich pod zmienną Z

- Wprowadzenie z pliku *.m, który właśnie otwarto. Po uruchomieniu pliku zmienna a zostanie zapisana w pamięci roboczej.



Rys. 2.5. Wprowadzanie zmiennej z pliku *.m

2.4.2.Formaty liczbowe

Podstawowym typem elementów macierzy są liczby rzeczywiste. Format liczb rzeczywistych można zmienić poleceniem np. **format long**. Polecenie to dotyczy wszystkich zmiennych w pamięci. W MATLABie są dostępne następujące formaty liczbowe (na przykładzie liczby 3.45):

short	format krótki – 3.4500
short e	format krótki z wykładnikiem – 3.4500e+000
long	format długi – 3.450000000000000
long e	format długi z wykładnikiem – 3.450000000000000e+000
bank	format walutowy, do dwóch miejsc po przecinku - 3.45
hex	format szesnastkowy – 400b99999999999a
rat	format przybliżający liczby ułamkami małych liczb całkowitych – 69/20

2.4.3.Operatory arytmetyczne i boolowskie

- Operatory arytmetyczne

+	-	zmiana znaku
+	-	dodawanie i odejmowanie
*		mnożenie
/		dzielenie
\		odwrotne dzielenie macierzowe
^		potęgowanie

Użycie tych operatorów w działaniach na skalarach i macierzach działa inaczej i wymaga przestrzegania zasad. Szczegóły budowy wyrażeń arytmetycznych dla macierzy i skalarów przedstawione są w Rozdz. 3.

- Operatory boolowskie

Wyrażenia logiczne buduje się przy użyciu następujących operatorów relacji

<	mniejsze	A < B
>	większe	A > B
<=	mniejsze lub równe	A <= B
>=	większe lub równe	A >= B

<code>==</code>	równe	<code>A == B</code>
<code>~=</code>	nierówne	<code>A ~= B</code>

oraz następujących operatorów logicznych

<code>&</code>	AND	<code>A&B</code>
<code> </code>	OR	<code>A B</code>
<code>~</code>	NOT	<code>~A</code>
<code>xor</code>	XOR	<code>xor(x,y)</code>

Gdy wyrażenie boolowskie zbudowane z użyciem operatorów relacji i operatorów logicznych jest prawdziwe, przyjmuje ono wartość 1 (prawda), jeśli nie 0 (fałsz). W przypadku porównywania macierzy muszą one mieć te same rozmiary. Porównywane są odpowiadające elementy. Zwracana jest macierz o tych samych rozmiarach, zawierająca jedynki tam gdzie relacja jest spełniona i zera tam gdzie nie jest.

Dla przykładu jeśli mamy następujące macierze

```
A=[ 1 3 5
    2 2 4]
B=[ 2 3 8
    2 1 4]
```

to wyrażenie

```
w=A>=B
```

przyjmuje wartość

```
w=[ 0 1 0
    1 1 1]
```

2.5. Specjalne znaki, nazwy i funkcje

- Znaki specjalne występujące w MATLABie:

<code>=</code>	podstawienie wyrażenia pod zmienną
<code>[]</code>	używane przy definiowaniu macierzy i przy definiowaniu listy argumentów wyjściowych funkcji
<code>{ }</code>	używane przy definiowaniu i indeksowaniu macierzy komórkowych

- () używane przy zaznaczeniu kolejności wykonywanych działań, przy definiowaniu listy argumentów wejściowych funkcji i przy dostępie do elementów macierzy
- .
- ... przenoszenie polecenia do następnej linii
- ,
- ; koniec wiersza, powoduje wynik tej operacji, nie jest wyświetlany w panelu **Command Window**
- :
- % początek komentarza ignorowanego przez kompilator MATLABa
- ' ogranicznik łańcuchów

- Polecenia i stałe specjalne występujące w MATLABie.

ans	zmienna robocza, nazwa zmiennej przypisana automatycznie do wykonywanego wyrażenia, które nie zostało przypisane znakiem = żadnej innej zmiennej
computer	nazwa systemu operacyjnego zainstalowanego na komputerze w panelu Command Window
eps	określa dokładność obliczeń (dokładność zmiennoprzecinkowa) – wywołana bez argumentu wynosi 2.2204e-016
flops	licznik operacji zmiennoprzecinkowych
i, j	jednostka urojona
Inf	nieskończoność
NaN	wartość nieokreślona (nie liczba)
pi	3.14159265358979
realmax	1.797693134862316e+308 największa dostępna liczba rzeczywista
realmin	2.225073858507201e-308 najmniejsza dostępna dodatnia liczba rzeczywista

- Funkcje dotyczące czasu występujące w MATLABie

clock	aktualny czas i data wyświetlone w postaci wektora na panelu Command Window
cputime	wyświetla upływający czas od chwili uruchomienia MATLABa
date	wyświetla datę w postaci łańcucha
now	wyświetla czas i datę w postaci liczby

etime wyświetla wartość wybranego przedziału czasu między dwiema wartościami uzyskanymi za pomocą funkcji **clock**
tic, toc funkcje do odmierzenia czasu wykonywanych przez komputer operacji. Zmienna **toc** przechowuje wartość czasu jaki upłynął od wywołania **tic**.

2.6. Pliki w MATLABie

- Podstawowe typy plików w MATLABie:
 - *.**m** m-pliki są to pliki tekstowe, można rozróżnić dwa typy m-plików: skryptowe i funkcyjne. W skryptach są zapisane sekwencje poleceń MATLABa i wyrażeń arytmetycznych, natomiast funkcje są to pliki, w których sekwencja rozpoczyna się od słowa **function**
 - *.**mat** są to pliki, w których zapisane są dane w postaci binarnej
 - *.**mex** są to pliki skompilowane napisane w języku C lub Fortranie wywoływane w MATLABie
 - *.**fig** są to pliki, w których zapisywane są wykresy, które można ponownie edytować. Wykresy mogą być również eksportowane do standardowych i często używanych plików graficznych typu bmp, jpg, gif, tif, pdf.
 - *.**p** są to pliki pseudokodu tworzone za pomocą polecenia **pcode**. Jest to odpowiednik plików skompilowanych w innych językach programowania. Pliki te można uruchamiać w środowisku MATLABa, ale nie można ich edytować.
 - .**asv** pliki generowane przez autozapis co 5 minut, mogą służyć do odzyskania pliku.
- Operacje na plikach i katalogach

Polecenia wykorzystywane do operacji na plikach i katalogach: (elementy w nawiasach ostrych < > wpisuje użytkownik, ... oznacza możliwość kontynuacji listy)

save('<nazwa pliku>')

zapisuje zmienne z przestrzeni roboczej do pliku o rozszerzeniu *.mat

save('<nazwa pliku>', '<zmienna1>', '<zmienna2>', ...)

zapisuje tylko wymienione zmienne do pliku o rozszerzeniu *.mat

save('<nazwa pliku>', '- <opcja zapisu>', ...)

zapisuje zmienne z przestrzeni roboczej do pliku o rozszerzeniu *.mat z jedną z opcji np.:

append	dodaje do już istniejącego pliku *.mat nową zmienną
ascii	8 bitowy kod ASCII
mat	binarny plik z rozszerzeniem mat
v4	w formacie dla MATLAB v. 4
v6	w formacie dla MATLAB v.6

load <nazwa pliku>
 wczytuje zmienne z pliku typu *.mat

load <nazwa pliku>.txt
 wczytuje dane z pliku tekstowego

id=fopen(<nazwa pliku>, opcja)
 otwiera plik i zwraca unikatowy dla danego pliku identyfikator (id),
 identyfikator ten powinien być używany we wszystkich operacjach
 związanych z danym plikiem

fclose(id)
 zamyka plik po zakończeniu operacji

cd path
 zmienia aktualny katalog

dir <nazwa katalogu>
 wyświetla w panelu **Command Window** listę plików z danego katalogu

del <nazwa pliku>
 kasuje plik o danej nazwie

what path
 wyświetla zawartość katalogu o podanej ścieżce dostępu.

2.7. Zarządzanie pamięcią

Polecenia wykorzystywane do zarządzania pamięcią:

clear x	usuwa zmienną x z pamięci
clear all	usuwa wszystkie zmienne z pamięci
clear functions	usuwa wszystkie skompilowane m – funkcje i mex – funkcje z pamięci
clear global	usuwa zmienne zadeklarowane jako globalne
pack	służy do przeprowadzenia defragmentacji pamięci – jeśli podczas przeprowadzanych obliczeń mamy do czynienia z macierzami o dużych rozmiarach lub wielowymiarowymi to pamięć może ulec fragmentacji, co może być powodem zwolnienia obliczeń oraz komunikatami błędów

who	polecenie to wyświetla w panelu Command Window wszystkie zmienne z pamięci roboczej
whos	polecenie to wyświetla w panelu Command Window wszystkie zmienne z pamięci roboczej wraz z rozmiarami i typem

2.8. Tworzenie skryptów i funkcji

2.8.1. Skrypt a funkcja

Skrypt jest to m-plik zawierający program napisany w języku MATLABa, nieposiadający żadnych wejść ani wyjść i operujący na zmiennych zawartych w pamięci roboczej. Postać skryptu będą zwykle miały proste programy zawarte w tej książce.

Wyrażenie może zajmować całą linię programu. Jeśli chcemy umieścić w jednej linii kilka wyrażeń oddzielamy je średnikiem lub przecinkiem np.

```
y=x.^2; y2=y./(y+2); xy=x*y;
plot(x,y), grid on, text(20,10,'X')
```

Należy jednak pamiętać, że wyrażenia niezakończone znakiem średnika będą drukować swoje wartości w oknie **Command Window**. Dlatego w pierwszej linijce wybrano średniki a w drugiej przecinki, gdyż polecenia w drugiej linijce nie generują wyników.

Warto umieszczać w kodzie komentarze (po znaku %), które ułatwiają zrozumienie pisanego kodu. Dodatkowo komentarz umieszczony w pierwszej linijce programu jest wyświetlany po wpisaniu **help** <nazwa pliku>.

Należy pamiętać, że MATLAB pobiera zmienne z pamięci roboczej, skrypt może celowo korzystać z tych zmiennych wytworzonych wcześniej, ale lepiej zawsze zwalniać pamięć roboczą przed uruchomieniem skryptu. Z tego względu nie należy też zapisywać skryptu na dysku z nazwą identyczną do nazwy wytwarzanej przezeń zmiennej. Powoduje to bowiem konflikt.

W przypadku większych, bardziej skomplikowanych programów, zwykle wyodrębnia się z nich mniejsze, bardziej funkcjonalne i przejrzyste części zwane funkcjami. Zadaniem funkcji jest wykonanie pewnej sekwencji kodu, które zwykle powtarza się w wielu miejscach programu. Jak mówią programiści – jeśli dana sekwencja wykonywana będzie więcej niż dwa razy to już opłaca się napisać dla niej funkcję. Funkcje mogą posiadać listę parametrów wejściowych i wyjściowych. Mogą one posiadać własne zmienne dostępne tylko w ich wnętrzu i/lub operować na zmiennych dostępnych w całym programie.

Jeśli rozwiązujemy większy problem wtedy zwykle będziemy mieli jeden program (skrypt) główny i wiele plików funkcji, z których będzie on korzystał. Z uwagi na dużą liczbę wytworzonych plików, pracując z MATLABem, najlepiej utworzyć dla każdego nowego programu nowy katalog plików.

2.8.2. Budowa funkcji

Budując funkcję w MATLABie, zwykle otwieramy nowy m-plik (inne przypadki omówimy dalej). Pamiętajmy, że nazwa pliku powinna być identyczna jak nazwa budowanej funkcji (mają one automatycznie przydzielane rozszerzenie `.m`). Jeśli tak nie będzie, to w programie zewnętrznym funkcja będzie dostępna wyłącznie pod nazwą pliku. Plik zawierający funkcję składa się z następujących części:

- Nagłówek
W pierwszym wierszu należy rozpocząć deklarację funkcji poprzez słowo kluczowe `function`, podać jej identyfikator np. `mojafunkcja` oraz argumenty wejściowe `[a,b,c]` i wyjściowe `[x,y]`, np.
`function [x,y]=mojafunkcja(a,b,c)`.
Może istnieć funkcja bez parametru wyjściowego np. `function r=rzad(M)` wtedy kiedy operacja wykonywana modyfikuje wartość zmiennej globalnej. Może też istnieć funkcja bez argumentów wejściowych np. `function r`. Wtedy pełni ona rolę znanej w innych językach programowania procedury. Działanie procedury może polegać np. na wykonaniu wykresu czy zapisaniu pewnych zmiennych na dysku. Wszystkie argumenty użyte do budowy funkcji są jej parametrami formalnymi i nie mają przypisanej wartości.
- Deklaracja zmiennych globalnych, tzn. tych które występują w głównym programie np. `global dp mi_p z`
- Deklaracja (inicjalizacja) zmiennych lokalnych, np. `w=2; ro=3; s=23; z=43;`
Wszystkie zmienne użyte wewnątrz funkcji poza globalnymi są lokalne w danej funkcji. Te zmienne mogą mieć nowe nazwy lub te same, jak zmienne istniejące w programie głównym. Jeśli dana lokalna zmienna ma tę samą nazwę co inna zmienna w programie głównym, to operacje na niej nie wpływają w żaden sposób na wartość zmiennej w programie głównym. Wartości zmiennych lokalnych nie są umieszczane w pamięci roboczej.

- Instrukcje, np.
 $y = w * r_o / s + a + b * c;$
 $x = 23 * c / b;$

Jeżeli funkcja ma parametry wyjściowe, musi nastąpić podstawienie wartości pod te parametry.

Kończąc budowanie funkcji, musimy ją zapisać w m-file. Pamiętajmy, aby nazwy funkcji były takie same jak nazwy plików, w których są zapisane. Taka systematyka bardzo ułatwi pracę z wieloma funkcjami.

Uwaga: wewnątrz funkcji może być zagnieżdżona (wbudowana) inna funkcja, wtedy jest ona podrzędna i dostępna tylko wewnątrz tej funkcji, w której jest zagnieżdżona. Funkcja podrzędna może być umieszczona w dowolnym miejscu funkcji nadrzędnej. Zarówno funkcja nadrzędna jak i podrzędna muszą być wtedy zakończone słowem `end`.

2.8.3. Wywołanie funkcji

Wywołując funkcję, zwykle podstawiamy jej wartość pod zmienną np. $[w,v] = \text{mojafunkcja}(x,y,z)$. W tym wywołaniu parametry x,y,z mają wartości, które mogą być inne w każdym wywołaniu. Noszą więc nazwę parametrów aktualnych. Można też w wywołaniu użyć wartości zamiast zmiennych np. $[x,y] = \text{mojafunkcja}(2,4,7)$. Funkcja może też dokonywać operacji na zmiennych globalnych znajdujących się w pamięci roboczej.

Przykład 2.1. ¹Przykład budowy funkcji

Deklaracja funkcji obliczającej ze wzorów analitycznych pierwiastki równania kwadratowego i jej wywołanie².

p2_1.m

```
%Przykład 2.1 Budowa i użycie funkcji
function r_kwad
%wywołanie funkcji z argumentami aktualnymi
row_kwad(1,-5,1);
```

¹ Plik **p2_1.m** z przykładem znajduje się w katalogu **PRZYKŁADY/Przykład 2-1**

² Uwaga: instrukcja `if` opisana jest w sekcji 2.9.1

```
% deklaracja funkcji z argumentami formalnymi
function x=row_kwad(a,b,c)
delta=b.^2-4*a*c;
if delta > 0
    x=[(-b-sqrt(delta))./(2*a),
        (-b+sqrt(delta))./(2*a)]
elseif delta==0
    x=b/(2*a)
else
    x='dziedzina liczb zespolonych'
end
```

Rezultat 2.1

```
x=
    0.2087
    4.7913
```

Po wywołaniu funkcji z argumentami wejścia w oknie **Command Window** otrzymamy wynik operacji podstawiony pod zmienną domyślną **ans**. Możemy również wartość zwracaną przez funkcję podstawić pod zmienną, pamiętając ile zmiennych jest zwracanych np. `[x,y]= mojafunkcja(2,4,7)`.

Charakterystyczną cechą funkcji w MATLABie, w odróżnieniu od niektórych języków programowania, jest możliwość umieszczenia mniejszej niż zadeklarowana liczba argumentów wejściowych w wywołaniu, np. powyższą funkcję można wywołać z jednym, dwoma lub trzema argumentami. Oczywiście takim wywołaniom trzeba przypisać odpowiednie działania wewnątrz funkcji. Służy do tego zmienna **nargin** oznaczająca aktualną liczbę parametrów w wywołaniu. Pokazano to w przykładzie 2.2. W zależności od liczby parametrów w wywołaniu rozwiązywane jest równanie kwadratowe $ax^2 + bx + c = 0$, liniowe $ax + b = 0$ lub trywialne $ax = 0$.

Przykład 2.2. Wykorzystanie parametru nargin

```
 p2_2.m
%Przykład 2.2 Wykorzystanie parametru nargin
function rown1i2
clear all; close all; clc
%wywołanie funkcji z argumentami aktualnymi
row_kwad(1,-5,1);
row_kwad(1,-5);
row_kwad(1);
% deklaracja funkcji z argumentami formalnymi
```



```
function x=row_kwad(a,b,c)
if nargin==3
    delta=b.^2-4*a*c;
    if delta > 0
        x=[(-b-sqrt(delta))./(2*a),
            (-b+sqrt(delta))./(2*a)]
    elseif delta==0
        x=b/(2*a)
    else
        x='dziedzina liczb zespolonych'
    end
elseif nargin==2
    x=-b/a
else
    x=0
end;
```

Rezultat 2.2

```
x =    0.2087
      4.7913
x =    5
x =    0
```

2.8.4. Umieszczanie wielu funkcji w jednym pliku

W przypadku skryptów, które posługują się dużą liczbą funkcji, umieszczanie ich osobno w m-plikach może być pracochłonne. Można to ominąć i umieścić je wszystkie w jednym skrypcie, ale wtedy należy na początku skryptu należy podać słowo kluczowe **function** a po nim nazwę, pod którą jest on zapisany na dysku (bez argumentów). Kolejne funkcje zadeklarowane pod pierwszą deklaracją funkcji w takim pliku są traktowane jako podrzędne (podfunkcje), i nie są dostępne z zewnątrz. Muszą być one wtedy kończone słowem kluczowym **end**. To słowo należy wtedy również umieścić na końcu pliku. Powstaje więc wykonywalny plik funkcyjny. Zostało to użyte w Przykładzie 2.1 i 2.2. Umieszczanie w skrypcie kilku funkcji podrzędnych ilustruje ten program:

```
function wyrażenie
%ten skrypt oblicza wartość złożonego wyrażenia
%wykorzystując dwie funkcje podrzędne
x=0.5;
```

```

z=w1(x)
function w=w1(x)
    w=(x^2+x^3)/(x^(1/2)+x^(1/3));
end
function w=w2(x)
    w=(exp(x)+exp(1/x))/(exp(x)-exp(1/x));
end
w=sin(w1(x)/w2(x)+w2(x))
end

```

W wyniku otrzymujemy

```

z =
    0.2499
w =
   -0.9869

```

Jeśli kilka różnych funkcji obliczać będziemy jednocześnie dla danego parametru, to wtedy można użyć wielu wyjść z funkcji. Wtedy jednak zmienne lokalne będą wspólne wewnątrz funkcji. Obrazuje to poniższy przypadek.

Funkcja ma postać:

```

function [w,v]=w1i2(x)
%ta funkcja oblicza wartości dwu wyrażeń
w=(x^2+x^3)/(x^(1/2)+x^(1/3));
v=(exp(x)+exp(1/x))/(exp(x)-exp(1/x));

```

skrypt z jej wywołaniem ma postać:

```

%ten skrypt oblicza wartość złożonego wyrażenia
%wykorzystując funkcję zewnętrzną
x=0.5;
[y z]=w1i2(x);
w=sin(y/z+z)

```

Po wywołaniu tego skryptu otrzymamy wynik:

```

w =
   -0.9869

```

2.8.5. Przekazywanie nazwy funkcji jako parametru

MATLAB umożliwia przekazywanie nazwy funkcji jako parametru w wywołaniu. Daje to możliwość wyboru funkcji w zależności od np. wartości parametru wprowadzonego przez użytkownika. Ilustruje to przykład 2.3.

Przykład 2.3. Baza danych – przekazywanie nazwy funkcji

Stworzono niewielką bazę danych fizykochemicznych w postaci przedstawionego tu pliku w Excelu (plik **Baza1.xls**).

Baza danych właściwości fizykochemicznych					
	Gęstość⇒			Ciepło wł.⇒	
Związek 1	651,4	-0,0264	1,12	0,014	0,00028
Związek 2	762,7	-0,082	1,01	0,011	0,0017
Związek 3	998,2	-0,018	2,02	0,021	0,0012
Związek 4	872,2	-0,045	0,98	0,016	0,0032
Związek 5	561,2	-0,014	0,92	0,018	0,0025
Związek 6	1221,3	-0,022	1,17	0,011	0,0016

Baza danych zawiera sześć związków chemicznych i współczynniki równań aproksymacyjnych dwu właściwości dla każdego z nich. Plik zostaje wczytany do pamięci roboczej za pomocą zbudowanej w tym celu funkcji `loadbase`. Wykorzystano tu polecenie `xlsread(' <nazwa>')` (zob. sekcja 5.1.2), które wczytuje dane z pliku Excela (pomijając kolumny i wiersze niezawierające danych liczbowych) i umieszcza je w tablicy `dane`.

loadbase.m

```
function loadbase(bnazwa);  
global dane;  
%Jednorazowe wczytanie do pamięci wybranej bazy danych  
%z pliku Excela  
dane=xlsread([bnazwa, '.xls']);
```

Uwaga: nawiasy `[ ]` w wywołaniu funkcji `xlsread` scalają dwa łańcuchy: `na-`
`zwa` i `' .xls'` (konkatenacja pozioma).

Plik **mybase.m** zawiera funkcje obliczające odpowiednią właściwość (metody). Zawiera on wszystkie formuły służące do obliczania właściwości (jako podfunkcje). Rozwiązanie takie pozwala na spakowanie wszystkich metod do jednego pliku.

mybase.m

```
function out=mybase(nazwa,i,t)
out=feval(nazwa,i,t);

function y = dens(i,t)
global dane
y=dane(i,1)+t*dane(i,2);

function y = hcap(i,t)
global dane
y=dane(i,3)+dane(i,4)*t+dane(i,5)*t^2;
```

Po uruchomieniu pliku ze skryptem o nazwie **usebase.m** (to główny plik przykładu – poniżej) następuje wczytanie wybranej bazy a następnie obliczenie żądanej właściwości. W wywołaniu funkcji **mybase** zostaje przekazana nazwa wybranej funkcji do obliczania właściwości oraz aktualne parametry. Nazwa ta zostaje zinterpretowana w pliku **mybase.m** za pomocą polecenia **feval**, które wybiera wywołaną funkcję. Wynik przypisany do wyjścia **out** wraca do pliku **usebase**.

usebase.m

```
clear all; close all; clc
%skrypt, który ilustruje wykorzystanie bazy danych

loadbase('Baza1');

i=2;           %Numer związku, którego właściwość odczytamy
t=20;          %Temperatura, w której obliczymy właściwość

%odczytanie gęstości
rho=mybase('dens',i,t);
rho

%odczytanie ciepła właściwego
cp=mybase('hcap',i,t);
cp
```

Rezultat 2.3

```
rho =    761.0600
cp  =     1.9100
```

Warto zauważyć, że odwołanie do nazwy funkcji w wywołaniu wymaga umieszczenia nazwy funkcji w apostrofach. Można tak postępować, jeżeli funkcja znajduje się w tym samym pliku. Jeśli funkcja znajduje się w osobnym pliku, można posłużyć się tzw. uchwyttem funkcji @ (ang. *handle*) np. @dens zamiast 'dens'. Tą metodą posługuje się wiele funkcji wbudowanych w MATLABa.

2.8.6. Funkcje anonimowe

Istnieje możliwość uproszczonego sformułowania funkcji zajmującej tylko jedną linię kodu. Może ono występować w dowolnym skrypcie, jak też być wprowadzane w linii poleceń. Tworząc funkcję anonimową, definiuje się w istocie zmienną będącą uchwyttem tej funkcji za pomocą wyrażenia

`fhandle = @( <lista argumentów> ) <wyrażenie>`

W ten sposób można zdefiniować funkcję np. przeliczającą temperaturę ze stopni Fahrenheita na stopnie Celsjusza

```
FtoC=@(tF) 5/9*(tF-32)
```

Po wpisaniu tej funkcji w **Command Line** można ją wykorzystać do przeliczenia temperatury 100 °F na °C

```
FtoC(100)
ans =
    37.7778
```

W przykładzie 2.4 pokazanym poniżej obliczymy jednostkę przenikania masy w górnej części kolumny rektyfikacyjnej przy destylacji układu dwuskładnikowego (*Serwiński, 1971*).

$$NTU = \int_{y_F}^{y_D} \frac{dy}{y^* - y} \quad (2.1)$$

Równowagę ciecz-para policzymy na podstawie lotności względnej α

$$y^*(x) = \frac{\alpha}{\alpha + \frac{1-x}{x}} \quad (2.2)$$

ślusznego dla $x > 0$. Równanie górnej linii operacyjnej wykorzystamy do obliczenia x dla danego y . Liczba powrotu R jest dana. Dane są zawarte w programie.

$$x = \frac{R+1}{R} \left(y - \frac{1}{R+1} x_D \right) \quad (2.3)$$

Przykład 2.4. Wykorzystanie funkcji anonimowych do obliczenia NTU

p2_4.m

```
%Przykład 2.4
%Użycie funkcji anonimowej
R=3;
alfa=2.5;
xD=0.95;
yD=0.95;
yF=0.5;
yeq=@(x) alfa./(alfa-(1-x)./x);
xg=@(y) (R+1)./R*(y-1./(R+1)*xD);
fpodcalkowa=@(y) 1./(yeq(xg(y))-y);
% funkcja quad oblicza całkę oznaczoną
N=quad(fpodcalkowa,yF,yD)
```

Rezultat 2.4

```
N =
    1.2776
```

Wyrażenie występujące w funkcji anonimowej może być wywołaniem innej funkcji, w tym także funkcji anonimowej jak np.

```
f1=@(x) f2(x)
```

Powyższe wywołanie nie ma większego sensu, gdy listy argumentów są identyczne, lecz staje się bardzo użyteczne, gdy listy argumentów są różne np.

```
f 1=@(x) f2(x,a,b,c)
```

gdzie a , b , c to zmienne występujące w programie głównym. Pozwala to na przekazanie dodatkowych parametrów do wnętrza funkcji, w wywołaniu której nie ma na nie miejsca jak np. w solverze równań `fzero` czy układów równań `fsolve` (zobacz przykład 2.5c)

2.8.7. Przekazywanie dodatkowych parametrów do funkcji wbudowanych

MATLAB posiada wiele wbudowanych funkcji służących do rozwiązywania typowych zagadnień numerycznych. Funkcje te posiadają sztywną składnię, nie zawsze pozwalającą na przekazywanie wszystkich niezbędnych parametrów. Przekazywanie dodatkowych parametrów można zorganizować na trzy sposoby:

- przez zmienne globalne,
- przez funkcje anonimowe,
- przez funkcje zagnieżdżone.

Sposoby te omówimy na przykładzie funkcji `fsolve` (dostępna w skrzynce Optimization Toolbox) służącej do poszukiwania miejsc zerowych równań (zob. Rozdz. 6). Rozwiązując będziemy równanie Darcy-Weissbacha, poszukując prędkości cieczy u w rurociągu, dla której znany jest spadek ciśnienia Δp na jego końcach. Rurociąg jest poziomy i ma stałą średnicę d . Wykonany jest z rury o długości L i szorstkości ε . Wg równania Darcy-Weissbacha spadek ciśnienia wynosi wtedy:

$$\Delta p = \lambda(Re, \varepsilon) \frac{L}{d} \frac{u^2}{2} \rho \quad (2.4)$$

gdzie

$$\lambda = \frac{1.325}{\left[\ln \left(\frac{\varepsilon}{3.7} + \frac{5.74}{Re^{0.9}} \right) \right]^2} \quad (2.5)$$

oraz $Re = u d \rho / \mu$. Tak więc rozwiązywane równanie ma postać:

$$\Delta p - \lambda \left(\frac{u d \rho}{\mu}, \varepsilon \right) \frac{L}{d} \frac{u^2}{2} \rho = 0 \quad (2.6)$$

Równanie (2.6) jest nieliniowe ze względu na u i wymaga rozwiązania numerycznego. Wykorzystamy funkcję `fsolve` o wywołaniu

```
[u fval] = fsolve(fun,u0)
```

gdzie u – znaleziony pierwiastek, $fval$ – wartość funkcji w miejscu zerowym, fun – lewa strona równania w funkcji u , $u0$ – wartość startowa u .

Do funkcji `fun` (lewa strona równania (2.6)) należy jednak przekazać oprócz u również wszystkie pozostałe parametry. Poniższe przykłady ilustrują jak to

zrobić wg wymienionych tu sposobów. Dla wszystkich wariantów napisana została wspólna funkcja do obliczania współczynnika oporu zapisana jako **lambda.m**

lambda.m

```
%współczynnik oporu przyściennego
%wzór ważny dla 5000<Re<10^8 i 10^-6<eps<10^-2
function y=lambda(Re,eps)
l=log(eps./3.7+5.74/Re^0.9);
y=1.325/l^2;
```

Przykład 2.5. a Przekazywanie parametrów przez zmienne globalne

Funkcja lewej strony rozwiązywanego równania ma postać (plik **DWg.m**)

DWg.m

```
%równanie Darcy-Weisbacha
function y=DWg(u)
global eps d L dp rho mu
Re=u*d*rho./mu;
y=dp-lambda(Re,eps)*L./d*u^2./2*rho;
```

a główny program (plik **rozwDWg.m**)

rozwDWg.m

```
%Przykład 2-5a
%Rozwiązanie równania Darcy-Weissbacha
%z przekazywaniem parametrów przez zmienne globalne
clear all
global eps d L dp rho mu
d=0.01;
L=100;
dp=98000;
rho=1000;
mu=10^-3;
eps=2*10^-3;
u0=1; %punkt startowy
[u fval]=fsolve(@DWg,u0)
```

Jak widać funkcja DWg przekazuje przez listę argumentów jedynie zmienną u, gdyż tak wymaga składnia funkcji **fsolve**. Pozostałe argumenty pobiera ze zmiennych **global**. Przekazywanie parametrów w ten sposób jest o tyle niewygodne, że nazw tych zmiennych nie można używać lokalnie do innych celów.

Przykład 2.5 b Przekazywanie parametrów przez funkcję anonimową

Funkcja lewej strony rozwiązywanego równania ma postać (plik **DWp.m**)

 DWp.m

```
%równanie Darcy-Weisbacha
function y=DWp(u,eps,d,L,dp,rho,mu)
Re=u*d*rho./mu;
y=dp-lambda(Re,eps)*L./d*u^2./2*rho;
```

a główny program (plik **rozwDWp.m**)

 rozwDWp.m

```
%Przykład 2-5b
%Rozwiązanie równania Darcy-Weisbacha
%z przekazywaniem parametrów za pomocą funkcji anonimowej
clear all
d=0.01;
L=100;
dp=98000;
rho=1000;
mu=10^-3;
eps=2*10^-3;
u0=1; %punkt startowy
f=@(u) DWp(u,eps,d,L,dp,rho,mu);
[u fval]=fsolve(f,u0)
```

Przykład 2.5 c Przekazywanie parametrów przez funkcję zagnieżdżoną

Funkcja lewej strony rozwiązywanego równania ma postać (plik **DWn.m**)

 DWn.m

```
%równanie Darcy-Weisbacha
function [x,fval]=DWn(u0,eps,d,L,dp,rho,mu)
[x,fval]=fsolve(@DW,u0);
function y=DW(u)
    Re=u*d*rho./mu;
    y=dp-lambda(Re,eps)*L./d*u^2./2*rho;
end
end
```

a główny program (plik **rozvDWn.m**)

rozvDWn.m

```
%Przykład 2-5c
%Rozwiązanie równania Darcy-Weisbacha
%z przekazywaniem parametrów przez funkcję zagnieżdżoną
clear all
d=0.01;
L=100;
dp=98000;
rho=1000;
mu=10^-3;
eps=2*10^-3;
u0=1;          %punkt startowy
[u fval]=DWn(u0,eps,d,L,dp,rho,mu)
```

Rezultat 2.5

W każdym z powyższych przypadków otrzymujemy ten sam wynik

```
u      =      0.7311
fval =      0
```

2.8.8. Funkcje matematyczne MATLABa przydatne w obliczeniach inżynierskich

Spośród wielu wbudowanych funkcji wybrano kilka najbardziej przydatnych w obliczeniach stosowanych w inżynierii procesowej. Zobacz więcej w menu pomocy w zakładce Functions/ Mathematics.

Tablica 2.1. Wybrane funkcje matematyczne MATLABA

Funkcja	Przykład wywołania	Działanie
<i>Różniczkowanie numeryczne</i>		
polyder	P=polyder(p)	Analityczna pochodna wielomianu – zwraca wektor współczynników wielomianu pochodnej, p – wektor współczynników wielomianu (od najwyższej potęgi zaczynając)
gradient	[FX]=gradient(F) [FX,FY]=gradient(F) itd.	Gradient – wektor pochodnej funkcji F po wektorze jej argumentu x, w przypadku funkcji wielowymiarowych po jej kolejnych parametrach

diff	Z=diff(X)	Zwraca wektor różnic między n elementami wektora X. Jeśli elementy wektora są równo oddalone o Δh wtedy wektor Z podzielony przez Δh daje wektor przybliżonych n-1 wartości pochodnej.
del2	Z=del2(U)	Jeśli U jest macierzą wartości funkcji macierzy dwu zmiennych X, Y to Z jest laplasjanem U: $\nabla^2 U = \left(\frac{\partial^2 U}{\partial X^2} + \frac{\partial^2 U}{\partial Y^2} \right)$
<i>Całkowanie numeryczne</i>		
quad	Z=quad(fun,a,b)	Całka oznaczona metodą Simpsona funkcji fun w przedziale a,b
polyint	P=polyint(p)	Analityczna całka oznaczona wielomianu, P jest wektorem współczynników wielomianu całki.
trapz	Z=trapz(X,Y)	Całka oznaczona z funkcji danej wektorami X i Y
<i>Funkcje specjalne</i>		
erf	Z=erf(X)	Funkcja błędu
erfc	Z=erfc(X)	Dopełnienie funkcji błędu
gamma	Z=gamma(X)	Funkcja gamma
besselj	Z=besselj(n,X)	Funkcja Bessela pierwszego rodzaju rzędu n.

2.9. Instrukcje

Przedstawione tu instrukcje zapewniają przepływ informacji w programie. Należy jednak pamiętać, że język MATLABa jest w pełni obiektowy i można w nim wykorzystywać możliwości programowania obiektowego. Tego aspektu tu nie poruszono.

2.9.1. Instrukcje warunkowe

Instrukcja warunkowa **if** sprawdza warunek logiczny. Warunek może być złożony – do jego budowy stosuje się operatory relacji i operatory logiczne. Jeżeli warunek jest spełniony (wyrażenie logiczne ma wartość 1 co oznacza prawdę), to jest wykonywana instrukcja, która występuje zaraz po warunku lub cały blok instrukcji. Jeżeli natomiast warunek nie jest spełniony, to ciąg instrukcji po **if** nie będzie wykonany, ale jeżeli zdefiniujemy **else** jest wykonywana instrukcja lub blok zaraz po **else**. W MATLABie mamy jeszcze możliwość zdefiniowania warunku pośredniego **elseif**. Oto składnia instrukcji:

```

if wyrażenie_logiczne
    instrukcje
elseif wyrażenie_logiczne
    instrukcje
else
    instrukcje
end

```

2.9.2. Instrukcje iteracyjne

Instrukcja **while** jest pętlą wykonującą cyklicznie umieszczony wewnątrz kod, pod warunkiem że wyrażenie warunkowe zwraca wartość **TRUE**. Składnia instrukcji:

```

while wyrażenie
    instrukcje
end

```

Przed każdym wykonaniem instrukcji następuje sprawdzenie warunku. Pętla zostanie przerwana, gdy wartość warunku będzie równa **FALSE**. W celu przerwania wykonywania instrukcji należy wywołać instrukcję **break** wewnątrz instrukcji **while**. Przestrzegamy przed pętlą nieskończoną, gdy warunek jest źle sformułowany. Jeśli program zawiesi się, można go przerwać przez **CTRL+C**.

Instrukcja **for** jest pętlą o liczonej liczbie wykonania. Składnia instrukcji:

```

for indeks=<zakres>
    instrukcje
end

```

Zakres to wektor wartości generowany za pomocą operatora **:** (ang. *colon operator*). Zapis **a:b** tworzy ciąg wartości **a**, **a + 1**, **a + 2**...**b**, a zapis **a:krok:b** tworzy ciąg **a**, **a + krok**, **a + 2*krok**,...**b**. Zakres może być również wcześniej zdefiniowanym wektorem.

Dla każdego elementu tego ciągu lub wektora wykonywany jest zestaw instrukcji poniżej. Wewnątrz pętli można wykorzystać aktualną wartość zmiennej **indeks**, lecz nie wolno zmieniać jej wartości. Po wyczerpaniu wszystkich wartości ciągu pętla jest przerywana.

2.9.3. Instrukcja wyboru

Struktura instrukcji wyboru **switch** składa się z serii przypadków **case** i opcjonalnego przypadku **otherwise**. Po słowie kluczowym **switch** następuje wyrażenie sterujące. Wartość tego wyrażenia jest porównywana z każdą z etykiet **case**. Jeśli wynik porównania jest prawdziwy wykonane zostaną instrukcje po tym przypadku **case**. Instrukcja **break** powoduje, że wykonanie programu jest kontynuowane od pierwszej instrukcji po strukturze **switch**. Jeżeli nie byłoby instrukcji **break** to po wystąpieniu dopasowania zostaną wykonane instrukcje wszystkich pozostałych przypadków **case**. Jeśli nie wystąpi dopasowanie w żadnym z przypadków **case**, zostanie wykonany przypadek domyślny **otherwise**. Każdy przypadek **case** może zawierać jedną lub więcej instrukcji. Składnia instrukcji jest następująca:

```
switch wyrażenie_logiczne
  case wartość1
    instrukcje
  case wartość2
    instrukcje
  otherwise
    instrukcje
end
```

2.9.4. Instrukcje wejścia/wyjścia w oknie Command Window

Wykonując obliczenia zarówno w trybie interaktywnym jak i za pomocą uruchamianych programów, często wprowadzamy do pamięci dane za pomocą klawiatury, jak i wyprowadzamy na ekran wyniki obliczeń.

Wprowadzanie zmiennych w trybie interaktywnym omówiono w punkcie 2.3.1.

W programach posługujemy się instrukcjami:

input('p')	polecenie wprowadzenia danej z klawiatury. Na ekranie pojawia się napis (łańcuch) p i program czeka na wprowadzenie danej. Wprowadzenie danej odbywa się przez wpisanie jej wartości i wciśnięcie Enter. Dokonana zostaje konwersja wprowadzonego łańcucha na wartość zmiennej, pod którą jest podstawiona funkcja input . Jeśli wystąpi błąd konwersji, program zatrzymuje się i wyświetlany jest komunikat błędu.
-------------------	--

`input('p','s')` działa jak wyżej, ale powoduje wczytanie łańcucha znaków ('s' – to opcja tej instrukcji).

`menu('Nazwa','o1','o2','o3',...)`

pozwała na wybór jednej z kilku opcji. Wybranej opcji przypisywana jest wartość liczby całkowitej odpowiadającej pozycji tej opcji na liście.

`disp(x)` wyświetla na ekranie macierz lub łańcuch znajdujący się w zmiennej `x` bez wyświetlania jej nazwy.

Przykładowe wczytanie danej z klawiatury

```
T=input('Podaj temperaturę pomiaru: ');  
Imie=input('Imię studenta:','s');
```

Przykładowy wybór opcji z menu

```
i_font = menu('Wybierz Kolor','Czarny','Biały','Czerwony');
```

Przykładowe wyświetlenie wyniku na ekranie

```
disp('Temperatura wynosi'); disp(T);
```

Okno **Command Window** jest przewijane w miarę, jak pojawiają się na nim kolejne wyniki. Można cofnąć je po zakończeniu działania programu za pomocą suwaka, można też zatrzymywać wydruki po zapełnieniu ekranu. Tę opcję uruchamia się komendą **more on**. Wtedy po zapełnieniu ekranu należy wcisnąć **Enter**, by pokazał się następny. Opcję tę wyłącza się poleceniem **more off**.

2.9.5. Uruchamianie programów

Pisanie nowego programu rozpoczyna się od wybrania opcji **File>New** z menu głównego. Spośród wielu typów plików, które można wybrać interesuje nas typ **Script** lub **Function**. Po wybraniu otwierane jest nowe okno edytora, w którym wpisujemy program pod wybraną nazwą np. **Wykres1**. Po ukończeniu plik zapisujemy na dysku w katalogu roboczym lub wybranym katalogu docelowym. Jeśli plik będzie uruchamiany w tej samej sesji należy zmienić nazwę katalogu roboczego na docelowy, który stanie się roboczym. Zapisany plik uruchamiamy bądź z edytora opcją **Debug>Run**, bądź wybierając ten plik z katalogu roboczego.

W trakcie uruchomienia mogą pojawić się błędy np.

```
??? Error: File: Wykres1.m Line: 7 Column: 11  
Unexpected MATLAB expression.
```

Należy po wejściu w edytora dokonać odpowiednich poprawek w programie i uruchomić go ponownie. Przy okazji uruchomienia następuje również zapisanie poprawionej wersji m-pliku w katalogu roboczym.

W trakcie poprawiania kolejnych błędów (ang. *debugging*) zachodzi potrzeba sprawdzania aktualnego stanu zmiennych po wykonaniu danej liniiki kodu. Można wtedy umieścić po tej linii polecenie `keyboard`. Po napotkaniu tej instrukcji program przerywa działanie i przekazuje kontrolę klawiaturze. W oknie **Command Window** pojawia się nowe ponaglenie postaci

```
k>>
```

Można teraz odczytać wartość aktualną dowolnej zmiennej z pamięci roboczej. Wpisanie `return` powoduje powrót do wykonywania programu.

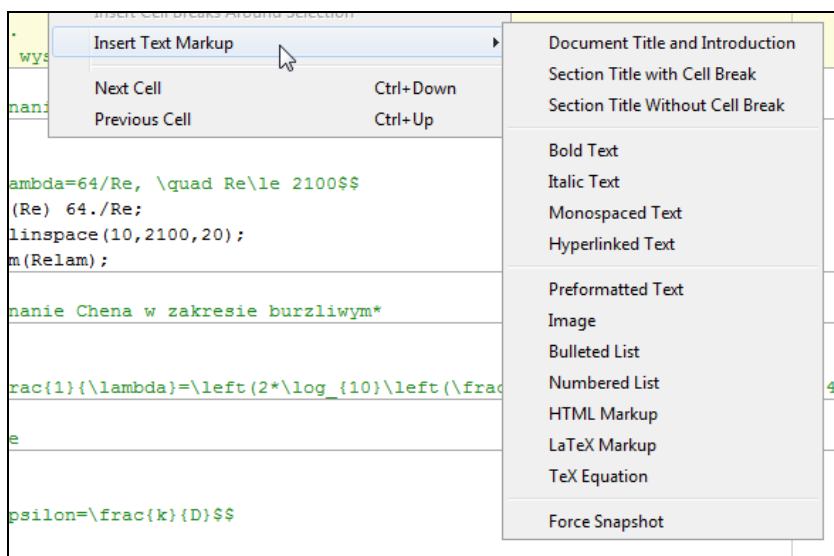
W MATLABie istnieje wyspecjalizowany debugger, którego polecenia można dołączać do programu. Nauka jego obsługi przekracza jednak zakres tej pracy.

2.9.6. Publikowanie raportów

Zwykle do prezentacji wyników obliczeń wystarcza przekopiowanie zawartości **Command Window** i uzupełnienie jej kopiami wykresów generowanych przez MATLABa. Jednak do tworzenia bardziej eleganckich prezentacji zarówno samego programu jak i jego wyników służy publikowanie raportów. Raporty można tworzyć w formacie MS Word[®], Power Pointa[®], pdf, html i innych. Domyślnym formatem jest html.

Publikowanie zezwala na umieszczenie w raporcie takich elementów, jak: tytuł raportu, nagłówki sekcji, listy wypunktowane, spis treści, równania, rysunki i inne. Pozwala też zmieniać krój czcionki w każdym z tych elementów, wytłuszczać druk lub pisać go kursywą.

Do wstawiania elementów tekstowych służą polecenia z menu Edytora plików. W tym celu wybieramy pozycję Cell/Insert Text Markup. Ma ona opcje pokazane na Rys. 2.6.



Rys. 2.6. Opcje wstawiania tekstu Insert Text Markup

Należy uważać, aby nie zmieniać znaków % i \$, które znajdują się we wstawionych przez MATLAB w nasz program wierszach. Służą one formatowaniu tzw. komórek programu i nie są traktowane jak zwykłe znaki komentarza.

Do edycji równań służy zestaw symboli edytora LaTeX. Ten edytor do budowy wyrażeń matematycznych posługuje się wyłącznie znakami z klawiatury. W zestawieniu w tablicy 2.1 podano kilkanaście przydatnych symboli. Poprzedza się je zawsze znakiem \ (backslash).

Tablica 2.2. Przydatne symbole LaTeXa

Symbol	Przykład	Rezultat
\nazwa greckiej litery	\beta , \Gamma	β Γ
\^{}{}	\a^{2+3}	a^{2+3} – wykładnik
{}{}	\a{w}	a_w – indeks dolny
\frac {} {}	\frac {a}{b}	$\frac{a}{b}$ – ułamek
\circ	\circ C	$^{\circ}\text{C}$ – znak stopnia
\partial	\partial T	∂T – znak pochodnej cząstkowej
\infty	\infty	∞ – znak nieskończoności

<code>\int</code>	<code>\int y dy</code>	$\int y dy$ – znak całki nieoznaczonej
<code>\int_x1^x2</code>	<code>\int_0^\infty \frac{1}{y} dy</code>	$\int_0^\infty \frac{1}{y} dy$ – znak całki oznaczonej
<code>\pm</code>	<code>a\pm b</code>	$a \pm b$
<code>\approx</code>	<code>a\approx b</code>	$a \approx b$
<code>\neq</code>	<code>a\neq b</code>	$a \neq b$
<code>\leq</code>	<code>a\leq 5</code>	$a \leq 5$ – mniejsze równe
<code>\geq</code>	<code>a\geq 5</code>	$a \geq 5$ – większe równe
<code>\nabla</code>	<code>\nabla T</code>	∇T – znak nabla
<code>\rightarrow</code>	<code>H+Cl\rightarrow HCl</code>	$H+Cl \rightarrow HCl$ – znak strzałki
<code>\sum</code>	<code>\sum w</code>	$\sum w$ – znak sumy
<code>\sum {}^{\wedge}{}^{\{}</code>	<code>\sum_{i=1}^n q_{i}</code>	$\sum_{i=1}^n q_i$ – znak sumy z zakresem
<code>\quad</code>	<code>\quad</code>	poczwórna spacja (odstęp)
<code>\sqrt{\}</code>	<code>\sqrt{\pi}</code>	$\sqrt{\pi}$ – pierwiastek kwadratowy
<code>\left (</code>	<code>\left (</code>	duży nawias otwarty
<code>\right)</code>	<code>\right)</code>	duży nawias zamknięty

Wiele innych symboli oraz konstrukcje używane do budowy macierzy i tablic można znaleźć w Internecie. Warto się z nimi zapoznać, gdyż można je używać także do opisu wykresów (por. Rozdz. 4).

Po zakończeniu budowy programu wraz z poleceniami publikowania należy program opublikować, tzn. przekształcić na plik wynikowy. Najprościej wybrać opcję File/Publish <nazwa> z głównego menu Edytora. Przykład publikowania programu zawiera Projekt 2.2

PROJEKT 2.1

Temat: Konstrukcja wykresu Moody'ego wraz z publikacją raportu
Toolbox: -

W katalogu **PROJEKTY/R2/Projekt 2-1** znajduje się plik główny **P2_1.m** oraz katalog html, gdzie znajduje się wynikowy raport. Pliku **P2_1.m** nie należy uruchamiać. Po ewentualnych modyfikacjach należy go opublikować przez wybór opcji File/Publish P2_1.

Uwaga: pewne instrukcje użyte w tym programie są zawarte w dalszych rozdziałach. Równanie Chena pochodzi z pracy *Chen, 1979*.

P2_1.m

```
%% PROJEKT 2.1 WYKRES MOODY'EGO
% Wykres Moody'ego zwany również Harfą Nikuradze
% przedstawia zależność współczynnika oporu przyściennego
% od liczby Reynoldsa dla przepływu w rurze, w zależności od
% szorstkości
% rury.
% (k - wysokość nierówności, D - średnica rury)
%%
% *Równanie w zakresie laminarnym*
%%
%
% 
$$\lambda = 64 / Re, \quad Re \leq 2100$$

llam=@(Re) 64./Re;
Relam=linspace(10,2100,20);
la=llam(Relam);
%%
% *Równanie Chena w zakresie burzliwym*
%%
%
%

$$\frac{1}{\lambda} = \frac{1}{\left(2 \cdot \log_{10} \left( \frac{\epsilon}{7.7056} + \frac{5.0452}{Re} \log_{10} \left( \frac{\epsilon^{1.1098}}{2.8257} + \frac{5.8506}{Re^{0.8981}} \right) \right) \right)^2}$$

%%
% gdzie
%%
%
% 
$$\epsilon = \frac{k}{D}$$

%
```

```

l1=@(Re,ep) (ep.^1.1098)./2.8257+5.8506./Re.^0.8981;
l2=@(Re,ep) ep./3.7056-5.0452./Re*log10(l1(Re,ep));
ltur=@(Re,ep) 1./(2*log10(l2(Re,ep))).^2;
Retur1=linspace(4100,1e4,10); %zagęszczony obszar początkowy Re
Retur2=logspace(4,8,50);      %reszta obszaru Re
Retur=[Retur1 Retur2];
epr=[1e-6 1e-5 1e-4 1e-3 1e-2 1e-1]';
lb=eye(6,60);
for i=1:6
    for j=1:60
        lb(i,j)=ltur(Retur(j),epr(i));
    end
end

%Wykres
loglog(Relam,la,'LineWidth',2);
hold on
loglog(Retur,lb,'LineWidth',2);
grid on;
title('Wykres Moody'ego dla rur szorstkich');
xlabel('Re')
ylabel('\lambda');

```

Wyniki 2.1

Rezultatem publikacji jest plik **P2_1.html**, którego zawartość pokazano poniżej.

PROJEKT 2.1 WYKRES MOODY'EGO

Wykres Moody'ego zwany również Harfą Nikuradze przedstawia zależność współczynnika oporu przyściennego od liczby Reynoldsa dla przepływu w rurze, w zależności od szorstkości rury. (k - wysokość nierówności, D - średnica rury)

Równanie w zakresie laminarnym

$$\lambda = 64/Re, \quad Re \leq 2100$$

```

llam=@(Re) 64./Re;
Relam=linspace(10,2100,20);
la=llam(Relam);

```

Równanie Chena w zakresie burzliwym

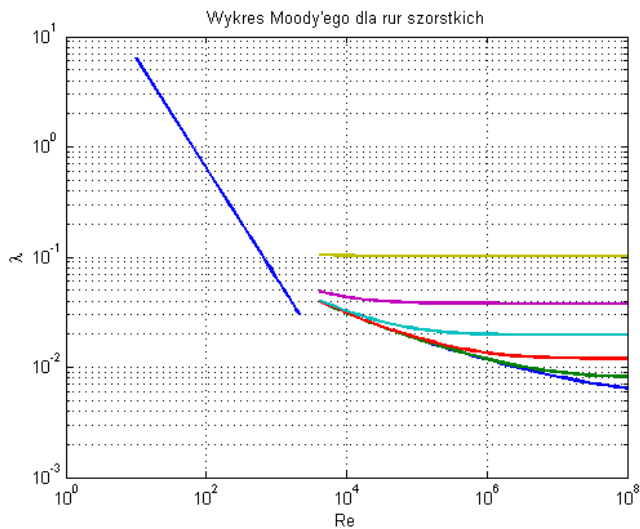
$$\frac{1}{\lambda} = \left(2 * \log_{10} \left(\frac{\epsilon}{7.7056} - \frac{5.0452}{Re} \log_{10} \left(\frac{\epsilon^{1.1098}}{2.8257} + \frac{5.8506}{Re^{0.8981}} \right) \right) \right)^2$$

gdzie

$$\epsilon = \frac{k}{D}$$

```
l1=@(Re,ep) (ep.^1.1098)./2.8257+5.8506./Re.^0.8981;
l2=@(Re,ep) ep./3.7056-5.0452./Re*log10(l1(Re,ep));
ltur=@(Re,ep) 1./(2*log10(l2(Re,ep))).^2;
Retur1=linspace(4100,1e4,10); %zagęszczony obszar początkowy Re
Retur2=logspace(4,8,50);      %reszta obszaru Re
Retur=[Retur1 Retur2];
epr=[1e-6 1e-5 1e-4 1e-3 1e-2 1e-1]';
lb=eye(6,60);
for i=1:6
    for j=1:60
        lb(i,j)=ltur(Retur(j),epr(i));
    end
end

%Wykres
loglog(Retur,lb,'LineWidth',2);
hold on
loglog(Retur,lb,'LineWidth',2);
grid on;
title('Wykres Moody'ego dla rur szorstkich');
xlabel('Re')
ylabel('\lambda');
```



Published with MATLAB® 7.10

Rys. 2.7. Raport wygenerowany w projekcie

LITERATURA

Brzózka J., Dobczyński L., Programowanie w Matlab, Wyd. Mikom, Warszawa 1998.

Chen N.H., 1979, An Explicit Equation for Friction Factor in Pipe, Ind.&Eng. Chem. Fundam., 18, 216-231.

Pratap R., MATLAB 7 dla naukowców i inżynierów, PWN, Warszawa 2007.

Serwiński M., Zasady inżynierii chemicznej, WNT, Warszawa 1971, s. 509.

Stachurski M., Metody numeryczne w programie MATLAB, Wyd. Mikom, Warszawa 2003.

Zalewski A., Cegiela R., Matlab – obliczenia numeryczne i ich zastosowania, Wyd. Nakom, Poznań 2003.

3. OPERACJE NA MACIERZACH

Najczęściej w MATLABie macierz (ang. *matrix*) oznacza prostokątną tablicę wypełnioną liczbami, o wymiarach $n \times m$. Jeśli $n = m$ to mówimy że macierz jest kwadratowa. Zazwyczaj pierwszy indeks elementu macierzy oznacza numer wiersza danej macierzy, a drugi numer kolumny. Wektor to macierz jednokolumnowa lub jednowierszowa. Skalar to macierz o jednym wierszu i jednej kolumnie.

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix}$$

Indeksowanie macierzy w MATLABie zaczyna się od 1, tak więc element w lewym górnym rogu powyższej macierzy ma indeksy 1,1.

W MATLABie wszystkie zmienne są macierzami, nawet skalary są traktowane jako macierze o liczbie wierszy i kolumn równej 1. Dlatego ucząc się posługiwania MATLABem, należy dobrze opanować zawartość tego rozdziału.

W typowej macierzy mogą znajdować się jedynie liczby. Struktury, w których występują inne typy danych lub dane o różnych typach, zostaną omówione dalej. Zasady rachunku macierzowego i algebry liniowej oraz jej zastosowania w inżynierii procesowej dobrze opisuje monografia *Zaleskiego, 1987*.

3.1. Macierze pełne

3.1.1. Tworzenie macierzy pełnych

Macierze pełne (ang. *full matrices*) zwane inaczej macierzami gęstymi zawierają przeważającą liczbę elementów niezerowych. Domyślnie MATLAB traktuje każdą macierz jako gęstą.

W programie MATLAB macierze można tworzyć różnymi sposobami, podstawowym sposobem jest wyliczanie elementów macierzy. Do budowy macierzy używamy nawiasów kwadratowych []. Macierze budujemy np. przez

- zdefiniowanie wektora wierszowego, wyliczymy elementy rozdzielając je spacjami

```
A=[ 9 48 4 57 023 9 45 21 8 534 76 4 12 3]
```

- zdefiniowanie wektora kolumnowego, wyliczymy elementy rozdzielając je średnikami

```
E=[ 9; 48; 4; 57; 023; 9; 45; 21; 8; 534; 76]
```

- zdefiniowanie wektora wierszowego przez podanie zakresu zmiennej całkowitej

```
A=[1:5]
```

Będzie on wypełniony liczbami 1 2 3 4 5.

Aby zamienić go na wektor kolumnowy, należy wykonać operację transpozycji $A=\text{transpose}(A)$ inaczej $A=A'$.

- zdefiniowanie macierzy o rozmiarach 3×4 przez wyliczanie

```
F=[ 9 48 4 57; 023 9 45 21 ; 534 76 4 12]
```

- zdefiniowanie macierzy o rozmiarach 4×4 przez wyliczanie, kolumny rozdzielamy spacjami, natomiast przejście do następnego wiersza następuje przez naciśnięcie klawisza ENTER

```
G=[ 9      48      4      57
    23      9      45      21
      8     534     76      4
      3      6      8      4 ]
```

- tworzenie macierzy z innych macierzy i wektorów np.

```
A=[ 0:5; 5:10; 40:45]
```

Dwukropek to operator zakresu znany z instrukcji `for`.

Należy pamiętać, że muszą zgadzać się wymiary wektorów lub macierzy, z których będzie tworzona macierz główna. Poniższe zdefiniowanie jest błędne, w pierwszym wierszu jest sześć kolumn, w drugim wierszu jest osiem kolumn, a w trzecim wierszu są cztery kolumny.

```
[0:5; 0:7; 0:3]
```

- przez zbudowanie z innych macierzy

```
A=[ 1 4 7; 4 7 9; 4 2 8]
```

```
B=[ 3 7 4; 9 5 7; 6 9 4]
```

```
C=[A B]
```

```
D=[A; B]
```

Użycie nawiasów kwadratowych `[ ]` jest dopuszczalne jedynie przy definiowaniu tablicy. Symbol `[]` oznacza macierz pustą. Jeśli chcemy uzyskać dostęp do pojedynczego elementu macierzy podajemy nazwę macierzy i indeksy elementu w nawiasach okrągłych `( )`. I tak $A(1,2) = 4$ oznacza podstawienie wartości 4 pod element macierzy A w drugiej kolumnie pierwszego wiersza natomiast $x=A(1,2)$ podstawienie pod zmienną x tego elementu (zob. rozdz. 3.6).

Dodanie wiersza lub kolumny do istniejącej macierzy jest możliwe, jeśli mają one tyle samo elementów co macierz wyjściowa. Dodanie kolumny do macierzy A w postaci wektora pionowego v ma postać $A=[A \ v]$, dodanie wiersza w postaci wektora poziomego h ma postać $A=[A; \ h]$.

Usunięcie wiersza lub kolumny z istniejącej macierzy dokonuje się przez podstawienie pustego elementu []. Tak więc polecenie $A(2, :) = []$; usuwa cały drugi wiersz z macierzy A, a polecenie $A(:, 3) = []$; usuwa całą trzecią kolumnę. Operator : oznacza w tym przypadku wszystkie elementy wiersza bądź kolumny macierzy A.

3.1.2. Generowanie macierzy pełnych

Do generacji niektórych specjalnych macierzy służą funkcje:

<code>zeros(n)</code>	generuje macierz zerową o wymiarach $n \times n$
<code>zeros(n,m)</code>	generuje macierz zerową o wymiarach $n \times m$
<code>ones(n)</code>	generuje macierz wypełnioną jedynkami o wymiarach $n \times n$
<code>ones(n,m)</code>	generuje macierz wypełnioną jedynkami o wymiarach $n \times m$
<code>eye(n)</code>	generuje macierz o wymiarach $n \times n$ wypełnioną zerami, w której na głównej przekątnej są jedynki
<code>eye(n,m)</code>	generuje macierz wypełnioną zerami w której na głównej przekątnej są jedynki o wymiarach $n \times m$
<code>magic(n)</code>	generuje macierz o wymiarach $n \times n$ o własnościach kwadratu magicznego
<code>pascal(n)</code>	generuje macierz o wymiarach $n \times n$ której elementy są liczbami dodatnimi całkowitymi będącymi współczynnikami w trójkącie Pascala
<code>poissrnd(k,n,m)</code>	generuje macierz o wymiarach $n \times m$ liczb losowych z rozkładu Poissona z parametrem k
<code>rand(n)</code>	generuje macierz o wymiarach $n \times n$ wypełnioną liczbami pseudolosowymi o rozkładzie jednostajnym
<code>rand(n,m)</code>	generuje macierz o wymiarach $n \times m$ wypełnioną liczbami pseudolosowymi o rozkładzie jednostajnym
<code>randn(n)</code>	generuje macierz o wymiarach $n \times n$ wypełnioną liczbami pseudolosowymi o rozkładzie normalnym
<code>randn(n,m)</code>	generuje macierz o wymiarach $n \times m$ wypełnioną liczbami pseudolosowymi o rozkładzie normalnym
<code>linspace(x1,x2,n)</code>	generuje wektor od x1 do x2 podzielony na n równych części

<code>linspace(x1,x2)</code>	generuje wektor od x_1 do x_2 podzielony na 100 (domyślnie) równych części
<code>logspace(x1,x2,n)</code>	generuje wektor liczb logarytmicznych od x_1 do x_2 podzielony na n równych części
<code>logspace(x1,x2)</code>	generuje wektor liczb logarytmicznych od x_1 do x_2 podzielony na 50 (domyślnie) równych części
<code>compan(u)</code>	generuje macierz stowarzyszoną wielomianu
<code>hilb(n)</code>	generuje macierz Hilberta o wymiarach $n \times n$ i wyrazach obliczonych jako $1/(i+j-1)$ gdzie i – numer wiersza, j – numer kolumny. Macierz Hilberta jest przykładem macierzy źle uwarunkowanej.

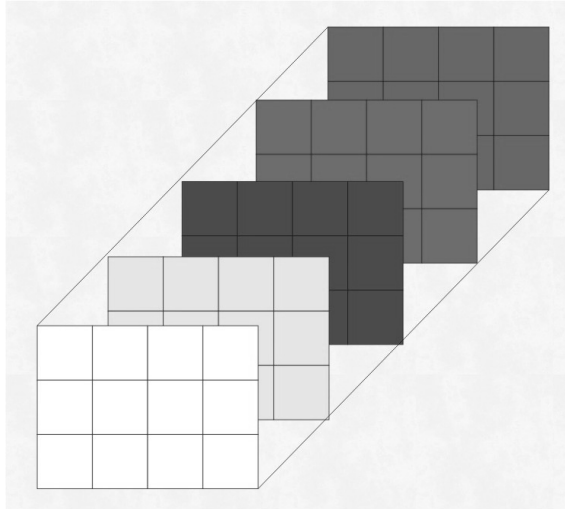
3.1.3. Funkcje przekształcające macierze pełne

Oto ważniejsze funkcje za pomocą których można dokonywać operacji na macierzach w MATLABie:

<code>diag(A,k)</code>	zwraca k -tą przekątną (główna przekątna ma $k = 0$)
<code>eig(A)</code>	zwraca wartości własne macierzy (lub wartości i wektory własne)
<code>flip1r(A)</code>	odwraca kolejno kolumny macierzy A (obróć pionowy)
<code>flipud(A)</code>	odwraca kolejno wiersze macierzy A (obróć poziomy)
<code>flipdim(A,dim)</code>	odwraca kolejno kolumny jeśli $dim=1$ lub kolejno wiersze jeśli $dim=2$
<code>repmat(A,m,n)</code>	tworzy macierz złożoną z m pionowych i n poziomych powtórzeń macierzy A
<code>reshape(A,n,m)</code>	zwraca macierz o n wierszach i m kolumnach z kolejnych elementów kolumn z macierzy A
<code>rot90(A)</code>	macierz A jest obracana w lewo o 90 stopni
<code>sort(A)</code>	uporządkuj wektor w porządku rosnącym
<code>sort(A,dim)</code>	uporządkuj macierz prostokątną w porządku rosnącym we wszystkich kolumnach ($dim=1$) lub wszystkich wierszach ($dim=2$)
<code>sortrows(A)</code>	posortuj wiersze macierzy prostokątnej w porządku rosnącym względem elementów w pierwszej kolumnie.
<code>transpose(A)</code>	zamienia kolumny z wierszami (inaczej A')
<code>tril(A,k)</code>	zwraca macierz trójkątną dolną z macierzy A względem k -tej przekątnej
<code>triu(A,k)</code>	zwraca macierz trójkątną górną z macierzy A względem k -tej przekątnej.

3.1.4. Macierze wielowymiarowe

Rozszerzeniem macierzy dwuwymiarowych są macierze wielowymiarowe. Macierze dwuwymiarowe mają dwa indeksy – pierwszy oznaczający wiersz a drugi kolumny, najczęściej oznaczany je $n \times m$ lub $i \times j$. W macierzy wielowymiarowej posługujemy się wieloma indeksami np. $n \times m \times p \times r$.



Rys. 3.1. Koncepcja tablicy trójwymiarowej

Definiowanie macierzy wielowymiarowej może być dokonane przez:

- wyliczanie elementów, np.

Przykład 3.1

 **p3_1.m**

% Przykład p3_1 Macierze wielowymiarowe

% Definiujemy macierz wielowymiarową

```
M(:,:,1)=[ 0.9501 0.4565 0.9218 0.4103 0.1389 0.0153  
            0.2311 0.0185 0.7382 0.8936 0.2028 0.7468  
            0.6068 0.8214 0.1763 0.0579 0.1987 0.4451  
            0.4860 0.4447 0.4057 0.3529 0.6038 0.9318  
            0.8913 0.6154 0.9355 0.8132 0.2722 0.4660  
            0.7621 0.7919 0.9169 0.0099 0.1988 0.4186 ];
```

```

M(:,:,2)=[ 0.4326 1.1892 0.5883 0.0956 0.6918 0.3999
          -1.6656 0.0376 2.1832 0.8323 0.8580 0.6900
           0.1253 0.3273 0.1364 0.2944 1.2540 0.8156
           0.2877 0.1746 0.1139 1.3362 1.5937 0.7119
          -1.1465 0.1867 1.0668 0.7143 1.4410 1.2902
           1.1909 0.7258 0.0593 1.6236 0.5711 0.6686 ];

M(:,:,3)=[ 0.8462 0.6813 0.3046 0.1509 0.4966 0.3420
           0.5252 0.3795 0.1897 0.6979 0.8998 0.2897
           0.2026 0.8318 0.1934 0.3784 0.8216 0.3412
           0.6721 0.5028 0.6822 0.8600 0.6449 0.5341
           0.8381 0.7095 0.3028 0.8537 0.8180 0.7271
           0.0196 0.4289 0.5417 0.5936 0.6602 0.3093 ];

% sprawdzamy rozmiar macierzy
size(M)

```

Rezultat 3.1

```

ans =
     6     6     3

```

- zastosowanie funkcji tworzących macierze specjalne

Stwórzmy tablicę wielowymiarową (trójwymiarową) za pomocą funkcji `randn`, tzn. generatora liczb pseudolosowych o rozkładzie normalnym.

Przykład 3.2

```

p3_2.m
%Przykład 3.2 Macierz trójwymiarowa
A=randn(5,5,5)
size(A)

```

Rezultat 3.2

```

A(:,:,1) = [ 1.7747 -1.5689 0.3025 -1.1206 -0.9177
             -0.4705 0.7002 -0.6566 1.4606 1.7010
             -0.9253 0.9738 -0.0522 0.4091 -0.0248
              0.0015 -0.0705 0.7732 1.1326 -0.7698
              0.6373 -0.0337 -0.3199 0.5700 -0.8375 ]
A(:,:,2) = [ -0.0918 1.1181 0.2771 1.2885 1.8336
              0.6301 -0.7003 -0.6745 -0.9561 -0.9653
              0.3845 -0.3501 0.2039 1.1957 -0.3970

```

```

-0.6099  1.8168  1.3074 -0.4056  0.0898
-0.4137 -1.6401  0.4621 -0.6034  0.2640 ]
A(:, :, 3) = [ 0.2922 -0.2943  0.5724  0.9175  0.7435
               0.7818 -1.7179  0.8150 -0.5469 -0.1620
               0.4040 -0.0539 -1.0789 -0.6051  0.4357
               -0.3254  0.2179 -0.5799  0.0253  0.8613
               -0.3739  2.1732 -1.8757  0.2615  0.0641 ]
A(:, :, 4) = [-1.7273  0.9900 -1.9037 -0.2199 -0.1121
               0.7160 -1.7817 -0.3921  0.5750  0.5628
               0.0366 -0.0440 -1.1070  0.1701 -0.0307
               -0.1849  0.8902  1.7575 -0.4958 -1.3228
               -0.8147 -0.4561 -0.7192  1.2027 -1.0830 ]
A(:, :, 5) = [ 0.1575  0.0742  0.0296 -1.0718 -0.3008
               -1.4664  1.0805  0.6917 -0.2211 -1.3683
               0.6736 -0.6624 -0.6856 -1.6758  0.7377
               0.3625  0.4754 -0.0431 -1.6981 -0.4043
               0.6065  1.2443 -1.5477 -0.1085  0.8568 ]
ans = 5 5 5

```

Stwórzmy tablicę pięciowymiarową za pomocą funkcji `rand`, tzn. generatora liczb pseudolosowych o rozkładzie jednostajnym.

```

A=rand(3,3,5,4,2)
size(A)

```

Wynik działania jest podobny jak w Przykładzie 3.2, więc rezultat nie będzie tu prezentowany. Użytkownik może sam wprowadzić te instrukcje.

3.2. Macierze rzadkie

3.2.1. Charakterystyka macierzy rzadkich

Macierzą rzadką (ang. *sparse matrix*) nazywamy macierz, w której większość elementów stanowią zera. Przykładem takich macierzy mogą być macierze pasmowe w tym diagonalne i trójdagonalne, a także macierze trójkątne.

Fizycznie macierze rzadkie opisują systemy o słabych związkach między elementami np. w przypadku elementów połączonych szeregowo, tak jak reaktory zbiornikowe w kaskadzie, macierz obrazująca taki związek jest macierzą trójdagonalną. Macierze połączeń występujące w analizie systemów są macierzami rzadkimi (zob. Projekt 3.2). Powstają one również podczas dyskretyzacji równań różniczkowych np. w mechanice płynów czy ruchu ciepła.

MATLAB zapisuje macierze rzadkie w pamięci w inny sposób niż macierze pełne. Podobnie algorytmy obliczeniowe dla macierzy rzadkich są inne niż dla macierzy pełnych, gdyż działań na elementach zerowych macierzy rzadkiej nie wykonuje się. Dlatego działania te realizowane na macierzach rzadkich trwają krócej niż działania algorytmów dla macierzach pełnych.

Macierze rzadkie są również wyświetlane w inny sposób niż macierze pełne. Pokazywane są jedynie numery wierszy i kolumn zawierających elementy niezerowe i wartość tych elementów. Przykładowo macierz gęsta postaci

```
A =
    1     0     0     0     0
    0     1     0     0     0
    0     0     1     0     0
```

po zamianie na macierz rzadką jest wyświetlana tak

```
A =
(1,1)      1
(2,2)      1
(3,3)      1
```

Do zamiany macierzy gęstej na rzadką służy funkcja `sparse(A)` a odwrotnie funkcja `full(A)`. O szybkości algorytmów macierzy rzadkich świadczy następujący eksperyment.

Przykład 3.3

p3_3.m

```
%Przykład 3.3 Macierze rzadkie a gęste
%porównanie czasów odwracania
A=sprand(100,100,0.2);    %generujemy dużą macierz rzadką
tic
B=inv(A);                %odwracamy macierz rzadką
toc
A=full(A);               %konwertujemy na macierz gęstą
tic
B=inv(A);                %odwracamy macierz gęstą
toc
```

Rezultat 3.3

```
Elapsed time is 0.018374 seconds.
Elapsed time is 0.112241 seconds.
```

Jak widać algorytm odwracania macierzy rzadkich jest sześć razy szybszy niż algorytm macierzy gęstych.

3.2.2. Generowanie macierzy rzadkich

Zostaną tu przedstawione następujące funkcje generujące macierze rzadkie:

<code>speye(m,n)</code>	macierz jednostkowa
<code>sprandn(m,n,density)</code>	macierz losowa o zadanym wypełnieniu, gęstość macierzy rozumie się jako stosunek elementów niezerowych do wszystkich elementów, $0 \leq \text{density} \leq 1$
<code>sprandsym(m,density)</code>	macierz losowa, symetryczna
<code>spdiags(A)</code>	macierz pasmowa
<code>sparse(i,j,s,m,n,nmax)</code>	generowanie macierzy rzadkiej, <code>[i,j,s]</code> trzy kolumny określające odpowiednio indeksy <code>i</code> , <code>j</code> oraz wartości elementów <code>s</code> macierzy rzadkiej, <code>[m,n]</code> wymiar macierzy rzadkiej, <code>nmax</code> – maksymalna liczba elementów niezerowych macierzy.

3.2.3. Operacje na macierzach rzadkich

Funkcje pozwalające przeprowadzać operacje na macierzach rzadkich:

<code>nnz(A)</code>	podaje liczbę elementów niezerowych
<code>nonzeros(A)</code>	przedstawia wszystkie elementy niezerowe w jednej kolumnie
<code>nzmax(A)</code>	maksymalna ilość elementów niezerowych (rezerwacja pamięci)
<code>spones(A)</code>	zamiana elementów niezerowych na jedynki
<code>issparse(A)</code>	sprawdza czy macierz <code>A</code> jest macierzą rzadką
<code>normest(A)</code>	oszacowanie normy
<code>condest(A)</code>	oszacowanie wskaźnika uwarunkowania
<code>sprank(A)</code>	strukturalny rząd macierzy rzadkiej
<code>spy(A)</code>	wykres przedstawiający niezerowe elementy macierzy rzadkiej
<code>etreeplot(A)</code>	wykres przedstawiający drzewo eliminacji.

Przykład 3.4

p3_4.m

%Przykład 3.4 Operacje na macierzach rzadkich

%Zdefiniujemy macierz rzadką A.

A=sparse(1:6,1:6,S);

%Sprawdźmy działanie kilku funkcji.

nnz(A)

nonzeros(A)

nzmax(A)

spones(A)

issparse(A)

normest(A)

condest(A)

sprank(A)

Rezultat 3.4

ans = 6

ans =

0.6338

0.0168

0.5155

0.4052

0.2488

0.3554

ans = 6

ans =

(1,1) 1

(2,2) 1

(3,3) 1

(4,4) 1

(5,5) 1

(6,6) 1

ans = 1

ans = 0.6338

ans = 37.7123

ans = 6

Przykład 3.5

 p3_5.m

%Przykład 3.5 Reprezentacja graficzna macierzy rzadkich

```
A=sprandn(100,100,0.012)
```

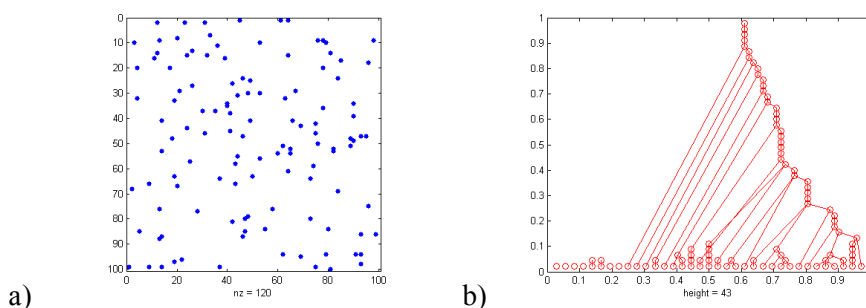
```
figure(1)3
```

```
spy(A)
```

```
figure(2)
```

```
etreeplot(A)
```

Rezultat 3.5



Rys. 3.2. a) Wykres przedstawiający niezerowe elementy macierzy rzadkiej A, uzyskany za pomocą polecenia `spy(A)`, b) wykres przedstawiający drzewo eliminacji, uzyskany za pomocą polecenia `etreeplot(A)`.

3.3. Łańcuchy i tablice znakowe

Łańcuch jest to ciąg znaków, który ograniczono apostrofami. Pamiętajmy, że typem zmiennej w MATLABie jest macierz, wobec tego jakikolwiek łańcuch znaków zapisywany jest jako macierz jednowierszowa, czyli wektor.

Przykład 3.6

 p3_6.m

%Przykład 3.6 łańcuch jako tablica znaków

```
Z='to jest książka o MATLABie';
```

```
% Sprawdźmy czy jest to na pewno wektor.
```

```
size(Z)
```

³ Ta funkcja, otwierająca okno graficzne zostanie omówiona w następnym rozdziale.

Rezultat 3.6

```
ans = 1 26
```

Jak widzimy jest to wektor jednowierszowy dwudziestosześcioelementowy. Sklejenia dwu łańcuchów (konkatenacja) można dokonać najprościej przez zapis

```
S=['Ala', ' ma ', 'Asa']
```

co jest równoznaczne z

```
S='Ala ma Asa'
```

Macierze znakowe tworzymy podobnie jak macierze pełne np. przez wyliczanie elementów, przez sklejenie wektorów łańcucha lub sklejenie macierzy z innych macierzy.

Przykład 3.7

 **p3_7.m**

%Przykład 3.7 Macierze znakowe

```
ZNAKI=['abc' 'dfg' 'wer'  
      'sdr' 'skh' 'jdg'  
      'sdf' 'ksd' 'kdk' ]  
  
ZNAKI_1= [ 'abc' ' ' 'dfg' ' ' 'wer'  
          'sdr' ' ' 'skh' ' ' 'jdg'  
          'sdf' ' ' 'ksd' ' ' 'kdk' ]  
  
size(ZNAKI)  
size(ZNAKI_1)
```

Rezultat 3.7

```
ZNAKI = abcdfgwer  
        sdrskhjdj  
        sdfksdkdk
```

Rozmiar macierzy ZNAKI, trzy wiersze i dziewięć kolumn.

```
ans = 3 9
```

```
ZNAKI_1 = abc dfg wer  
          sdr skh jdg  
          sdf ksd kdk
```

Rozmiar macierzy ZNAKI_1, trzy wiersze i jedenaście kolumn, zwróćmy uwagę iż spacja to też znak.

```
ans = 3 11
```

3.3.1. Funkcje wykonujące operacje na łańcuchach

<code>int2str(A)</code>	zamiana liczby całkowitej na łańcuch
<code>num2str(A,format)</code>	zamiana liczby rzeczywistej na łańcuch
<code>str2num(A,format)</code>	zamiana łańcucha na liczbę rzeczywistą
<code>sprintf(format, A, ...)</code>	zamiana liczby na łańcuch w wybranym formacie
<code>sscanf (s, format, size)</code>	zamiana łańcucha na liczbę o wybranym formacie
<code>mat2str(A)</code>	zamiana macierzy na łańcuch
<code>str2mat(A)</code>	zamiana łańcucha na macierz
<code>double(A)</code>	zamiana łańcucha na wektor jego wartości numerycznych kodu ascii
<code>char(A)</code>	działa odwrotnie do <code>double</code>
<code>lower(A)</code>	zamiana liter dużych na małe
<code>upper(A)</code>	zamiana liter małych na duże
<code>blanks(m)</code>	generuje łańcuch spacji
<code>deblank(A)</code>	kasuje wszystkie spacje w łańcuchu
<code>findstr(s,A)</code>	wyszukuje łańcuch znaków s w innym łańcuchu znaków A i podaje numer znaku, od którego zaczyna się łańcuch s
<code>strcat(s1,s2...)</code>	łączy łańcuchy poziomo
<code>char(s1,s2,...)</code>	łączy łańcuchy pionowo, uzupełniając krótsze spacjami do równej długości.

3.4. Tablice strukturalne

Mają one budowę zbliżoną do typowych struktur baz danych, a więc są to tablice o dowolnej liczbie pól. Każde pole może zawierać dane różnych typów, ale w obrębie pola muszą być one jednego typu. Pola mogą zawierać wielokrotne rekordy. Powstała struktura ma więc postać tablicy.

Pola można tworzyć przez indywidualne podstawienie, oddzielając nazwy pól od nazwy tablicy kropką np.

```
DANE_TERM.Mmol=25.2
```

Powstała struktura to baza danych o jednym polu, a w nim jednym rekordzie. Inną metodą tworzenia struktury jest jej wprowadzenie poleceniem `struct`. Za pomocą tego polecenia tworzy się strukturę wg pól bądź wg rekordów, jak to pokazuje przykład 3.8. Pokazano jednocześnie sposób dostępu do danych z tej struktury.

Przykład 3.8

p3_8.m

```
%Przykład 3.8 Macierze strukturalne
clc
%Pole DaneCr zawiera Tcr[K], pcr[bar], vcr[cm3/mol]

%struktura tworzona polami
DANE_TERM1=struct('Nazwa',char('metanol','etanol','aceton','2-
propanol'),...
                  'CAS',char('67-56-1','64-17-5','71-23-8','67-
63-0'),...
                  'DaneCr',[512.58 80.96 117.8; 400.1 53.7 170;
508.2 47.02 209; 508.31 46.64 220.1],...
                  'Mmol',[32.042; 46.069; 58.08; 60.096]);

%struktura tworzona rekordami
DANE_TERM2=[struct('Nazwa','metanol','CAS','67-56-
1','DaneCr',[512.58 80.96 117.8],'Mmol',32.042),...
            struct('Nazwa','etanol','CAS','64-17-
5','DaneCr',[400.1 53.7 170.0],'Mmol',46.069),...
            struct('Nazwa','aceton','CAS','71-23-
8','DaneCr',[508.2 47.02 209.0],'Mmol',58.08),...
            struct('Nazwa','2-propanol','CAS','67-63-
0','DaneCr',[508.31 46.64 220.1],'Mmol',60.096)];
```

Rezultat 3.8

Wytworzone struktury zawierają te same dane, ale nie są identyczne.

```
>>DANE_TERM1.Nazwa(1,:)

ans =
metanol
```

```
>>DANE_TERM2.Nazwa(1,:)
??? Field reference for multiple structure elements that is followed by more reference blocks is an error.
```

Po wpisaniu

```
>>DANE_TERM1.Nazwa
```

otrzymujemy

```
ans =
metanol
etanol
aceton
2-propanol
```

Natomiast dla bazy 2

```
>> DANE_TERM2.Nazwa
```

otrzymujemy

```
ans =
metanol
ans =
etanol
ans =
aceton
ans =
2-propanol
```

Z bazy 2 w prosty sposób można odczytać wszystkie pola jednego rekordu

```
>> DANE_TERM2(:,2)
```

```
ans =
      Nazwa: 'etanol'
      CAS: '64-17-5'
DaneCr: [400.1000 53.7000 170]
      Mmol: 46.0690
```

Odczytanie danego rekordu z bazy 1 wymaga odczytania kolejno wszystkich pól.

3.5. Macierze komórkowe

Macierze komórkowe są to macierze, które w swoich komórkach zawierają dane dowolnego typu. Komórkami mogą być inne macierze różnego typu np. w macierzy 3×3 w komórce (1,1) może być zapisana macierz 30×50 , a w komórce (2,3) może być przechowywana macierz znakowa. W komórkach można zapisywać także łańcuchy, struktury i inne komórki. Komórki tablicy wskazuje się poprzez podanie nazwy tablicy i indeksów komórki w nawiasie klamrowym {}. Tablice komórkowe tworzy się poprzez przypisanie odpowiednim komórkom ich zawartości np. poniżej utworzono macierz o wymiarach 2×3

```
A{1,1}=2;  
A{1,2}=rand(3);  
A{1,3}='łańcuch znaków';  
A{2,1}=magic(2);  
A{2,2}=sparse(1:6,1:6,rand(1,6));  
A{2,3}=num2str(pascal(3)) ;
```

Wyświetlenie zawartości tej macierzy pokazano poniżej.

```
A = [      2]      [3x3 double]      'łańcuch znaków'  
     [2x2 double]  [6x6 double]      [3x7 char]
```

Jak widać zawiera ona w swych komórkach różne typy danych, w tym inne macierze, lecz podstawowy jej wymiar pozostaje 2×3 .

3.5.1. Funkcje działające na macierzach komórkowych

Oto wybrane funkcje pozwalające przeprowadzać operacje na macierzach komórkowych.

<code>cell(m,n)</code>	generuje pustą macierz komórkową, m-liczba wierszy, n-liczba kolumn
<code>celldisp(A)</code>	drukuje w oknie Command Window zawartość macierzy komórkowej i wszystkich komórek
<code>cellplot(A)</code>	przedstawia graficznie macierz komórkową
<code>cell2struct(c,field,dim)</code>	zamienia macierz komórkową w strukturę
<code>struct2cell(A)</code>	zamienia strukturę w macierz komórkową
<code>iscell(A)</code>	sprawdza czy macierz a jest macierzą komórkową.

Przykład 3.9

p3_9.m

```
% Przykład 3.9 Macierze komórkowe
% Zdefiniujemy macierz komórkową A o rozmiarze 3x3.
A{1,1}=2;
A{1,2}=rand(3);
A{1,3}='łańcuch znaków';
A{2,1}=magic(2);
A{2,2}=sparse(1:6,1:6,rand(1,6));
A{2,3}=num2str(pascal(3)) ;
A{3,1}=[ 'abc' ' ' 'dfg' ' ' 'wer'
         'sdr' ' ' 'skh' ' ' 'jdj'
         'sdf' ' ' 'ksd' ' ' 'kdk' ];
A{3,2}=hilb(5);
A{3,3}=zeros(6);
% Wypiszmy wszystkie elementy macierzy komórkowej A.
celldisp(A)
%Przedstawmy graficznie budowę macierzy komórkowej A.
cellplot(A)
```

Rezultat 3.9

```
A{1,1} = 2

A{2,1} =      1      3
          4      2

A{3,1} =      abc dfg wer
            sdr skh jdj
            sdf ksd kdk

A{1,2} =      0.2642      0.4984      0.9580
            0.9995      0.2905      0.7666
            0.2120      0.6728      0.6661

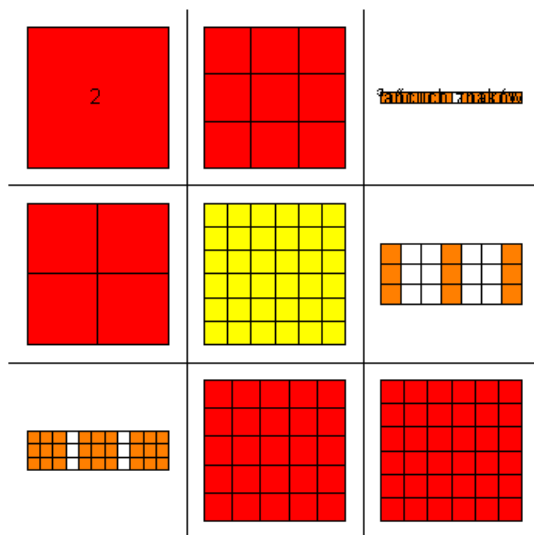
A{2,2} =      (1,1)      0.1309
            (2,2)      0.0954
            (3,3)      0.0149
            (4,4)      0.2882
            (5,5)      0.8167
            (6,6)      0.9855
```

$A\{3,2\} =$	1.0000	0.5000	0.3333	0.2500	0.2000
	0.5000	0.3333	0.2500	0.2000	0.1667
	0.3333	0.2500	0.2000	0.1667	0.1429
	0.2500	0.2000	0.1667	0.1429	0.1250
	0.2000	0.1667	0.1429	0.1250	0.1111

$A\{1,3\} =$ łańcuch znaków

$A\{2,3\} =$	1	1	1
	1	2	3
	1	3	6

$A\{3,3\} =$	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	0	0	0	0	0



Rys. 3.3. Graficzna prezentacja macierzy komórkowej A wygenerowana przez przykład 3.9 za pomocą polecenia `cellplot(A)`

3.6. Odwołanie do elementów macierzy

Stwórzmy przykładową macierz A o wymiarach 10×10 za pomocą dowolnej z ww. funkcji.

```
A=pascal(10)
```

Pokażemy teraz odwołania do jej elementów.

Odwołanie do całej macierzy:

```
A=[
1  1  1  1  1  1  1  1  1  1
1  2  3  4  5  6  7  8  9 10
1  3  6 10 15 21 28 36 45 55
1  4 10 20 35 56 84 120 165 220
1  5 15 35 70 126 210 330 495 715
1  6 21 56 126 252 462 792 1287 2002
1  7 28 84 210 462 924 1716 3003 5005
1  8 36 120 330 792 1716 3432 6435 11440
1  9 45 165 495 1287 3003 6435 12870 24310
1 10 55 220 715 2002 5005 11440 24310 48620 ]
```

Odwołanie do elementu w 8 wierszu i 5 kolumnie:

```
A(8,5)
```

```
ans = 330
```

Odwołanie do elementu w 9 wierszu i 3 kolumnie:

```
A(9,3)
```

```
ans = 45
```

Odwołanie się do podmacierzy rozpiętej między drugim a piątym wierszem i trzecią a dziewiątą kolumną

```
A(2:5,3:9)
```

```
ans = [ 3  4  5  6  7  8  9
        6 10 15 21 28 36 45
        10 20 35 56 84 120 165
        15 35 70 126 210 330 495 ]
```

Odwołanie do szóstego wiersza , trzeciej i dziewiątej kolumny

```
A(6,3:9)
```

```
ans = [ 21 56 126 252 462 792 1287 ]
```


Odwołanie od trzeciego do dziewiątego wiersza i piątej kolumny

```
A(3:9,5)
```

```
ans =  
    15  
    35  
    70  
   126  
   210  
   330  
   495
```

Odwołanie do wszystkich wierszy w 7 kolumnie

```
A(:,7)
```

```
ans =  
     1  
     7  
    28  
    84  
   210  
   462  
   924  
  1716  
  3003  
  5005
```

odwołanie do 7 wiersza i wszystkich kolumn

```
A(7,:)
```

```
ans = [1 7 28 84 210 462 924 1716 3003 5005 ]
```

3.7. Operacje macierzowe i tablicowe

Operacje macierzowe spełniają reguły algebry macierzy. Operacje tablicowe są możliwe na dwu macierzach o tych samych rozmiarach. Wykonywana jest wtedy operacja na odpowiadających sobie elementach macierzy. Przyspiesza to znacznie obliczenia w porównaniu do wykonywania tej samej operacji za pomocą instrukcji pętli `for`. Do wykonywania operacji tablicowych są używane zwykle operatory poprzedzone kropką.

Tablica 3.1. Operacje macierzowe i tablicowe

Działanie	Zapis	Objaśnienia
Zmiana znaku	$-A$	znak wszystkich elementów macierzy zmieniany jest na przeciwny.
Dodawanie	$A+B$	dodawane są skalary albo odpowiadające sobie elementy macierzy, macierze muszą mieć te same rozmiary.
Odejmowanie	$A-B$	jw. ale dotyczy odejmowania.
Mnożenie tablicowe	$A \cdot B$	jw. ale dotyczy mnożenia.
Dzielenie tablicowe	$A./B$	jw. ale dotyczy dzielenia.
Potęgowanie tablicowe	$A.^n$	jw. ale dotyczy potęgowania.
Mnożenie macierzowe	$A*B$	macierz B musi mieć tyle kolumn ile wierszy ma macierz A, wynik spełnia zależność $C_{w,k} = \sum_{i=1}^{nA} A_{w,i} \cdot B_{i,k}$ gdzie nA – liczba kolumn macierzy A.
Dzielenie macierzowe	A/B	odpowiada wyrażeniu $B*A^{-1}$.
Dzielenie lewostronne macierzy	$A \setminus B$	odpowiada rozwiązaniu równania macierzowego $A*x = B$, gdzie x to wynik tej operacji.
Potęgowanie macierzowe	A^n	n jest liczbą całkowitą, wykonywane jest kolejne n mnożeń macierzy A przez nią samą. A^{-1} jest równoważne odwrotności macierzy A.
Transpozycja	A'	zamiana wierszy na kolumny.
Mnożenie skalarne wektorów	$\text{dot}(A,B)$	Produkt skalarny wektorów (tylko trójelementowych)
Mnożenie wektorowe wektorów	$\text{cross}(A,B)$	Produkt wektorowy wektorów (tylko trójelementowych)

Przykłady operacji macierzowych pokażemy poniżej.

- Dodawanie

Utwórzmy macierze A i B o wymiarach 3×3

$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

```
    7 8 9 ];  
B=[ 1 1 1  
    1 1 1  
    1 1 1 ];
```

Dodajmy do macierzy A macierz B, a wynik zapiszmy w macierzy C, pamiętajmy, aby wymiary macierzy się zgadzały. Nie możemy dodawać do siebie macierzy o różnych rozmiarach.

```
C=A+B
```

Oto wynik, który uzyskamy pisząc C i wciskając Enter

```
C=[ 2 3 4  
    5 6 7  
    8 9 10 ]
```

- odejmowanie

Odejmijmy od macierzy A macierz B, a wynik zapiszmy w macierzy C

```
C= A-B
```

```
C =[ 0 1 2  
     3 4 5  
     6 7 8 ]
```

- mnożenie

Stworzymy macierz A o wymiarach 3×4

```
A=[ 1 2 3 5  
    4 5 6 7  
    7 8 9 2 ];
```

Stworzymy macierz B o wymiarach 4×3

```
B=[ 2 5 6  
    8 2 6  
    3 2 6  
    4 5 7 ];
```

Stworzymy macierz C o wymiarach 3×4

```
C=[ 2 5 6 7  
    8 2 6 4  
    3 2 6 1 ];
```

Stworzymy wektor D o wymiarach 4×1

```
D=[ 2; 6; 3; 1];
```

Wykonajmy mnożenie macierzowe

```
Mm=A*B
```

```
Mm1=A*D
```

Wykonajmy mnożenie tablicowe, czyli mnożone są odpowiadające sobie elementy. Za operacje tablicowe odpowiada znak kropki.

```
Mt=A.*C
```

Oto wyniki

```
Mm =[ 47  40  71
      94  77 139
      113 79 158 ]
```

```
Mm1=[ 28
      63
      91 ]
```

```
Mt =[ 2  10  18  35
      32  10  36  28
      21  16  54  2 ]
```

- dzielenie

Stwórzmy macierze A i B o wymiarach 4×4

```
A=[ 1 2 3 5
    4 5 6 7
    7 8 9 2 ];
```

```
B=[ 2 5 6 7
    8 2 6 4
    3 2 6 1 ];
```

Dzielenie macierzowe macierzy w prawą stronę A przez B

```
D1=A/B
```

```
D1 =    0.6436    0.1060   -0.3037
        0.8568    0.3357   -0.1675
        0.2344    0.1901    1.3248
```

Dzielenie macierzowe w prawą stronę macierzy B przez A

```
D2=B/A
```

```
D2 =    2.3492   -0.8095    0.4603
       -4.0317    3.6190   -0.5873
        2.2698   -1.7619    0.9921
```

Dzielenie macierzowe w lewą stronę macierzy A przez B

D3=A\B

D3 =	4.3571	-4.6429	-3.0000	-5.0000
	0	0	0	0
	-3.4048	3.9286	3.0000	4.0000
	1.5714	-0.4286	0.0000	0.0000

Dzielenie macierzowe w lewą stronę macierzy B przez A

D4=B\A

D4 =	0.0000	0.0000	0.0000	0.6364
	0	0	0	0
	1.3333	1.5000	1.6667	-0.0859
	-1.0000	-1.0000	-1.0000	0.6061

Przeprowadźmy teraz dzielenie tablicowe, tzn. z użyciem znaku kropki. Najpierw utworzymy macierze A i B o wymiarach 4×4

A=[1 2 3 5
4 5 6 7
7 8 9 2];

B=[2 5 6 7
8 2 6 4
3 2 6 1];

Dzielenie tablicowe w prawą stronę macierzy A przez B (wynik poniżej):

D1=A./B

D1 =	0.5000	0.4000	0.5000	0.7143
	0.5000	2.5000	1.0000	1.7500
	2.3333	4.0000	1.5000	2.0000

Dzielenie tablicowe w lewą stronę macierzy A przez B:

D2=B.\A

D2 =	0.5000	0.4000	0.5000	0.7143
	0.5000	2.5000	1.0000	1.7500
	2.3333	4.0000	1.5000	2.0000

Dzielenie tablicowe w prawą stronę macierzy B przez A:

D3=B./A

D3 =	2.0000	2.5000	2.0000	1.4000
	2.0000	0.4000	1.0000	0.5714
	0.4286	0.2500	0.6667	0.5000

Dzielenie tablicowe w lewą stronę macierzy B przez A:

D4=A.\B

D4 =	2.0000	2.5000	2.0000	1.4000
	2.0000	0.4000	1.0000	0.5714
	0.4286	0.2500	0.6667	0.5000

Zwróćmy uwagę, że przy dzieleniu tablicowym dzielenie macierzy w prawą stronę A przez B daje te same wyniki co dzielenie macierzy w lewą stronę A przez B.

- potęgowanie

Stworzymy macierz A o wymiarach 4×4.

A=[1 2 3 5
4 5 6 7
7 8 9 2];

Wykonajmy potęgowanie tablicowe, tj. podnosimy każdy element tablicy A do potęgi i wynik zapisujemy w macierzy C, przy operacji potęgowania tablicowego przy elemencie podnoszonym do potęgi musi wystąpić znak kropki.

C=A.^2

C =[1 4 9 25
16 25 36 49
49 64 81 4]

Wykonajmy teraz potęgowanie macierzowe. Ma ono miejsce, kiedy przy elemencie podnoszonym do potęgi nie wystąpi znak kropki. Aby podnieść macierz do potęgi musi być on kwadratowa, tzn. n×n, wynik zostanie zapisany w macierzy D.

D=A^2

D =[35 61 77 55
73 116 145 109
104 136 164 121
76 113 138 90]

3.8. Funkcje macierzowe

expm(A)	eksponent macierzowy
inv(A)	odwrotność macierzy = 1/A
logm(A)	logarytm naturalny przy podstawie e macierzowy

<code>sqrtm(A)</code>	pierwiastek kwadratowy z A macierzowy
<code>funm(A,@fun)</code>	inne funkcje macierzowe
<code>dot(A,B)</code>	oblicza iloczyn skalarny wektorów A i B
<code>cross(A,B)</code>	oblicza iloczyn wektorowy
<code>prod(A,dim)</code>	oblicza iloczyn elementów zapisanych w wierszach i kolumnach
<code>cumprod(A)</code>	generuje macierz, którego kolejne wiersze są iloczynem aktualnego wiersza przez poprzedni
<code>sum(A)</code>	suma elementów macierzy A z każdej kolumn

3.9. Właściwości i normy macierzy

<code>det(A)</code>	wyznacznik macierzy A
<code>norm(A)</code>	normy wektora i macierzy A
<code>cond(A)</code>	wskaźnik uwarunkowania macierzy
<code>rcond(A)</code>	estymator odwrotności uwarunkowania macierzy
<code>rank(A)</code>	rzęd macierzy A
<code>size(A)</code>	rozmiar macierzy A
<code>length(A)</code>	długość macierzy A
<code>exist('nazwa')</code>	ustala, czy zmienna o danej nazwie istnieje jeśli tak to zwraca 1, jeśli nie to 0
<code>isempty(A)</code>	sprawdza, czy macierz A jest macierzą pustą jeśli tak to zwraca 1, jeśli nie to 0
<code>issparse(A)</code>	sprawdza, czy macierz A jest macierzą rzadką jeśli tak to zwraca 1, jeśli nie to 0
<code>isstr(A)</code>	sprawdza, czy elementy macierzy A są łańcuchami tekstowymi jeśli tak to zwraca 1, jeśli nie to 0
<code>isglobal(A)</code>	sprawdza, czy zmienna A jest zmienną globalną jeśli tak to zwraca 1, jeśli nie to 0
<code>any(A)</code>	sprawdza, który element macierzy A jest niezerowy jeśli tak to zwraca 1, jeśli nie to 0
<code>all(A)</code>	sprawdza, który element macierzy A jest zerowy jeśli tak to zwraca 1, jeśli nie to 0
<code>find(A)</code>	sprawdza i zwraca numery indeksów niezerowych elementów macierzy A

<code>isnan(A)</code>	sprawdza, który element macierzy A jest NaN jeśli tak to zwraca 1, jeśli nie to 0
<code>isinf(A)</code>	sprawdza, który element macierzy A jest Inf jeśli tak to zwraca 1, jeśli nie to 0
<code>finite(A)</code>	sprawdza, który element macierzy A ma wartość skończoną jeśli tak to zwraca 1, jeśli nie to 0.

Jako przykłady operacji macierzowych w zastosowaniu do inżynierii procesowej posłużą Projekty 3.1 do 3.3 przedstawione poniżej.

PROJEKT 3.1	
Temat:	Profile stężeń w dyfuzji trójskładnikowej na podstawie rozwiązania macierzowego równania dyfuzji wieloskładnikowej GSM
Toolbox:	–

Z cieczy dwuskładnikowej aceton-metanol (składniki 1 i 2) znajdującej się na dnie rurki o długości L zachodzi odparowanie do czystego powietrza (składnik 3). Obliczyć natężenia strumieni masy obu składników 1 i 2. Obliczyć też rozkład ułamków molowych składników 1, 2 i 3 w rurce i porównać z eksperymentalnym.

Rozwiązanie ściśle uogólnionych równań Stefana-Maxwella (GSM) dyfuzji wieloskładnikowej (*Krishna, Taylor, 1993, Pakowski, 2008-2014*) ma postać macierzową określoną odmiennymi równaniami na każdym z krańców drogi dyfuzji (indeks 0 oznaczać będzie lewy kraniec a δ prawy kraniec). Dla przypadku jednowymiarowego we współrzędnych kartezjańskich, na lewym krańcu przedziału, dla układu n składników rozwiązanie to ma postać:

$$N = C\beta_0 k_0^\bullet \Xi_0(x_0 - x_\delta) \quad (3.1)$$

gdzie: N – wektor natężeń strumieni masy składników, $\text{kmol}/(\text{m}^2 \text{ s})$

C – stężenie ogólne, kmol/m^3

$k_0^\bullet$ – macierz współczynników wnikania masy przy zerowych strumieniach masy.

Oblicza się ją ze wzoru:

$$k_0^\bullet = \mathbf{D} / L \quad (3.2)$$

gdzie $\mathbf{D}$ jest macierzą dyfuzyjności wieloskładnikowych obliczaną na podstawie macierzy dyfuzyjności binarnych Stefana-Maxwella $\mathcal{D}$ postaci

$$\mathcal{D} = \begin{bmatrix} 1 & \mathcal{D}_{1,2} & \dots & \mathcal{D}_{1,n} \\ \mathcal{D}_{2,1} & 1 & \dots & \mathcal{D}_{2,n} \\ \dots & \dots & 1 & \dots \\ \mathcal{D}_{n,1} & \mathcal{D}_{n,2} & \dots & 1 \end{bmatrix} \quad (3.3)$$

a L oznacza długość drogi dyfuzji, m.

Macierz $\mathbf{D}$ oblicza się za pomocą przekształcenia

$$\mathbf{D} = B_0^{-1}, \quad (3.4)$$

gdzie B to macierz o współczynnikach

$$B_{ii} = \frac{x_i}{\mathcal{D}_{in}} + \sum_{k=1, k \neq i}^n \frac{x_k}{\mathcal{D}_{ik}} \quad (3.5)$$

$$B_{ij} = -x_i \left(\frac{1}{\mathcal{D}_{ij}} - \frac{1}{\mathcal{D}_{in}} \right)$$

β_0 – macierz „bootstrapowa” zawierająca tzw. warunek określoności. Jest ona dana wyrażeniem

$$\beta_{ij} = \delta_{ij} + x_i \frac{\lambda_j - \lambda_n}{\sum_{k=1}^n \lambda_k x_k}, \quad (3.6)$$

gdzie współczynniki λ spełniają zależność

$$\sum_{i=1}^n \lambda_i N_i = 0 \quad (3.7)$$

δ_{ij} to symbol Kroneckera (1 dla $i = j$, 0 dla $i \neq j$). W naszym przypadku powietrze jest składnikiem inertnym, a więc $N_i = 0$. To określa wektor λ jako [0; 0; 1].

Φ – macierz bezwymiarowych strumieni masy o elementach

$$\begin{aligned}\Phi_{ii} &= \frac{N_i}{Ck_{in}} + \sum_{k=1, k \neq i}^n \frac{N_k}{Ck_{ik}} \text{ dla } i \neq j = 1, 2, \dots, n-1 \\ \Phi_{ij} &= -\frac{N_i}{C} \left(\frac{1}{k_{ij}} - \frac{1}{k_{in}} \right) \text{ dla } i = 1, 2, \dots, n-1\end{aligned}\quad (3.8)$$

Ξ – macierz korekcyjna na skończone strumienie masy

$$\Xi_0 = \Phi (EXP(\Phi) - I)^{-1}, \quad (3.9)$$

gdzie I to macierz jednostkowa. EXP oznacza eksponent macierzowy (funkcja `expm(A)`).

Równanie (3.1) jest uwikłane, gdyż wektor strumieni masy N występuje po obu stronach znaku równości. W Projekcie 3.1 posłużono się obliczonymi wcześniej strumieniami masy (zob. Projekt 6.1) ponieważ rozwiązanie równania (3.1) jest możliwe za pomocą metod rozwiązywania równań opisanych w rozdziale 6. Obliczono również rozkłady ułamków molowych x wszystkich składników wzdłuż drogi dyfuzji na podstawie wzoru

$$x = EXP\left(\Phi \frac{l}{L} - I\right) (EXP(\Phi) - I)^{-1} (x_\delta - x_0) + x_0, \quad (3.10)$$

gdzie x_δ – wektor stężeń na prawym krańcu drogi dyfuzji, l – długość bieżąca, m. Wiadomości niezbędne do wykonania wykresu zawarte są w rozdziale 4. Więcej wiadomości na temat dyfuzji wieloskładnikowej zawiera cytowana literatura.

P3_1.m

%PROJEKT 3.1

%Profile stężeń w dyfuzji trójskładnikowej na podstawie

%rozwiązania macierzowego równania dyfuzji wieloskładnikowej GSM

function SM3

clear all; close all; clc;

global n C L D N xd x0 bet0 D0 k0 Dane

%składnik 1 - acetone

```

%składnik 2 - metanol
%składnik 3 - powietrze

%Dane
P0=99.4;      %ciśnienie ogólne, kPa
T=328.5;      %temperatura, K
L=0.238;      %długość drogi dyfuzji,m
R=8.314;      %uniwersalna stała gazowa, kJ/(kmol K)
n=3;          %liczba składników;
C=P0/(R*T);   %stężenie ogólne, kmol/m3

%Eksperymentalne profile stężeń
x01=0.319;
x02=0.528;
x03=1-x01-x02;
l=0;
Dane=[1 x01 x02 x03;
      0.047 0.265 0.463 0.221;
      0.072 0.250 0.456 0.279;
      0.094 0.235 0.426 0.338;
      0.120 0.221 0.382 0.412;
      0.146 0.184 0.316 0.515;
      0.173 0.135 0.250 0.603;
      0.190 0.125 0.162 0.721;
      0.238 0 0 1;];

%Macierz dyfuzyjności binarnych SM
D12=8.48e-6;   D21=D12;
D13=13.72e-6; D31=D13;
D23=19.91e-6; D32=D23;

D=[1 D12 D13;
   D21 1 D23;
   D31 D32 1;];

x0=[x01;x02];      %wektor ułamków molowych na lewym krańcu
xd=[0;0];          %wektor ułamków molowych na prawym krańcu
lamb=[0;0;1];      %współczynniki macierzy bootstrapowej
N=[1.783e-6;3.127e-6;0]; %założony wektor strumieni molowych
D0=(B(D,x0))^-1;
k0=D0/L;
bet0=beta(lamb,x0);
Fi=Fif(D,N);

```

```

I=eye(n-1);
Sig0=Fi*(expm(Fi)-I)^-1;
N0=C*bet0*k0*Sig0*(x0-xd); %obliczony wektor strumieni molowych

disp('strumień molowy składnika 1'); disp(N(1));
disp('strumień molowy składnika 2'); disp(N(2));
wykres;

function wykres
%wykreślenie eksperymentalnych profili stężeń
global Dane N D L
figure(1);
plot(Dane(:,1),Dane(:,2),'mo');
hold on
plot(Dane(:,1),Dane(:,3),'bs');
plot(Dane(:,1),Dane(:,4),'r^');
%wykreślenie obliczonych profili stężeń
NN=[N(1);N(2);0];
Fi0=Fif(D,NN);
z=0:0.005:L;
[m,r]=size(z);
for i=1:r
    y(i,:)=xmol(z(i),Fi0);
end
plot(z,y(:,1),'-m')
plot(z,y(:,2),'-b')
plot(z,1-y(:,1)-y(:,2),'-r')
title('Dyfuzja wieloskładnikowa na podstawie równania GSM');
legend('aceton','metanol','powietrze')
xlabel('Długość bieżąca [m]')
ylabel('ułamek molowy [-]')

function w=B(D,x) %oblicza odwrotność macierzy dyfuzyjności SM
[n,m]=size(D);
y=x;
y(n)=1-sum(x);
Fi=zeros(n-1); %przydzielenie pamięci na macierz Fi
for i=1:n-1
    for j=1:n-1
        if i==j
            for l=1:n

```

```

        W(l)=y(l)/D(i,l)-y(i)/D(i,i);
    end;
    Fi(i,i)=y(i)/D(i,n)+sum(W);
else
    Fi(i,j)=y(i)*(1/D(i,n)-1/D(i,j));
end
end
end
w=Fi;

function w=Fif(D,N) %oblicza macierz Fi, N to wektor[1xn]
global L C
k=D./L; %macierz k ma rozmiar [nxn]
[n,m]=size(N);
Fi=zeros(n-1); %macierz Fi ma rozmiar [(n-1)x(n-1)]
I=eye(n-1);
for i=1:n-1
    for j=1:n-1
        if i==j
            for l=1:n
                W(l)=N(l)/C/k(i,l);
            end;
            Fi(i,i)=N(i)/C/k(i,n)+sum(W);
        else
            Fi(i,j)=N(i)/C*(1/k(i,n)-1/k(i,j));
        end
    end
end
end
w=Fi;

function w=beta(lam,x) % oblicza macierz bootstrapową
[n,m]=size(lam);
xx=x;
xx(n)=1-sum(x);
s=sum(lam.*xx);
I=eye(n-1);
bet=ones(n-1);
for i=1:n-1
    for j=1:n-1
        bet(i,j)=I(i,j)-xx(i)*(lam(j)-lam(n))/s;
    end
end
end
w=bet;

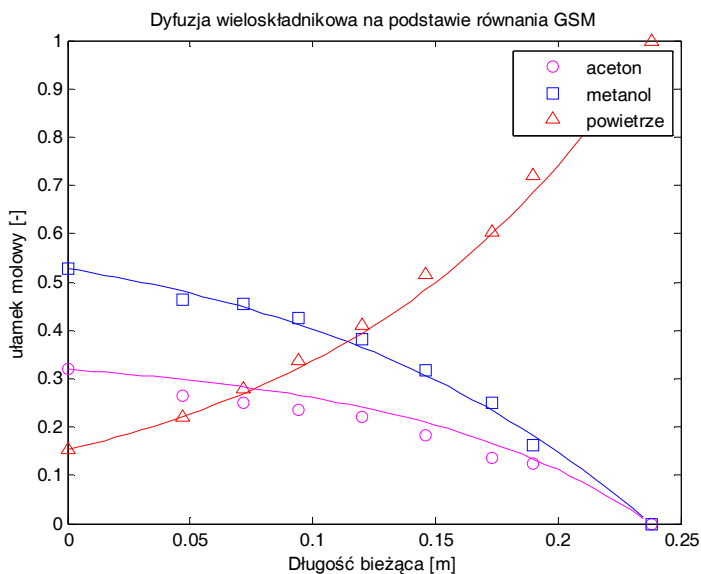
```

```
function y=xmol(x,Fi) %oblicza ułamki molowe
global L xd x0 n
I=eye(n-1);
f=(expm(Fi*x/L)-I)*(expm(Fi)-I)^-1;
y=f*(xd-x0)+x0;
```

Wyniki 3.1

strumień molowy składnika 1
1.7836e-006

strumień molowy składnika 2
3.1290e-006



Rys. 3.4. Obliczone i eksperymentalne rozkłady stężeń trzech składników dyfundujących w rurce

PROJEKT 3.2

Temat:	Bilanse masy dużych systemów liniowych – układ reaktorów
Toolbox:	-

Dany jest układ n reaktorów typu CSTR (ciągłych reaktorów zbiornikowych z idealnym wymieszaniem) połączonych w kaskadę. W układzie są możliwe pętle recyrkulacji strumieni (recykl) lub obejścia reaktorów (bypass). W reaktorach zachodzi reakcja rozkładu substancji znajdującej się w strumieniu wlotowym zachodząca wg kinetyki reakcji I-rzędu o stałej szybkości k . Oblicz stopień przereagowania dla danej konfiguracji recykli i bypassów i porównaj z tym dla kaskady bez recykli i bypassów. W kaskadzie reaktorów zastosowano recykl z reaktora 7 do reaktora 2 oraz z reaktora 9 do reaktora 5, a także by-pass z reaktora 2 do reaktora 8 (omija reaktory od 3 do 7). Pętle te przecinają się.

Stopniem recyklu nazywamy stosunek wielkości strumienia zawracanego do strumienia podstawowego.

$$r = \frac{V_r}{V} . \quad (3.11)$$

W identyczny sposób jest zdefiniowany stopień bypassu b . W przypadku układu n aparatów identyfikacji strumieni r & b dokonujemy, stosując indeksy oznaczające odpowiednie numery aparatów, pomiędzy którymi istnieje pętla. Zasady indeksowania pokazują rysunki 3.5 i 3.6.

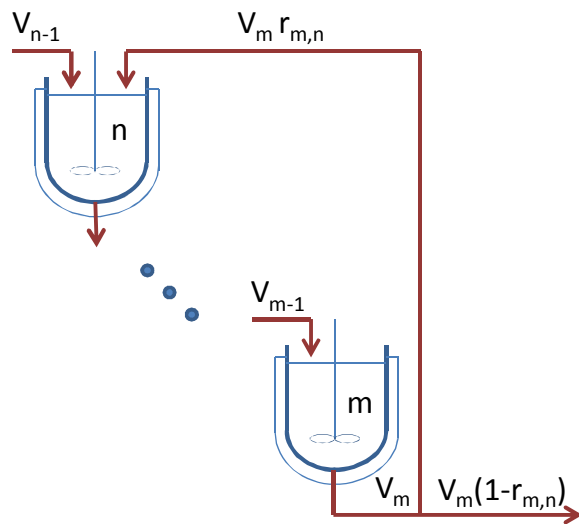
Bilans ogólny masy pojedynczego reaktora o numerze i , w kaskadzie w stanie ustalonym ma postać

$$\sum V_{in} - V_i = 0 . \quad (3.12)$$

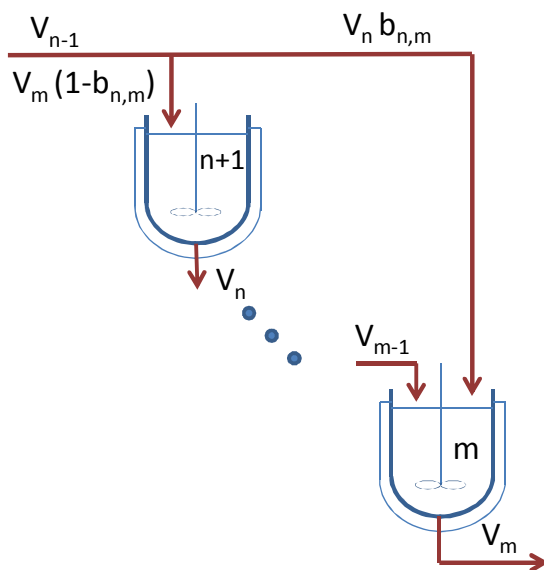
Natomiast bilans masy reagenta ma postać

$$\sum_j (V_j c_j)_{in} - V_i c_i - v_i k c_i = 0 , \quad (3.13)$$

gdzie V – natężenie objętościowe [m^3/s], v – objętość reaktora [m^3], k – stała szybkości reakcji [$\text{kmol}/(\text{m}^3 \text{ s})$], c – stężenie molowe reagenta [kmol/m^3]. Indeks in oznacza strumienie wlotowe. W kaskadzie bez r & b stężenie po każdym reaktorze można obliczać po kolei. W przypadku kaskady z pętlami r & b problem rozwiązuje się metodą kolejnych iteracji lub za pomocą rozwiązania układu równań opisującego bilanse wszystkich aparatów w systemie.



Rys. 3.5. Indeksowanie recykli



Rys. 3.6. Indeksowanie by-passów

Dla zdefiniowanego wyżej zadania układ równań ogólnego bilansu masy (3.12) wszystkich aparatów ma postać

$$\begin{aligned}
 1: & V_0 - V_1 = 0 \\
 2: & V_1 - V_2 + V_7 r_{7,2} = 0 \\
 3: & V_2(1 - b_{2,8}) - V_3 = 0 \\
 4: & V_3 - V_4 = 0 \\
 5: & V_4 - V_5 + V_9 r_{9,5} = 0 \\
 6: & V_5 - V_6 = 0 \\
 7: & V_6 - V_7 = 0 \\
 8: & V_2 b_{2,8} + V_7(1 - r_{7,2}) - V_8 = 0 \\
 9: & V_8 - V_9 = 0 \\
 10: & V_9(1 - r_{9,5}) - V_{10} = 0
 \end{aligned} \tag{3.14}$$

Ten układ równań można przedstawić w formie macierzowej jako

$$A \cdot V = b, \tag{3.15}$$

gdzie A ma postać macierzy rzadkiej o współczynnikach

	1	2	3	4	5	6	7	8	9	10
1	-1	0	0	0	0	0	0	0	0	0
2	1	-1	0	0	0	0	$r_{7,2}$	0	0	0
3	0	$1-b_{2,8}$	-1	0	0	0	0	0	0	0
4	0	0	1	-1	0	0	0	0	0	0
5	0	0	0	1	-1	0	0	0	$r_{9,5}$	0
6	0	0	0	0	1	-1	0	0	0	0
7	0	0	0	0	0	1	-1	0	0	0
8	0	$b_{2,8}$	0	0	0	0	$1-r_{7,2}$	-1	0	0
9	0	0	0	0	0	0	0	1	-1	0
10	0	0	0	0	0	0	0	0	$1-r_{9,5}$	-1

V jest wektorem natężeń przepływu, zaś b jest wektorem wyrazów wolnych, który zawiera w pierwszym wierszu $-V_0$ a zera w pozostałych. Warto zauważyć, że macierz ta odwzorowuje połączenia w układzie. Na dwu sąsiadujących przekątnych występują człony niezerowe, co oznacza istnienie połączenia między kolejnymi sąsiadującymi aparatami. Pozostałe niezerowe elementy macierzy pokazują istnienie połączeń między odległymi aparatami o numerach odpowiadających numerowi kolumny i wiersza elementu niezerowego. Dodatkowo

widać, że wszystkie recykle leżą w górnej macierzy trójkątnej (U), a wszystkie by-passy w dolnej macierzy trójkątnej (L).

Powyższe liniowe równanie macierzowe rozwiązuje się przez odwracanie macierzy (zob. rozdz. 6).

$$V = A^{-1} \cdot b. \quad (3.16)$$

W analogiczny sposób należy ułożyć i rozwiązać bilanse masy reagenta oparte na równaniu (7.13).

Stopień przereagowania jest zdefiniowany jako

$$s = \frac{c_0 - c_n}{c_0}. \quad (3.17)$$

P3_2.m

%PROJEKT 3.2

%Stopień przereagowania w kaskadzie CSTR z recyklami i bypassami(r&b)

clear all; close all; clc

%Dane

n=10;

c0=1; %stężenie wlotowe kmol/m3

V0=0.01; %natężenie objętościowe dopływu, m3/s

k=0.005; %stała szybkości reakcji, kmol/(m3 s)

v(1:n)=1; %objętości reaktorów, m3

v=v';

% KASKADA BEZ RECYKLI I BYPASSÓW

%1. Bilanse całkowite masy

A=-eye(n); %Wypełnienie macierzy współczynników układu

for i=2:n

 A(i,i-1)=1;

end;

b(1:n)=0; %Wypełnienie wektora poziomego wyrazów wolnych

b(1)=-V0;

b=b'; %Transpozycja do wektora pionowego

V=A^-1*b; % Rozwiązanie układu równań metodą odwracania macierzy

```

%2. Bilanse masy reagenta
Ar=zeros(n);           %Macierz główna
for i=1:n
    Ar(i,i)=-(V(i)+k*v(i));
end
for i=2:n
    Ar(i,i-1)=V(i-1);
end

br(1:n)=0;             %wektor wyrazów wolnych
br(1)=-c0*V0;
br=br';

c1=Ar^-1*br;           %rozwiązanie
s1=(c0-c1(n))/c0;
disp(['s1= ', num2str(s1)]);

%KASKADA Z RECYKLAMI I BYPASSAMI
% Pętle recyklu przebiegają
    % od reaktora 7 do reaktora 2 ze stopniem recyklu r7
    r72=0.4;
    % od reaktora 9 do reaktora 5 ze stopniem recyklu r9
    r95=0.5;
% Pętle bypassu przebiegają od reaktora 2 do reaktora 8
% (tj. obejmują reaktory od 3 do 7) ze stopniem bypassu b3
    b28=0.2;

%1. Bilanse całkowite masy

A=-eye(n);
for i=2:n
    A(i,i-1)=1;
end
%elementy zależne od r&b
A(3,2)=1-b28;
A(8,2)=b28;
A(5,9)=r95;
A(10,9)=1-r95;
A(2,7)=r72;
A(8,7)=1-r72;

V=A^-1*b;

```

```

% 2. Bilanse masy reagenta

Ar=zeros(n);
for i=1:n
    Ar(i,i)=-V(i)-k*v(i);
end
for i=2:n
    Ar(i,i-1)=V(i-1);
end

%elementy zależne od r&b
A(3,2)=V(2)*(1-b28);
A(8,2)=V(2)*b28;
A(5,9)=V(9)*r95;
A(10,9)=V(9)*(1-r95);
A(2,7)=V(7)*r72;
A(8,7)=V(7)*(1-r72);

br(1)=-V0*c0;

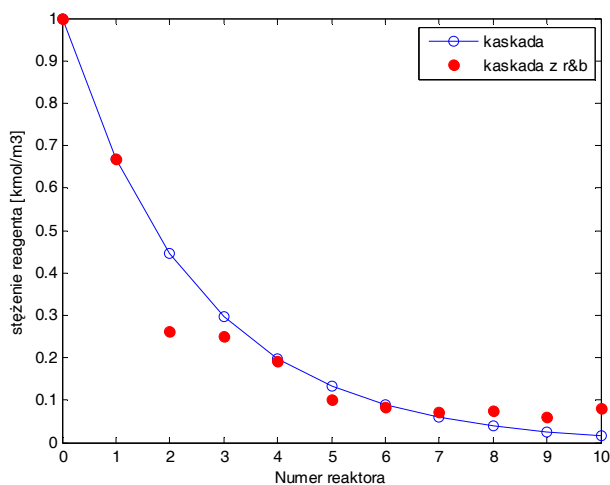
c2=Ar^-1*br;
s2=(c0-c2(n))/c0;
disp(['s2= ', num2str(s2)]);

%Dopisanie do wektorów stężeń c0 w pierwszym wierszu
c1n=1:n+1;
c2n=1:n+1;
for i=1:n
    c1n(i+1)=c1(i);
    c2n(i+1)=c2(i);
end
c1n(1)=c0;
c2n(1)=c0;

%----- Wykres stężeń ---zob. Rozdz.4 -----
i=0:n;
plot(i,c1n(i+1),'-b');
hold on
plot(i,c2n(i+1),'or','MarkerFaceColor','r');
xlabel('Numer reaktora');
ylabel('stezenie reagenta, kmol/m3');
legend('kaskada','kaskada z r&b');

```

Wyniki 3.2



Rys. 3.7. Porównanie stężeń w kolejnych reaktorach w przypadku kaskady i kaskady z pętlami r&b. Stężenie dla fikcyjnego reaktora 0 to stężenie wlotowe

Obliczone stopnie przereagowania są następujące

$s_1=0.9827$

$s_2=0.9199$

Jak widać dla tej reakcji stopień przereagowania jest wyższy w kaskadzie bez pętli r&b. W miarę jak liczba reaktorów w takiej kaskadzie rośnie stopień przereagowania zbliża się do uzyskiwanego w reaktorze okresowym po czasie odpowiadającym łącznemu czasowi przebywania w kaskadzie.

PROJEKT 3.3

Temat: Baza danych fizykochemicznych 22 wybranych związków (baza BC)

Toolbox: -

Do celów dydaktycznych stworzono bazę 22 związków chemicznych, których podstawowe dane niezależne i zależne od temperatury są zawarte w pliku Excela o nazwie **BC.xls**. Ten plik danych pochodzi z książki *Pakowski, Głębowski 2001*. W projekcie zastosowano elementy budowy funkcji z przekazaniem nazwy funkcji, zmiennej liczbie parametrów wyjściowych (zob. Rozdz. 2), a także

macierze komórkowe, łańcuchy i inne. Przykład ilustruje techniki posługiwania się macierzami jako tablicami danych. Metody wczytywania danych z pliku zastosowane w programie są opisane w rozdz. 5.

Baza ta jest wiarygodna i może być stosowana do obliczeń projektowych wykonywanych podczas studiów. Nie gwarantujemy jednak przydatności tej bazy do celów profesjonalnych. Oczywiście, bazę można udoskonalić np. przez sprawdzanie czy temperatura wywołania mieści się w zakresie stosowalności Prace rozwojowe pozostawiamy Czytelnikowi.

Poniżej są pokazane pliki MATLABa, nie pokazano zawartości pliku **BC.xls**.

P3_3.m

%PROJEKT 3.3

%Baza danych własności fizykochemicznych

%wybranych 22 związków chemicznych (baza BC).

%Źródło: Pakowski Z., Głębowski M., Symulacja procesów inżynierii

%chemicznej, Pł, 2001

```
clear all; clc
```

```
close all;
```

```
global num_BC comp_BC dane_BC t0 %zmienne globalne wymagane  
przez bazę
```

```
t0=273.15;
```

```
loadbase('BC'); %wczytanie bazy
```

```
[icomp info]=searchbase('H2O'); %odczytanie składnika z bazy
```

```
%przy użyciu wzoru stechiometrycznego.
```

```
%Można też używać nazwy polskiej 'Woda' i angielskiej 'Water'
```

```
%icomp - numer składnika w bazie
```

```
%info - 1 jeśli jest, 0 -gdy nie ma
```

```
if info==0
```

```
    disp('Brak składnika w bazie -koniec programu');
```

```
    break
```

```
end
```

```
%-----Właściwości niezależne od temperatury
```

```
%Kolejność w rekordzie jest następująca
```

```
% 2 - masa molowa, kg/kmol
```

```
% 3 - temperatura topnienia, K
```

```
% 4 - temperatura wrzenia pod ciśn. norm., K
```

```
% 5 - temperatura krytyczna, K
```

```
% 6 - ciśnienie krytyczne, bar
```

```
% 7 - molowa objętość krytyczna, cm3/mol
```

```
% 8 - gęstość krytyczna, g/cm3
```

```
% 9 - krytyczny współczynnik ściśliwości, -
```

```

%10 - czynnik acentryczny, -
%11 - objętość dyfuzyjna we wzorze Fullera, -

%Odczytanie masy molowej
Point=readbase('Point',icomp);
MM=Point(2)

%-----Właściwości zależne od temperatury
%Psat      - prężność pary nasyconej
%Liqvisc   - lepkość cieczy, Pa s
%Vapvisc   - lepkość pary, Pa s
%Liqhcap   - ciepło właściwe cieczy, kJ/(kg K)
%Vaphcap   - ciepło właściwe pary, kJ/(kg K)
%Hvap      - ciepło parowania, kJ/kg
%Liqdens   - gęstość cieczy, kg/m3
%Liqcond   - przewodność cieplna cieczy, kW/(m K)
%Vapcond   - przewodność cieplna pary, kW/(m K)

%Odczytanie gęstości składnika w danej temperaturze
[Ps jedn zakres]=readbase('Psat',icomp,20)

%Wykres przewodności cieplnej cieczy
[d unit range]=readbase('Liqcond',icomp,20);
t=linspace(range(1),range(2),50);
[d unit range]=readbase('Liqcond',icomp,t);
figure(1)
plot(t,d);
grid on; xlabel('temperatura, \circ C')
ylabel(['przewodność cieplna ' unit]);

```

Funkcja, która wczytuje bazę ma postać:

loadbase.m

```

function loadbase(bnazwa)
global num_BC comp_BC dane_BC
plik=strcat(bnazwa,'.xls');
%Wczytanie arkusza ze spisem związków
[num_BC comp_BC]=xlsread(plik,'Comps');
%Wczytanie arkuszy z danymi liczbowymi
dane_BC{1}=xlsread(plik,'Point');
dane_BC{2}=xlsread(plik,'Psat');
dane_BC{3}=xlsread(plik,'Liqvisc');
dane_BC{4}=xlsread(plik,'Vapvisc');

```

```
dane_BC{5}=xlsread(plik,'Liqhcap');
dane_BC{6}=xlsread(plik,'Vaphcap');
dane_BC{7}=xlsread(plik,'Hvap');
dane_BC{8}=xlsread(plik,'Liqdens');
dane_BC{9}=xlsread(plik,'Liqcond');
dane_BC{10}=xlsread(plik,'Vapcond');
```

Funkcja, która przeszukuje bazę:

searchbase.m

```
function [j,info]=searchbase(comp)
global num_BC comp_BC
n=length(num_BC);
icomp=0;
for i=1:n
    w1=strcmp(comp_BC(i,2),comp);
    w2=strcmp(comp_BC(i,3),comp);
    w3=strcmp(comp_BC(i,4),comp);
    if w1+w2+w3>0
        icomp=i;
    end
end
j=icomp;
if icomp==0
    info=0;
else info=1;
end
```

Główną częścią bazy jest funkcja obliczająca właściwości na podstawie odpowiednich równań aproksymacyjnych.

readbase.m

```
function [out unit range]=readbase(nazwa,i,t)
if nargin==2
    out=Point(i,0);
else
    [out unit range]=feval(nazwa,i,t);
end

function y = Point(i,~)
%Dane niezależne od temperatury (wektor)
global dane_BC
```



```

sheet=dane_BC{1};
y=sheet(i,:);

function [y u r] = Psat(i,t)
%Prężność pary nasyconej
global dane_BC t0
sheet=dane_BC{2};
vi=sheet(i,:);
T=t+t0;
w=vi(2)+vi(3)./T+vi(4)*log10(T)+vi(5)*T+vi(6)*T.^2;
y=10.^w*0.133322368;
u='kPa';
r=[vi(7) vi(8)]-t0;

function [y u r] = Liqvisc(i,t)
%Lepkość cieczy
global dane_BC t0
sheet=dane_BC{3};
vi=sheet(i,:);
T=t+t0;
w=vi(2)+vi(3)./T+vi(4)*T+vi(5)*T.^2;
y=10.^w*1e-3;
u='Pa s';
r=[vi(6) vi(7)]-t0;

function [y u r] = Vapvisc(i,t)
%Lepkość pary
global dane_BC t0
sheet=dane_BC{4};
vi=sheet(i,:);
T=t+t0;
y=(vi(2)+vi(3)*T+vi(4)*T.^2)*1e-6;
u='Pa s';
r=[vi(5) vi(6)]-t0;

function [y u r] = Liqhcap(i,t)
%Ciepło właściwe cieczy
global dane_BC t0
sheet=dane_BC{5};
vi=sheet(i,:);
li=Point(i,t); MM=li(2);
T=t+t0;
w=vi(2)+vi(3)*T+vi(4)*T.^2+vi(5)*T.^3;

```

```

y=w./MM;
u='kJ/(kg K)';
r=[vi(6) vi(7)]-t0;

function [y u r] = Vaphcap(i,t)
%Ciepło właściwe pary
global dane_BC t0
sheet=dane_BC{6};
vi=sheet(i,:);
li=Point(i,t); MM=li(2);
T=t+t0;
w=vi(2)+vi(3)*T+vi(4)*T.^2+vi(5)*T.^3+vi(6)*T.^4;
y=w./MM;
u='kJ/(kg K)';
r=[vi(7) vi(8)]-t0;

function [y u r] = Hvap(i,t)
%Ciepło parowania
global dane_BC t0
sheet=dane_BC{7};
vi=sheet(i,:);
li=Point(i,t); MM=li(2);
T=t+t0;
w=vi(2)*(1-T./vi(3)).^vi(4);
y=w*1e3./MM;
u='Pa s';
r=[vi(5) vi(6)]-t0;

function [y u r] = Liqdens(i,t)
%Gęstość cieczy
global dane_BC t0
sheet=dane_BC{8};
vi=sheet(i,:);
T=t+t0;
w=-(1-T./vi(5)).^vi(4);
y=vi(2)*vi(3).^w*1e3;
u='kg/m3';
r=[vi(5) vi(6)]-t0;

function [y u r] = Liqcond(i,t)
%Lepkość cieczy
global dane_BC comp_BC t0
sheet=dane_BC{9};

```

```

vi=sheet(i,:);
io=strcmp(comp_BC(i,5),'organic');
T=t+t0;
if io
    w=10.^(vi(2)+vi(3)*(1-T./vi(4)).^(2/7));
else
    w=10.^(vi(2)+vi(3)*T+vi(4)*T.^2);
end
y=w*1e-3;
u='kW/(m K)';
r=[vi(5) vi(6)]-t0;

function [y u r] = Vapcond(i,t)
%Lepkość cieczy
global dane_BC t0
sheet=dane_BC{10};
vi=sheet(i,:);
T=t+t0;
w=vi(2)+vi(3)*T+vi(4)*T.^2;
y=w*1e-3;
u='kW/(m K)';
r=[vi(5) vi(6)]-t0;

```

Wyniki 3.3

Program wydrukował:

- Odczytaną masę molową wody

```
MM =
    18.0150
```

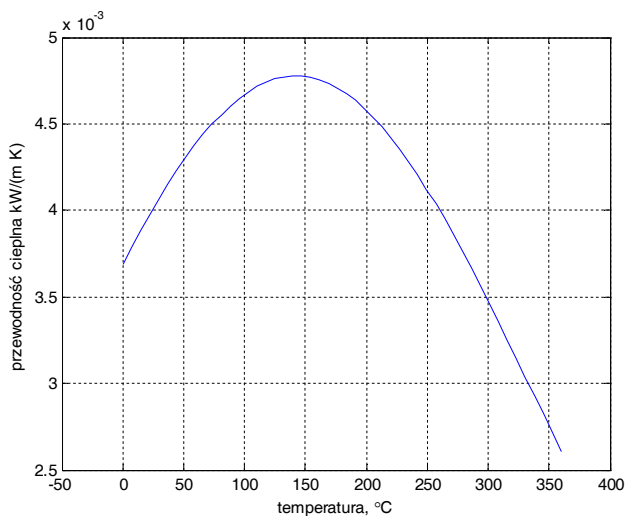
- Prężność pary wodnej w 20°C oraz jej jednostkę i zakres stosowalności równania (w °C):

```
Ps =
    2.3395
```

```
jedn =
kPa
```

```
zakres =
    0 373.9800
```

- Wykres zależności przewodności cieplnej ciekłej wody od temperatury



Rys. 3.8. Wykres zależności przewodności cieplnej ciekłej wody od temperatury w całym zakresie stosowalności równania bazy

LITERATURA

- Pakowski Z., „Procesy dyfuzyjne” (materiały do wykładów), PŁ, 2008-2014.*
- Krishna R., Taylor R., „Multicomponent Mass Transfer”, J. Wiley & Sons, 1993.*
- Pakowski Z., Głębowski M., „Symulacja procesów inżynierii chemicznej”, Wyd. PŁ, 2001, s. 170.*
- Zaleski T., „Metody algebry liniowej w inżynierii procesowej”, PWN, Warszawa 1987.*

4. WYKRESY I GRAFIKA

4.1. Wykresy 2D

4.1.1. Tworzenie wykresu 2D

Najprostszym poleceniem umożliwiającym wykreślenie wykresu funkcji jednej zmiennej jest polecenie `plot(x,y)`. Pozwala ono wykreślić wykres funkcji jednej zmiennej, co zilustrujemy przykładem.

Na początku, oprócz znanych poleceń `clear all` i `clc`, wydamy polecenia:

```
close all    zamyka wszystkie okna graficzne
clf         czyści zawartość okna Figure.
```

Wykres dowolnej funkcji w MATLABie jest rysowany w sposób dyskretny, jako zbiór punktów umieszczonych w dwu wektorach i połączonych linią. Musimy zdefiniować wektor argumentów określający dziedzinę funkcji w znany sposób:

```
x=0:0.01:20;
```

Następnie zdefiniujemy funkcję, którą chcemy wykreślić. W tym przypadku definiujemy funkcję bezpośrednio w **Command Window** lub w kodzie programu. Możemy również zdefiniować ją w osobnym m-pliku i wywołać ją w tym miejscu, podstawiając pod zmienną, w której mają być umieszczone wartości funkcji.

```
y=abs(sin(4*x.^0.4));
```

Gdy mamy już wektor argumentów `x` i wektor wartości funkcji `y` kolejne polecenie `figure(1)` otworzy okno graficzne, w którym będzie umieszczony wykres.

```
figure(1)
```

Możemy teraz wywołać polecenie `plot`, które naniesie wykres na płaszczyznę `xy`.

```
plot(x,y)
```

Każdy wykres powinien posiadać tytuł oraz opisane osie. Polecenie dodające tytuł wykresu ma postać:

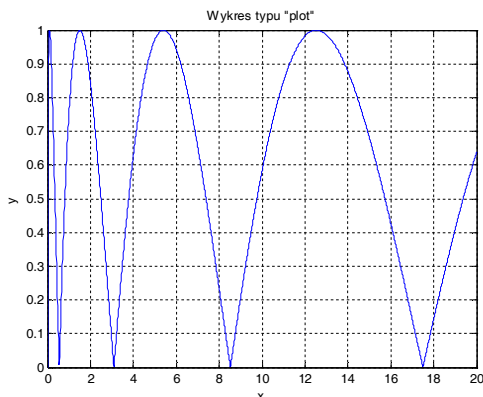
```
title('Wykres typu "plot" ')
```

Polecenia dodające opis osi to:

```
xlabel('x'); ylabel('y');
```

Na koniec dodamy na wykresie siatkę, aby łatwiej można było odczytywać wyniki.

```
grid on
```



Rys. 4.1. Wykres uzyskany za pomocą funkcji `plot` z domyślnym stylem

Otrzymany wykres pokazuje rysunek powyżej. Jest on jednym z najprostszych wykresów, jaki możemy uzyskać w MATLABie. Wykres uzyskany za pomocą polecenia `plot(x,y)` jest szybki, ale niezbyt efektowny. MATLAB oferuje nam szereg opcji umożliwiających bardziej elegancką prezentację. Pokażemy to na następującym przykładzie.

Z doświadczalnych pomiarów wartości przewodności cieplnej pewnej cieczy y w funkcji temperatury x , otrzymano punkty, które chcemy przedstawić na wykresie $y = f(x)$. W pierwszej kolejności trzeba wprowadzić do programu dane doświadczalne. Najprościej będzie to wykonać poprzez zdefiniowanie odpowiadających sobie wektorów x i y .

```
x=[0 20 40 60 80 100 120 140 150 160 180 200...  
  220 240 250 260 280 300]';  
y=[0.555 0.598 0.627 0.651 0.669 0.682 0.685...  
  0.684 0.683 0.680 0.673 0.665 0.652 0.634...  
  0.624 0.613 0.588 0.564]' .* 10e-3;
```

Uwaga: wektory muszą być uporządkowane według rosnącego x . Gdy tak nie jest, punkty będą łączone linią w kolejności występowania (zob. Rys. 5.2a). Można to wykonać w następujący sposób:

```
A=[x y];           %Sklejenie wektorów w macierz
A=sortrows(A);     %Sortowanie rosnąco wg pierwszej kolumny czyli x
x=A(:,1);          %Rozdzielenie macierzy na wektory
y=A(:,2);
```

Gdy mamy już uporządkowany wektor argumentów `x` i wektor wartości funkcji `y`, możemy przystąpić do ich prezentacji graficznej. Otwieramy nowe okno graficzne poleceniem `figure(n)` o kolejnym numerze `n`. Polecenie to może być pominięte, jeśli wystarcza nam tylko jedno okno graficzne, ponieważ funkcje graficzne wyświetlają wyniki działań w aktywnym oknie, a jeśli żadne okno graficzne **Figure** nie było wcześniej otwarte, to automatycznie tworzą nowe.

```
figure(2);
```

Wywołujemy polecenie `plot(x,y)`. Zwróćmy uwagę, że poniższe wywołanie różni się od wcześniejszego. W poleceniu zostały wywołane opcje wykresu, które wzbogacą wizualnie nasz wykres.

```
plot(x,y,'-mo','MarkerEdgeColor','k',...
      'MarkerFaceColor',[0.49 1 0.63],...
      'MarkerSize',6);
```

W poleceniu `plot` zostały użyte opcje odpowiadające za styl linii, styl punktów oraz rozmiar punktów. Argumenty polecenia `plot` użyte w tym wywołaniu są następujące:

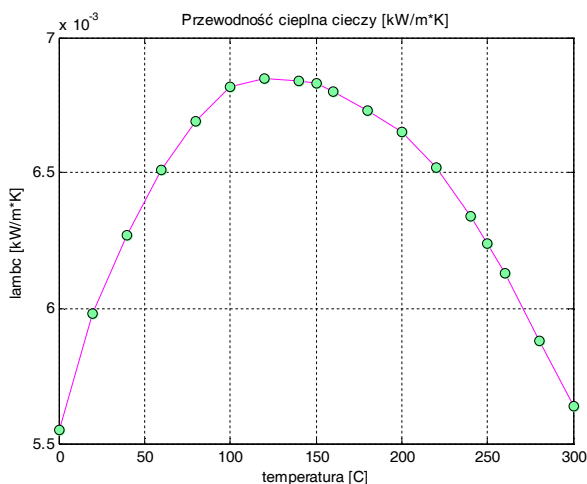
<code>x</code>	wektor zmiennej niezależnej,
<code>y</code>	wektor zmiennej zależnej,
<code>'-mo'</code>	definicja stylu linii i punktów: <code>-m</code> oznacza linię ciągłą koloru Magenta, a <code>o</code> oznacza punkty o kształcie kółek,
<code>'MarkerEdgeColor'</code>	wywołanie opcji zmiany koloru obrzeża punktów,
<code>'k'</code>	oznacza kolor czarny (lista kolorów w sekcji 4.1.4),
<code>'MarkerFaceColor'</code>	wywołanie opcji zmiany koloru wnętrza punktów,
<code>[0.49 1 0.63]</code>	oznacza udział kolorów RGB w kolorze punktu (w zakresie 0 - 1)
<code>'MarkerSize'</code>	wywołanie opcji zmiany rozmiaru punktów,
<code>6</code>	wybrany rozmiar.

Dodajmy siatkę na wykres.

```
grid on
```


oraz tytuł i opis osi

```
title('Przewodność cieplna cieczy [kW/m*K]')  
xlabel('temperatura [C]'); ylabel('lambc [kW/m*K]')
```



Rys. 4.2. Wykres uzyskany za pomocą funkcji `plot` z indywidualnie zdefiniowanym stylem

MATLAB oferuje również polecenia eksportu wygenerowanego wykresu do pliku graficznego, który następnie możemy wykorzystać w procesorze tekstu, czy zamieścić na stronie w Internecie. Umożliwia to polecenie `print`. Plik zostanie umieszczony w aktualnym katalogu.

```
print -djpeg w2.jpg
```

Argumenty polecenia `print` definiują format graficzny w jakim ma być zapisany plik, którego nazwa z rozszerzeniem podana jest po spacji. Często używane opcje i odpowiadające im formaty graficzne to:

djpg	jpg	– format jpg
dpdf	pdf	– format pdf
dbmp	bmp	– mapa bitowa
dmeta	emf	– Windows enhanced metafile

W oknie Figure można też wybrać opcję File/Save. Wtedy wykres zostanie zapisany z rozszerzeniem `fig`. Ten format pozwala na późniejszą edycję wykresu w edytorze graficznym MATLABa.

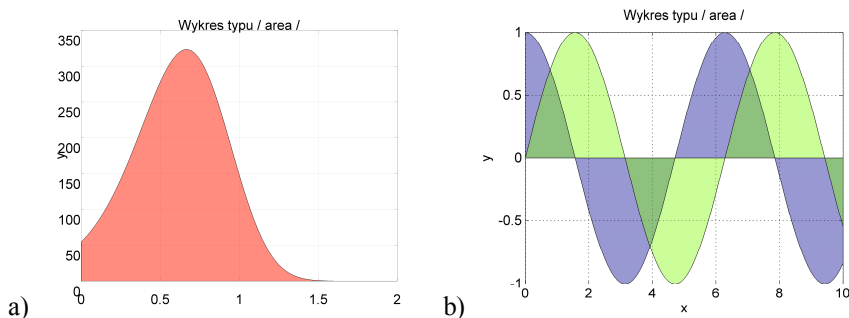
4.1.2. Typy wykresów 2D

Dostępne w wersji podstawowej MATLABa typy wykresów 2D zawiera Tablica 4.1.

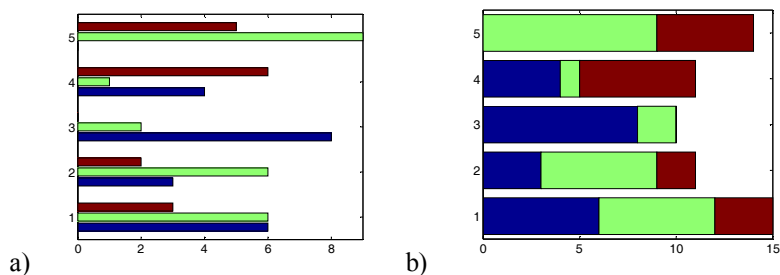
Tablica 4.1. Typy wykresów 2D w MATLABie

<code>area(x,y)</code>	wykres funkcji z wypełnionym obszarem pod nią
<code>bar(x,y)</code>	wykres słupkowy
<code>barh(x,y)</code>	wykres słupkowy poziomy
<code>comet(x,y)</code>	animowany wykres funkcji pozostawiający jakby ogon komety
<code>compass(x,y)</code>	wykres przedstawiony na tarczy kompasu
<code>contour(x,n)</code>	wykres poziomic
<code>contourf(x,n)</code>	wykres poziomic
<code>errorbar(x,y,e)</code>	wykres funkcji zaznaczając jednocześnie odchylenie
<code>ezplot(f)</code>	wykres dowolnej funkcji
<code>fplot(f,zakres)</code>	wykres funkcji o danej nazwie w podanym zakresie
<code>loglog(x,y)</code>	wykres w układzie dwulogarytmicznym
<code>pie(x,r)</code>	wykres kołowy
<code>plot(x,y)</code>	wykres typu (x,y)
<code>scatter(x,y)</code>	wykres punktowy
<code>semilogx(x,y)</code>	wykres w układzie półlogarytmicznym po x
<code>semilogy(x,y)</code>	wykres w układzie półlogarytmicznym po y
<code>stairs(x,y)</code>	wykres schodkowy
<code>stem(x,y)</code>	odległość wartości y od osi x

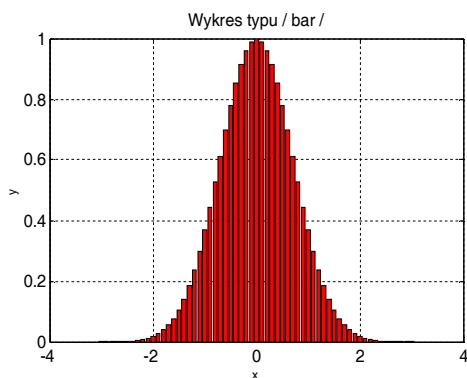
Na rysunkach 4.3-4.12 zilustrowano te typy przykładowymi wykresami.



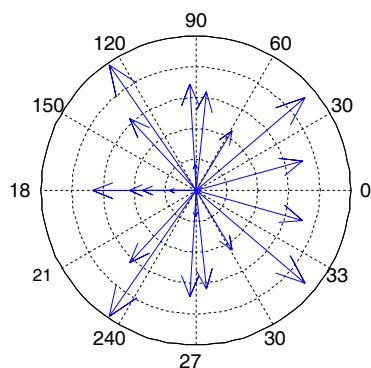
Rys. 4.3. Wykres typu `area(x,y)` funkcji z wypełnionym obszarem pod nią,
a) pojedyncza krzywa z zaznaczonym obszarem, b) krzywe przecinające się z nakładającymi się powierzchniami półprzezroczystymi



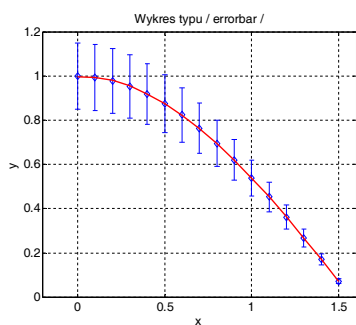
Rys. 4.4. Wykres typu $\text{barh}(x,y)$ słupkowy poziomy:
a) barh , b) barh z opcją 'stack'



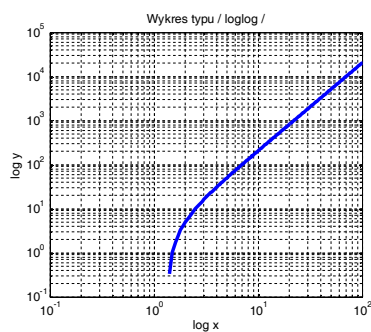
Rys. 4.5. Wykres typu $\text{bar}(x,y)$
rysuje wykres słupkowy



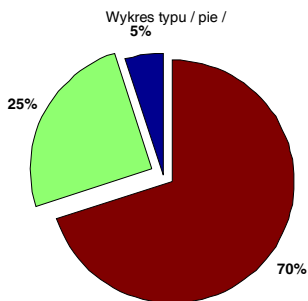
Rys. 4.6. Wykres typu $\text{compass}(x,y)$ przedstawiony
jakby na tarczy kompasu



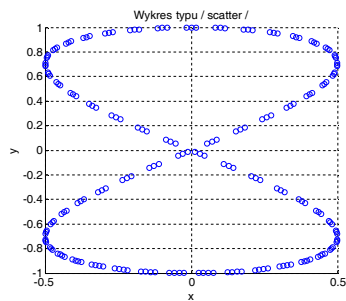
Rys. 4.7. Wykres typu $\text{errorbar}(x,y,e)$
z zaznaczonym jednocześnie odchyleniem



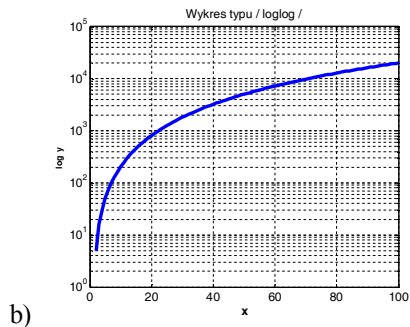
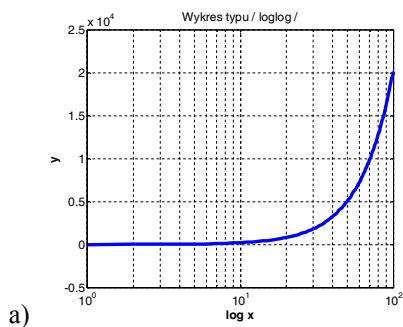
Rys. 4.8. Wykres typu $\text{loglog}(x,y)$ – układ
dwulogarytmiczny



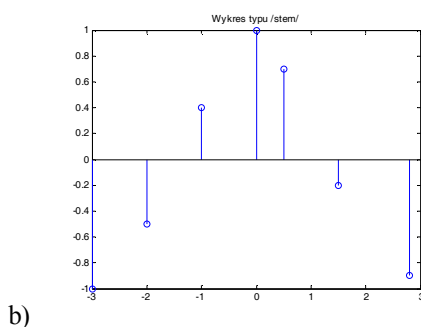
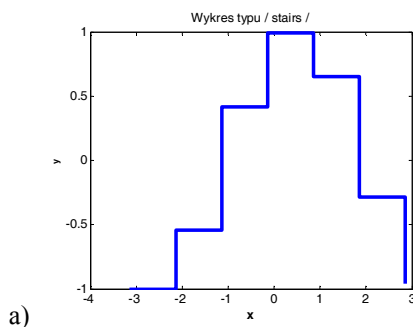
Rys. 4.9. Wykres typu `pie(x,r)` – kołowy



Rys. 4.10. Wykres typu `scatter(x,y)` – wykres punktowy



Rys. 4.11. Wykres typu: a) `semilogx(x,y)` – układ półlogarytmiczny po x, b) `semilogy(x,y)` – układ półlogarytmiczny po y



Rys. 4.12. Wykres typu: a) `stairs(x,y)` – wykres schodkowy, b) `stem(x,y)` – odległość wartości od osi x

4.1.3. Inne funkcje wykorzystywane do pracy z wykresami

Tablica 4.2. Funkcje przydatne w pracy z wykresami

<code>figure(nr)</code>	otwiera okno graficzne, w którym umieszczamy wykres lub obraz
<code>xlabel('tekst')</code>	opis osi ox
<code>ylabel('tekst')</code>	opis osi oy
<code>title('tekst')</code>	tytuł wykresu
<code>axis([xmin xmax ymin ymax])</code>	zakres osi
<code>subplot(m,n,p)</code>	wiele wykresów w jednym oknie, m-liczba wierszy, n-liczba kolumn, p-numer pozycji
<code>legend('tekst','tekst',pozycja)</code>	legenda (pozycja 1, 2, 3, 4 narożnik)
<code>text(x,y,'tekst')</code>	naniesienie tekstu na wykres, x, y – współrzędne lewego początku napisu we jednostkach osi wykresu. W tekście można używać symboli Latexa (Tabl. 2.2).
<code>hold on</code> <code>hold off</code>	włącza/wyłącza rysowanie następnego wykresu w tym samym oknie
<code>[x,y,przycisk]=ginput</code>	polecenie pobiera współrzędne n punktów wskazanych przez użytkownika kursorem i zaznaczonych przez wciśnięcie klawisza lub przycisku myszki; zwracane są współrzędne x i y punktu oraz kod klawisza lub przycisku myszki, działanie funkcji kończymy klawiszem Enter
<code>gtext(napis)</code>	polecenie pozwala umieścić podany napis w miejscu wybranym przez użytkownika myszką
<code>line(X,Y)</code>	rysuje prostą między punktami 1 i 2 gdzie $X = [x_1 \ x_2]$, $Y = [y_1 \ y_2]$

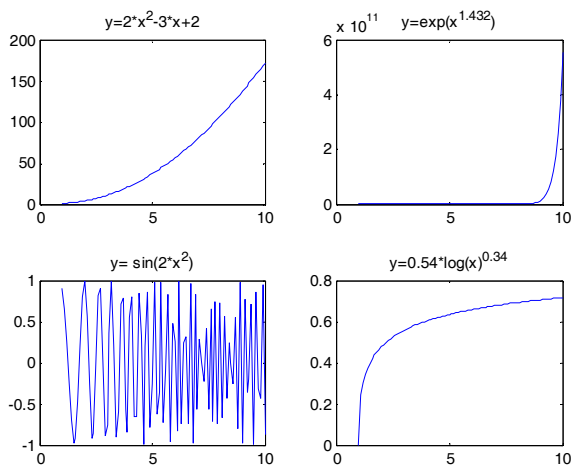
Większość tych funkcji była już wykorzystana w wyżej przedstawionych przykładach, oprócz polecenia `subplot`, które zostało zilustrowane poniżej.

Przykład 4.1. Wykorzystanie polecenia `subplot`

 **p4_1.m**

```
%Przykład 4.1 Cztery okna graficzne
clear all; close all; clc
x=1:0.1:10;
% definiujemy cztery różne funkcje obliczane na tym samym wekto-
    rze argumentów.
y1=2.*x.^2-3.*x+2;
y2=exp(x.^1.432);
y3=sin(2.*x.^2);
y4=0.54.*log(x).^0.34;
figure(1)
% subplot(m,n,p) dzieli okno na m x n pól i otwiera p-te z nich
% do rysowania wykresu
subplot(2,2,1); plot(x,y1)
title('y=2*x^2-3*x+2')
subplot(2,2,2); plot(x,y2)
title('y=exp(x^{1.432})')
subplot(2,2,3); plot(x,y3)
title('y= sin(2*x^2)')
subplot(2,2,4); plot(x,y4)
title('y=0.54*log(x)^{0.34}')
```

Rezultat 4.1



Rys. 4.13. Efekt działania polecenia `subplot`

4.1.4. Style linii i punktów

MATLAB dostarcza cały szereg poleceń pozwalających na sterowanie grubością i stylem linii, kolorem linii i powierzchni, cieniowaniem, oświetlaniem, położeniem punktu, z którego patrzymy na wykres, tytułem wykresu itd. Najważniejsze z nich podamy poniżej.

Tablica 4.3. Style linii używane do kreślenia wykresów

-	ciągła (domyślnie)
--	przerywana
:	kropkowana
-.	przerywana-kropkowana

Tablica 4.4. Rodzaje punktów nanoszonych na wykres

+	znak plus
o	kółko
*	gwiazdka
.	punkt (kropka)
x	krzyżyk
s	kwadrat
d	karo (romb)
^	trójkąt 1. typ
v	trójkąt 2. typ
>	trójkąt 3. typ
<	trójkąt 4. typ
p	5-ramienna gwiazda (pentagram)
h	6-ramienna gwiazda (hexagram)

Tablica 4.5. Podstawowe kolory dostępne przy kreśleniu krzywych i nanoszeniu punktów na wykres

r	czerwony
g	zielony
b	niebieski
c	cyan
m	magenta
y	żółty
k	czarny
w	biały

4.2. Wykresy 3D

4.2.1. Przygotowanie danych

Przed rozpoczęciem rysowania wykresu 3D lub wykresu funkcji dwóch zmiennych $z = f(x, y)$, należy wygenerować specjalną siatkę na płaszczyźnie (x, y) . W jej węzłach będą określone wartości funkcji na osi z . Do tego celu wykorzystuje się funkcję `meshgrid(x, y)`. Definiujemy wektory x , y na których będzie rozpięta siatka. W zależności od tych definicji siatka będzie miała oczka prostokątne bądź kwadratowe.

```
xi=linspace(0,50,100);  
yi= linspace(20,75,100);  
[X,Y]=meshgrid(xi,yi);
```

Funkcja ta transformuje obszar opisany przez wektory xi i yi do dwóch macierzy X i Y , wiersze macierzy X są kopiami wektora x , natomiast kolumny macierzy Y są kopiami wektora y . Otrzymujemy dwie tablice X , Y , w których zdefiniowane są współrzędne siatki.

Konstrukcja ta musi być użyta, jeśli do utworzenia wykresu 3D wykorzystujemy polecenie rysujące powierzchnię. Nie wszystkie polecenia rysujące wykresy 3D wykorzystują ww. konstrukcję, będzie to przedstawione w dalszej części.

4.2.2. Wykres funkcji dwóch zmiennych – siatka lub powierzchnia

Pierwsza linijka programu zawiera funkcje zwalniającą pamięć z wcześniejszych obliczeń:

```
clear all; close all; clc
```

Przygotowanie argumentów:

```
a=linspace(-pi,1*pi,100);  
b=linspace(-pi,1*pi,100);  
[x,y]=meshgrid(a,b);
```

Definiowanie funkcji dwóch zmiennych – zauważmy, że jako argumenty użyto wektorów x oraz y , wcześniej przygotowanych:

```
z=sin(x).*cos(y);
```

Wykresy funkcji dwu zmiennych przedstawia się jako siatkę drucianą (mesh) bądź powierzchnię (surf). Zastosujemy trzy rodzaje poleceń:


```
figure(1);    mesh(x,y,z);surf(x,y,z) ;
figure(2) ;   meshc(x,y,z);surfc(x,y,z) ;
figure(3) ;   meshz(x,y,z);surfl(x,y,z);
```

Otrzymane wykresy są pokazane w następnej sekcji – wykresy typu mesh na rysunkach 4.14, 4.15 i 4.16, a wykresy typu surf na Rys. 4.17-4.19 (dla innych funkcji).

4.2.3. Typy wykresów 3D

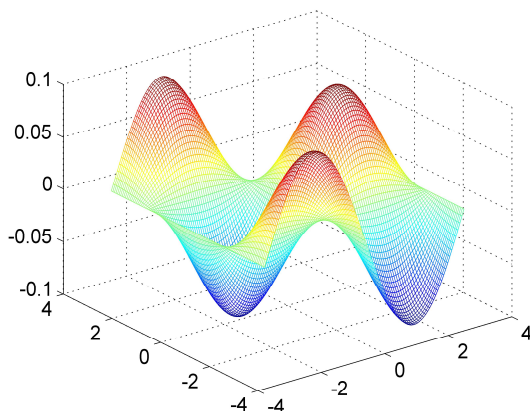
MATLAB udostępnia szereg poleceń, za pomocą których możemy przedstawiać wyniki w postaci trójwymiarowej. W rozdziale tym zostaną one zestawione i przedstawione w skondensowanej formie i dotyczyć będą wykresów funkcji 3D. MATLAB umożliwia również prezentację innych obiektów 3D (zob. sekcja 4.4). Podstawą do tworzenia wykresów 3D są wektory współrzędnych x , y i z węzłów siatki. Muszą one mieć tę samą liczbę elementów.

4.2.3.1. Powierzchnie w przestrzeni

Do budowy wykresów 3D służą następujące funkcje:

mesh(x,y,z)

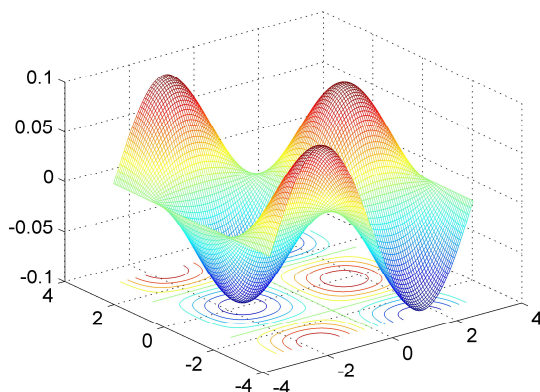
gdzie x , y – wektory węzłów siatki prostokątnej, z – wektor wartości funkcji w tych węzłach.



Rys. 4.14. Polecenie **mesh(x,y,z)** rysuje wykres przestrzenny w postaci siatki, działanie ograniczone do powierzchni zdefiniowanych na regularnych siatkach prostokątnych

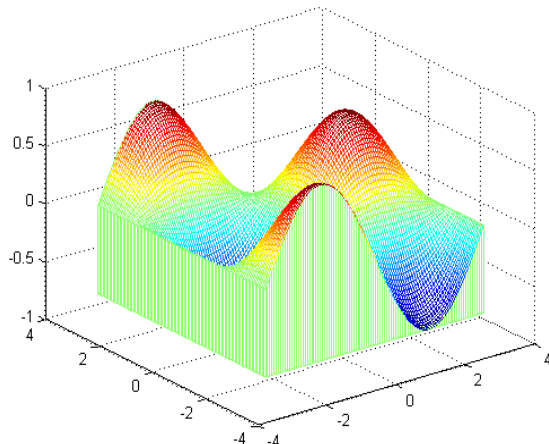
`meshc(x,y,z,c)`

gdzie c – parametr określający liczbę konturów na płaszczyźnie x,y



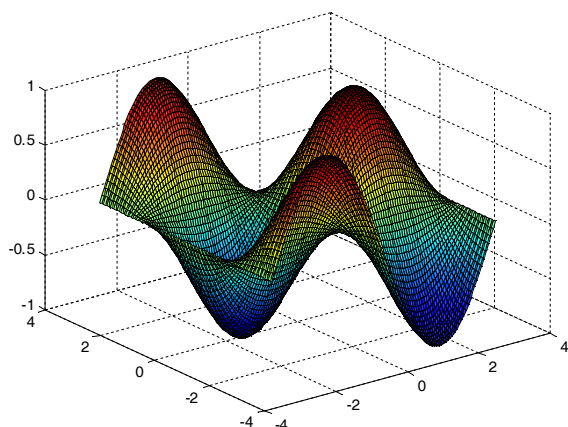
Rys. 4.15. Polecenie `meshc(x,y,z,c)` rysuje siatkę identyczną jak funkcja `mesh` i umieszcza pod nią wykres poziomicowy (łączy działanie `mesh` i `contour`), działanie ograniczone do powierzchni zdefiniowanych na regularnych siatkach prostokątnych

`meshz(x,y,z)`



Rys. 4.16. Polecenie `meshz(x,y,z)` działa podobnie jak `mesh`, z tym, że dodatkowo w dół od krawędzi wykresu są rysowane linie określające płaszczyzny boczne, działanie ograniczone do powierzchni zdefiniowanych na regularnych siatkach prostokątnych

`surf(x,y,z)`



Rys. 4.17. Polecenie `surf(x,y,z)` rysuje różnokolorową powierzchnię wraz z siatką pisaną wektorami x , y , z . Siatkę można usunąć z wykresu poleceniem `shading interp`

Przykład 4.2. Rysowanie kuli za pomocą polecenia `surf(x,y,z)`.

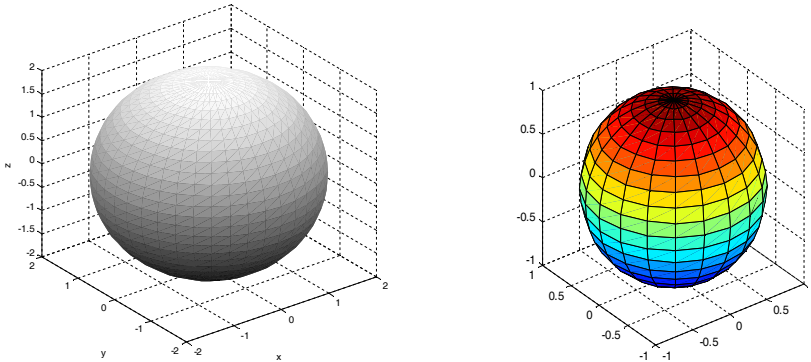
 **p4_2.m**

```
%Przykład 4.2 Rysowanie 3D kuli
close all
r = 2;           % promień
x = linspace(0, 2*pi, 41); % argumenty
y = linspace(-pi/2, pi/2, 31);
[Xs,Ys] = meshgrid(x,y);      % generowanie siatki
% równanie sfery
X = r*cos(Ys).*cos(Xs);
Y = r*cos(Ys).*sin(Xs);
Z = r*sin(Ys);
% wykreślenie powierzchni
surf(X,Y,Z); colormap(gray);
shading flat
```

MATLAB dysponuje również gotowym poleceniem za pomocą którego można wykreślić sferę. Zastępuje ono cały powyższy program. Wystarczy je wpisać z ręki w **Command Window**.

sphere

Rezultat 4.2



Rys. 4.18. Wykreślone powierzchnie kuliste: a) za pomocą polecenia `surf(x,y,z)`, b) za pomocą polecenia `sphere`

`surfc(x,y,z,c)`,

gdzie c – parametr określający liczbę konturów na płaszczyźnie x,y

`surfl(x,y,z,s,k)`

gdzie s – parametr określający kierunek światła, jest wektorem współrzędnych $s = [s_x, s_y, s_z]$, k – parametr określający stałą odbicia, jest wektorem składającym się czterech elementów definiowanych odpowiednio $k = [\text{natężenie światła z otoczenia, odbicie, odbicie lustrzane, współczynnik połysku}]$.

Przykład 4.3. Ilustracja działania funkcji `surfc` i `surfl`

p4_3.m

%Przykład 4.3 Wykresy `surfc` i `surfl`

`close all`

`rv = linspace(.3,1,50);`

`thv = linspace(pi/4,5*pi/4,50);`

`[r,th] = meshgrid(rv,thv);`

`x = r.*cos(th);`

`y = r.*sin(th);`

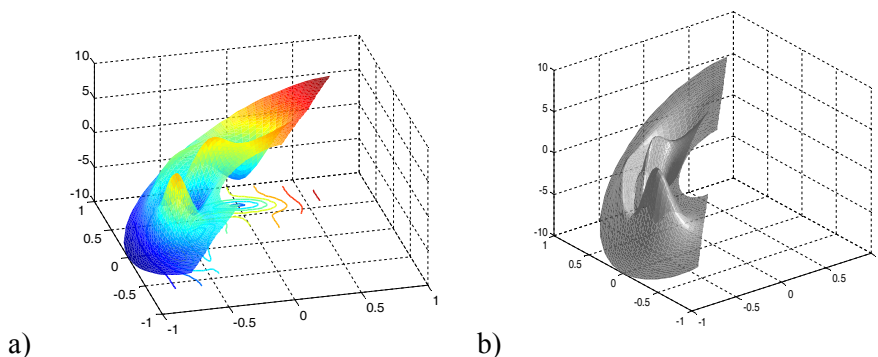
`z = peaks(50) + 10*x;`

`figure(1)`

`surfc(x,y,z)` %wyrysuj pow. i jej rzut jako linie konturowe

```
%funkcje poniżej zob. sekcja 4.4
shading interp
alpha(0.5)      %ustaw przezroczystość
figure(2)
surf1(x,y,z)
shading interp  %ustaw cieniowanie
colormap(gray)  %ustaw paletę barw
```

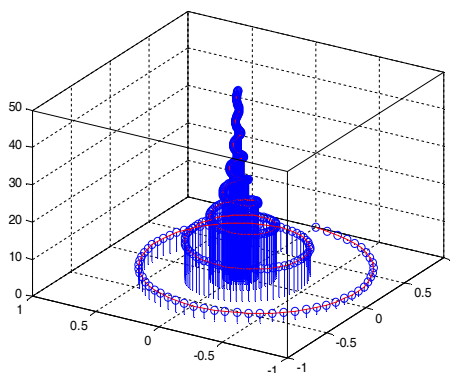
Rezultat 4.3



Rys. 4.19. Polecenie a) `surf(x,y,z)` oraz `surf1(x,y,z)`

4.2.3.2. Krzywe w przestrzeni

```
plot3(x,y,z)
```

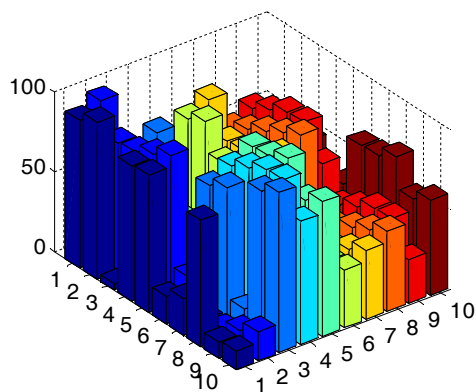


Rys. 4.20. Polecenie `plot3(x,y,z)` rysuje trójwymiarową łamaną łączącą punkty o współrzędnych x , y i z zawartych w odpowiednich wektorach

4.2.3.3. Wykresy danych statystycznych

bar3(M)

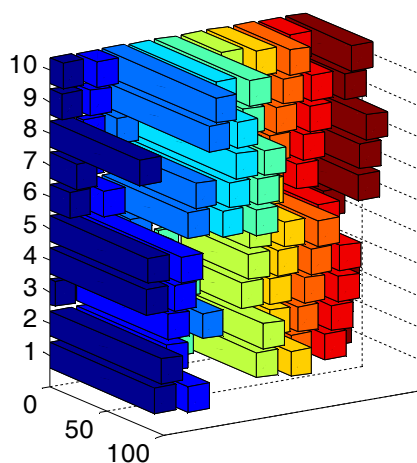
– gdzie M jest macierzą danych.



Rys. 4.21. Polecenie **bar3** rysuje pionowy wykres słupkowy 3D

bar3h(M)

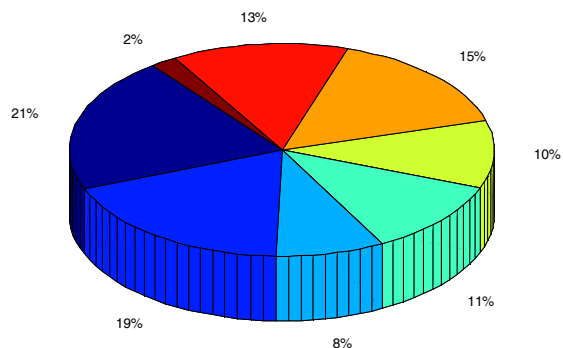
– wykres jw., ale słupki w układzie poziomym.



Rys. 4.22. Polecenie **bar3h** rysuje poziomy wykres słupkowy 3D

`pie3(X)`

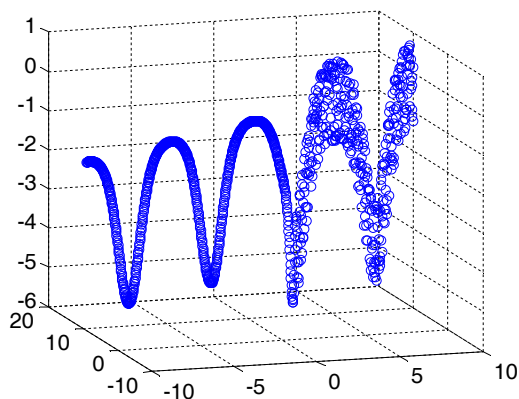
– gdzie X – wektor wartości, których udziały procentowe zostaną pokazane



Rys. 4.23. Polecenie `pie3` rysuje wykres kołowy 3D

`scatter3(x,y,z,s,c)`

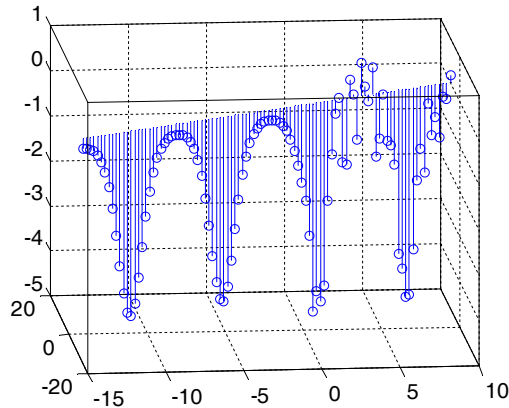
– gdzie x , y , z – wektory współrzędnych punktów, s – rozmiar znacznika, c – kolor znacznika.



Rys. 4.24. Polecenie `scatter3` rysuje wykres punktowy 3D

stem3(x,y,z)

– rysuje punkty o współrzędnych x, y, z prowadząc z nich linie pionowe do z = 0.



Rys. 4.25. Polecenie `stem3` rysuje odległość wartości z od płaszczyzny xy w 3D

4.2.4. Prezentacja przestrzenna funkcji wielu zmiennych

W praktyce inżynierskiej mamy najczęściej do czynienia z dwoma rodzajami funkcji więcej niż dwu zmiennych, są to trójwymiarowe pola skalarne np. pole stężenia czy temperatury i trójwymiarowe pola wektorowe np. pole prędkości. Omówimy tu podstawy wizualizacji takich pól. Zagadnienie to w MATLABie nosi nazwę wizualizacji danych objętościowych (ang. *volume visualization*).

4.2.4.1. Pola skalarne

Pole skalarne jest określone macierzą czterowymiarową, której trzy pierwsze wymiary zawierają współrzędne przestrzenne x, y i z, a czwarty wartość skalarną odpowiadającą tym współrzędnym.

slice(x,y,z,v,xslice,yslice,zslice)

– rysuje wykres funkcji czterowymiarowej v jako przekroje objętości danej wektorami x,y,z w płaszczyznach określonych wektorami xslice, yslice i zslice.

contourslice(x,y,z,v,xslice,yslice,zslice)

– rysuje wykres warstwiczny funkcji czterowymiarowej v w przekrojach objętości danej wektorami x,y,z w płaszczyznach określonych wektorami xslice, yslice i zslice.


```
fz=isosurface(x,y,z,v,c)
```

– wyznacza wartości współrzędnych siatki przestrzennej niezbędne do wykreślenia powierzchni o stałej wartości c dla czterowymiarowej funkcji v w przestrzeni o współrzędnych x,y,z .

Poniższy przykład podaje ilustrację działania tych funkcji.

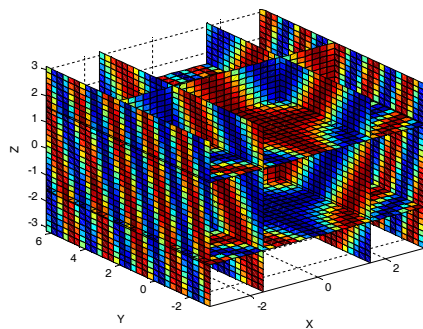
Przykład 4.4.

p4 4.m

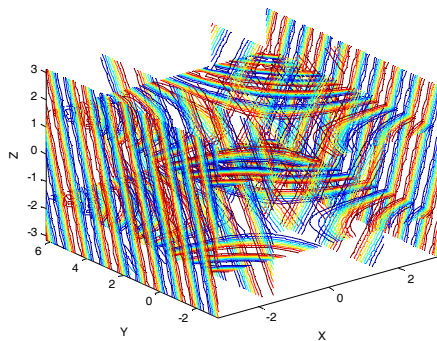
```
%Przykład 4.4 Wizualizacja pola skalarnego
clear all; close all; clc
%Przygotowanie argumentów
xi=linspace(-pi,pi,30);
yi=linspace(-pi,2.*pi,30);
zi=linspace(-pi,pi,30);
[x,y,z] = meshgrid(xi,yi,zi);
%Definiowanie funkcji skalarnej trzech zmiennych
v=sin(x.*y+z);
%Definiowanie widoków przekrojów
xslice = [-pi,-pi/2,pi/2,pi];
yslice = 2; zslice = [-pi/2,pi/2];

%Wykresy
figure(1) %przekroje
slice(x,y,z,v,xslice,yslice,zslice)
axis tight
xlabel('X'); ylabel('Y'); zlabel('Z')
figure(2) %kontury
contourslice(x,y,z,v,xslice,yslice,zslice)
axis tight
xlabel('X'); ylabel('Y'); zlabel('Z')
view(-38,30)
figure(3) %powierzchnie stałej wartości v=0.3
p = patch(isosurface(x,y,z,v,0.3));
set(p,'FaceColor','yellow','EdgeColor','none');
view(-38,30); axis tight
xlabel('X'); ylabel('Y'); zlabel('Z')
camlight right
lighting gouraud
```

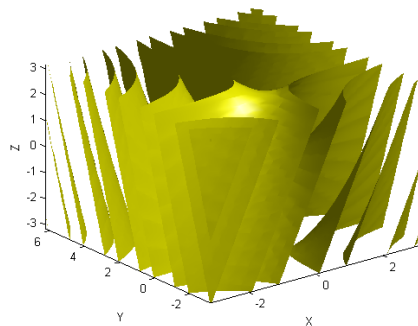
Rezultat 4.4



Rys. 4.26. Efekt działania polecenia slice



Rys. 4.27. Efekt działania polecenia contourslice



Rys. 4.28. Efekt działania polecenia isosurface

4.2.3.2. Pola wektorowe

Pole wektorowe dla każdego punktu o współrzędnych x, y, z ma określony wektor o składowych u, v, w odpowiadających kierunkom osi ox, oy, oz .

Poniżej przedstawiono kilka funkcji służących do wizualizacji takich pól wraz z przykładami.

`quiver3(x,y,z,u,v,w)`

– rysuje wektory o składowych opisanych przez elementy macierzy u, v, w jako strzałki zaczepione w punktach o współrzędnych wyznaczonych przez elementy macierzy x, y, z – macierze muszą mieć identyczne wymiary. Istnieje też funkcja `quiver(x,y,u,v)` dla pól dwuwymiarowych.

`streamslice(x,y,z,u,v,w,sx,sy,sz)`

– rysuje płaskie linie prądu ze strzałkami kierunkowymi pola wektorowego o składowych opisanych przez elementy macierzy u, v, w określone w punktach o współrzędnych x, y, z w przekrojach wyznaczonych wektorami sx, sy, sz .

`streamline(x,y,z,u,v,w,sx,sy,sz)`

– rysuje przestrzenne linie prądu dla pola jw. zaczynające się w punktach startowych wyznaczonych przez wektory sx, sy i sz .

`coneplot(x,y,z,u,v,w,sx,sy,sz,s)`

– rysuje wektory pola o składowych opisanych przez elementy macierzy u, v, w oraz x, y, z jako stożki zaczepione w punktach o współrzędnych wyznaczonych przez elementy macierzy sx, sy, sz . s to współczynnik skali stożków.

Uwaga: w funkcjach `streamslice`, `streamline` i `coneplot` macierze x, y, z oraz u, v, w zawierające komponenty pola wektorowego są trójwymiarowe. Jeśli kartezjańskie wymiary pola wektorowego wyznaczają wektory $X(1:m)$, $Y(1:n)$ i $Z(1:p)$, to macierze współrzędnych mają postać $x(:,X,:)$, $y(Y,,:)$ i $z(:,Z)$. Ta sama zasada dotyczy macierzy składowych u, v, z .

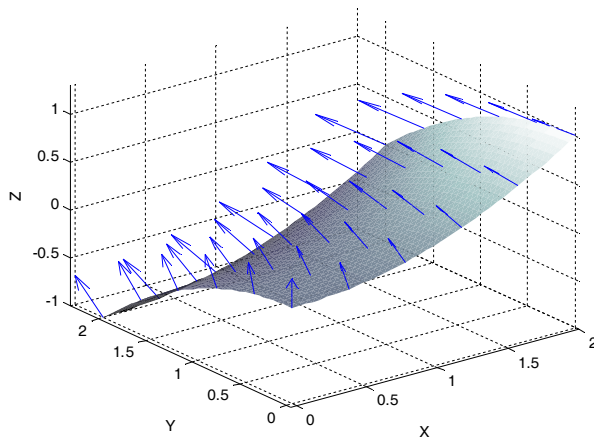
Przykład 4.5. Ilustracja działania funkcji `quiver3`

Wykreślmy powierzchnię zdefiniowaną następująco: $f(x, y, z) = z + (y^2 - x^2)/4$, i pole wektora nachylenia określone przez $\nabla f(x, y, z) = [-x/2, y/2, 1]$. Pole wektora nachylenia to właśnie pole wektorowe.

p4_5.m

```
%Przykład 4.5 Funkcja quiver3
clear all; clf; clc
%współrzędne punktów, do których
%przymocowano wektory normalne
%siatka ma małą gęstość
[X,Y] = meshgrid(0:.4 :2);
Z = .25*(X.^2 - Y.^2);
%składowe wektorów normalnych
U = -X/2;
V = Y/2;
W = 1+0*X; %dodanie 0*X tworzy wektor
%kreślenie powierzchni
%tu siatka ma dużą gęstość
[XX,YY] = meshgrid(0: .05: 2);
ZZ = .25*(XX.^2 - YY.^2);
surf(XX,YY,ZZ);
xlabel('X'); ylabel('Y'); zlabel('Z');
shading interp; colormap bone
hold on
%kreślenie wektorów normalnych
quiver3(X,Y,Z,U,V,W)
axis tight
```

Rezultat 4.5



Rys. 4.29. Powierzchnia $f(x,y,z)$ i wektory normalne do niej

Przykład 4.6. Ilustracja działania funkcji coneplot i streamslice

Pole wektorowe w tym przykładzie zostało obliczone jako pole wektorów normalnych do funkcji $v = \sin(x \cdot y + z)$ z przykładu 4.4.

 p4_6.m

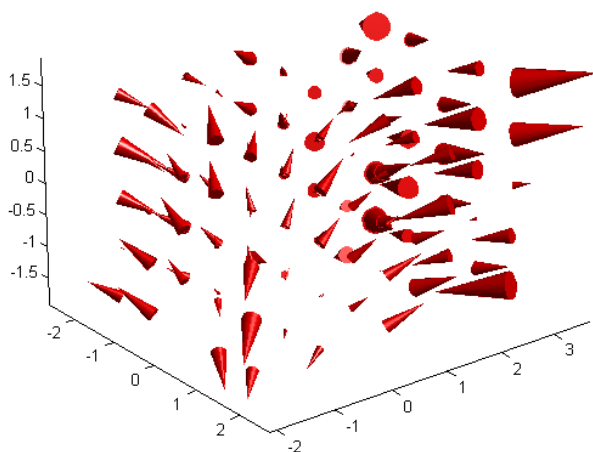
```
%Przykład 4.6 Wizualizacja pola wektorowego
clear all; close all; clc
%Przygotowanie argumentów
n=10;
xi=linspace(-pi/2,pi/2,n);
yi=linspace(-pi/2,pi,n);
zi=linspace(-pi/2,pi/2,n);
%Definiowanie funkcji wektorowej trzech zmiennych
%---macierze współrzędnych
for j=1:n
    for i=1:n
        x(i,:,j)=xi;
        y(:,i,j)=yi;
        z(i,j,:)=zi;
    end
end

%---macierze składowych wektorów
u=y.*cos(x.*y+z);
v=x.*cos(x.*y+z);
w=1.*cos(x.*y+z);
%---obliczanie rozmiarów stożków
xmin = min(x(:)); xmax = max(x(:));
ymin = min(y(:)); ymax = max(y(:));
zmin = min(z(:)); zmax = max(z(:));
xrange = linspace(xmin,xmax,5);
yrange = linspace(ymin,ymax,5);
zrange = linspace(zmin,zmax,5);
[cx cy cz] = meshgrid(xrange,yrange,zrange);

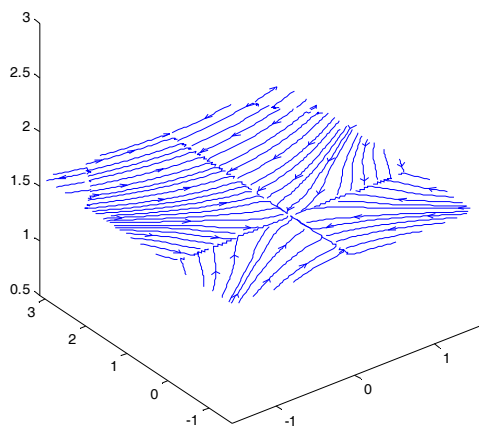
%---wykonanie wykresu stożkowego
figure(1)
hcones = coneplot(x,y,z,u,v,w,cx,cy,cz,2);
set(hcones,'FaceColor','red','EdgeColor','none')
view(52,22);axis image
camproj perspective;
camlight right; lighting phong
```

```
%--wykonanie wykresu linii prądu w danym przekroju
figure(2)
streamslice(x,y,z,u,v,w,[],[],(zmax-zmin)/2)
view(3); axis tight
```

Rezultat 4.6



Rys. 4.30. Wykres pola wektorowego w postaci stożków



Rys. 4.31. Wykres pola wektorowego w postaci przekroju z liniami prądu

4.3. Grafika za pomocą uchwytów obiektów graficznych

Grafika ta (ang. *handle graphics*) opiera się na zasadach programowania obiektowego. Wszystkie elementy graficzne wykresów w MATABie są obiektami i posiadają listę właściwości. Niektóre z nich posiadają elementy podległe („dzieci”), które dziedziczą po nich właściwości. Dzieci posiadają też swe własne właściwości. Dla przykładu osie wykresu posiadają szereg właściwości takich jak kolor, grubość linii, krój czcionki i wielkość czcionki do opisu itd. Ich elementem podległym jest siatka wykresu, która dziedziczy np. kolor i grubość linii, ale także ma własne właściwości jak np. osiatkowanie poziome i pionowe.

Właściwości te można zmieniać podczas edycji rysunku w oknie **Figure**, korzystając z menu tego okna. Jednakże dostęp do właściwości obiektów graficznych jest także możliwy z poziomu programu. Cały zestaw właściwości danego obiektu graficznego mieści się pod adresem pamięci, zapisanym w uchwycie obiektu (ang. *handle*), jak np. uchwyt aktualnego rysunku, który ma nazwę **gcf** czy uchwyt aktualnego układu osi **gca**. Można ich użyć w programie lub wyświetlić po wykonaniu dowolnego wykresu.

Aby uzyskać uchwyt danego obiektu, można dokonać podstawienia pod zmienną wykonywanego polecenia, np. zapisując

```
h = xlabel('temperatura [C]')
```

Po wykonaniu tego polecenia otrzymamy np.

```
h =  
189.0023
```

gdzie h to uchwyt aktualnego polecenia **xlabel**. Aby sprawdzić jakie wszystkie właściwości posiada dany obiekt graficzny, należy użyć polecenia **set** np.

```
set(h)
```

wyświetli wszystkie właściwości obiektu **xlabel** wraz z dostępnymi opcjami. Aktualna opcja objęta jest nawiasami { }.

```
BackgroundColor  
Color  
DisplayName  
EdgeColor  
Editing: [ on | off ]  
FontAngle: [ {normal} | italic | oblique ]  
FontName  
FontSize
```

```

FontUnits: [ inches | centimeters | normalized | {points} | pixels
            | {characters} ]
FontWeight: [ light | {normal} | demi | bold ]
HorizontalAlignment: [ {left} | center | right ]
LineStyle: [ {-} | -- | : | -. | none ]
LineWidth
Margin
Position
Rotation
String
Units: [ inches | centimeters | normalized | points | pixels |
        | characters | {data} ]
Interpreter: [ latex | {tex} | none ]
VerticalAlignment: [ top | cap | {middle} | baseline | bottom ]
ButtonDownFcn: string -or- function handle -or- cell array
Children
Clipping: [ {on} | off ]
CreateFcn: string -or- function handle -or- cell array
DeleteFcn: string -or- function handle -or- cell array
BusyAction: [ {queue} | cancel ]
HandleVisibility: [ {on} | callback | off ]
HitTest: [ {on} | off ]
Interruptible: [ {on} | off ]
Parent
Selected: [ on | off ]
SelectionHighlight: [ {on} | off ]
Tag
UIContextMenu
UserData
Visible: [ {on} | off ]

```

Za pomocą polecenia `get(h)` dostaniemy natomiast wykaz wszystkich aktualnych właściwości obiektu o uchwycie `h`.

```

Annotation = [ (1 by 1) hg.Annotation array]
BackgroundColor = none
Color = [0 0 0]
DisplayName =
EdgeColor = none
Editing = off
Extent = [30.5287 -1.81924 18.3908 0.886297]
FontAngle = normal
FontName = Helvetica

```



```

FontSize = [10]
FontUnits = points
FontWeight = normal
HorizontalAlignment = center
LineStyle = -
LineWidth = [0.5]
Margin = [2]
Position = [39.9078 -1.09942 1.00011]
Rotation = [0]
String = Temperatura [\circC]
Units = data
Interpreter = tex
VerticalAlignment = cap

BeingDeleted = off
ButtonDownFcn =
Children = []
Clipping = off
CreateFcn =
DeleteFcn =
BusyAction = queue
HandleVisibility = off
HitTest = on
Interruptible = on
Parent = [173.002]
Selected = off
SelectionHighlight = on
Tag =
Type = text
UIContextMenu = []
UserData = []
Visible = on

```

Właściwości te można zmieniać, stosując dopuszczalne nazwy i wartości parametrów sterujących np. polecenie

```
set(h, 'FontSize', 20)
```

zmienia rozmiar czcionki przypisanej obiektowi `xlabel` na 20 punktów.

Warto nadmienić, że początkujący mogą po wykonaniu wstępnego wykresu zapisać go w formacie m-pliku, gdzie znajdą wszystkie polecenia sterujące aktualnymi ustawieniami wykresu, co można wykorzystać do jego modyfikacji za

pomocą programu. Robi się to z poziomu menu **Figure** przez wybranie opcji **File/Generate M-File**.

Przykład poniżej ilustruje część możliwości praktycznego wykorzystania *handle graphics*.

Przykład 4.7. Wykres metody wyznaczania NTU w rektyfikacji na wypełnieniu

p4_7.m

%Przykład 4.7 Wykres NTU za pomocą Handle Graphics

function NTU

clear all; close all; clc;

global xp

%dane

alfa=2.3; %lotność względna

xD=0.95; %uł. mol. destylatu, -

xF=0.4; %uł. mol. surówki, -

xW=0.03; %uł. mol. cieczy wyczerp. -

R=2.1; %liczba powrotu, -

e=1.5; %liczba stanu cieplnego surówki, -

%duży wykres kwadratu jednostkowego i linii:

yeq=@(x) alfa*x./((alfa-1)*x+1); %równowaga

GLO=@(x) R./(R+1)*x+1./(R+1)*xD; %górną linią operacyjną

EL =@(x) e./(e-1)*x-1./(e-1)*xF; %linia e

fx=@(x) GLO(x)-EL(x);

xp=fzero(fx,xF); %znalezienie punktu przecięcia GLO i EL

DLO=@(x) xW+(GLO(xp)-xW)./(xp-xW)*(x-xW); %dolną linią operacyjną

X =linspace(0,1,50);

xd=linspace(xW,xp,20);

xg=linspace(xp,xD,20);

xe=linspace(xF,xp+0.1,10);

y=yeq(x);

yg=GLO(xg);

ye=EL(xe);

yd=DLO(xd);

axes('position',[.1 .1 .8 .8]);

hold on

plot(x,y,'k');

plot(x,x,'--');

```

plot(xg,yg,'b','LineWidth',2);
plot(xe,ye,'c');
plot(xd,yd,'r','LineWidth',2);
axis([0 1 0 1]);
set(gca,'GridLineStyle','-','PlotBoxAspectRatio',[1 1 1], 'Xcolor',
      'b','Ycolor','b');
hl=legend('linia równowagi','przekątna','GLO', 'linia e', 'DLO');
set(hl,'FontName','Times','Location','NorthWest');
h1=xlabel('x [-]'); h2=ylabel('y [-]');
set(h1,'FontSize',20); set(h2,'FontSize',20);
grid on

%mały wykres odwrotności siły napędowej
GNT=@(x) 1./(yeq(x)-GLO(x));
DNT=@(x) 1./(yeq(x)-DLO(x));
NTU=quad(DNT,xw,xp)+quad(GNT,xp,xD);

%Wzór na NTU w LaTeXie
ha=axes('position',[.50 .15 .3 .3]);
ht=text(0.10,35,'$$NTU=\int_{y_{W}}^{y_{D}}\frac{dy}{y^{*}-y}$$');
set(ht,'Interpreter','latex');
yd=DNT(xd);
yg=GNT(xg);
hold on
h1=area(xd,yd);
h2=area(xg,yg);
set(h1,'FaceColor',[1 0.5 0.4]);
set(h2,'FaceColor',[0.2,0.4,1]);
set(ha,'FontName','Times','FontAngle','italic');
xlabel('x'); ylabel('1/(y^{*}-y)');
set(ha,'FontName','Times','FontAngle','italic');
xlabel('x'); ylabel('1/(y^{*}-y)');

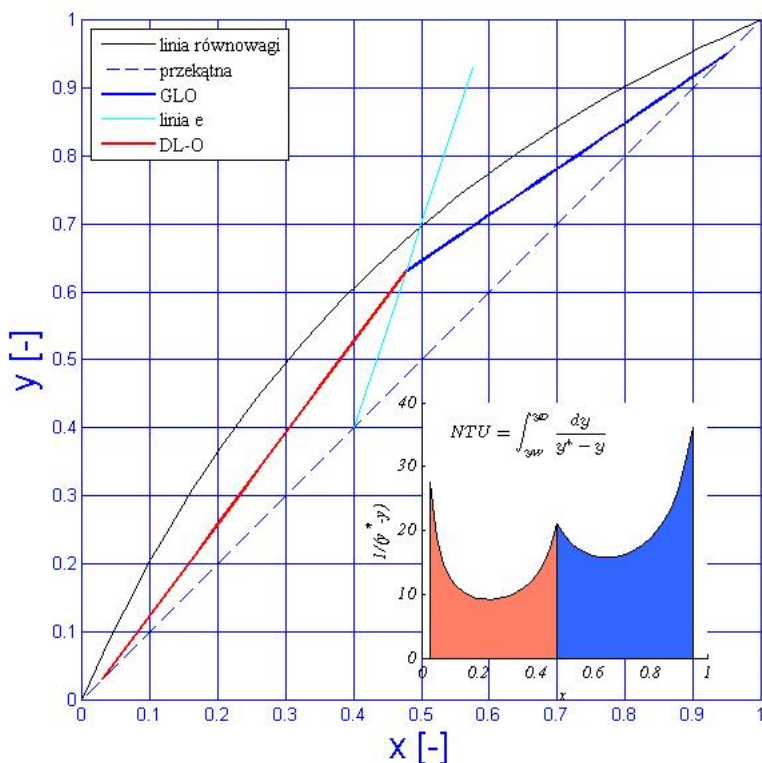
%wydruk obliczonej wartości NTU
disp(['Obliczone NTU = ',num2str(NTU)]);

```

Rezultat 4.7

W oknie poleceń program wydrukował:

```
Obliczone NTU = 14.7256
```



Rys. 4.32. Ilustracja niektórych możliwości *handle graphics* – ilustracja wyznaczania NTU w rektyfikacji dwuskładnikowej na wypełnieniu

PROJEKT 4.1

Temat: Wykres Molliera

Toolbox: -

Wykres Moliera, zwany także wykresem Molliera-Ramzina, jest wykresem dwu funkcji dla wilgotnego gazu: wilgotności względnej i temperatury w skośnym układzie współrzędnych wilgotność bezwzględna-entalpia.

Wilgotność względna jest to ilość kilogramów pary substancji na kilogram suchego gazu (np. powietrza).

Wilgotność bezwzględna jest to stosunek aktualnego ciśnienia cząstkowego pary do ciśnienia pary nasyconej w tej samej temperaturze.

$$\varphi = \frac{p}{p_s(T)}. \quad (4.1)$$

Relacja pomiędzy obiema wilgotnościami jest następująca

$$Y = \frac{M_A}{M_B} \frac{\varphi p_s(T)}{P - \varphi p_s(T)}. \quad (4.2)$$

Entalpia gazu wilgotnego jest obliczana z wyrażenia

$$i = (c_{Bm} + c_{Am} Y) T + \Delta h_{v0} Y \quad (4.3)$$

gdzie c_{Am} i c_{Bm} – średnie ciepła właściwe pary i powietrza (w zakresie temperatur od 0 do danej), kJ/(kg K) T – temperatura, °C, Y – wilgotność względna, kg/kg, Δh_{v0} – ciepło parowania w temperaturze 0°C, kJ/kg.

Wykres Molliera posiada skośny układ współrzędnych (Y , i). Pozostałe linie – izotermy i linie stałej wartości φ bywają ręcznie nanoszone na tej siatce współrzędnych. Jednakże, wykonując wykres, automatycznie będziemy posługiwać się układem kartezjańskim współrzędnych (x , y). W tym układzie równania poszczególnych linii będą miały postać (Pakowski, 1986)

- Siatka Y

$$x_Y = Y / s_Y \quad (4.4)$$

- Siatka i

$$y_i = -\Delta h_{v0} \frac{s_Y}{s_i} x_Y + i / s_i \quad (4.5)$$

- Siatka t

$$y_t = c_{Am}(T) T \frac{s_Y}{s_i} x_Y + T / s_i(T), \quad (4.6)$$

gdzie

$$s_t(T) = \frac{s_i}{c_{Bm}(T)} \quad (4.7)$$

- Siatka φ

$$y_{\varphi} = y_t(T_{\varphi}) \quad (4.8)$$

gdzie T_{φ} jest obliczane z zależności 4.4 i 4.2 dla danego x oraz Y . Współczynniki s_Y i s_i to współczynniki skali. Powstają one przez podzielenie maksymalnej wartości skali Y oraz i przez odpowiedni wymiar liniowy (długość) osi.

P4_1.m

```
%PROJEKT 4.1
%Wykres Molliera-Ramzina
%dla układu powietrze-para wodna
function Mollier
clear all; close all; clc;
global P0 fi Y
% warning off all
sY=1;
si=1;
hv0=hvp(0);
%-----
P0=101.325;      % ciśnienie otoczenia [kPa]
fip=0.1;         % początkowa wilgotność względna
dfi=0.1;        % krok wilgotności względnej
Yp=1e-9;        % pocz. wilg. bezwzględna [kg/kg]
nY=1000;        % liczba kroków wilg. bezwzględnej [kg/kg]
Yk=.1;          % końc. wilg. bezwzględna [kg/kg]
Tp=10;          % początkowa izoterma [C]
dT=10;          % krok izotermy [C]
Tk=200;         % końcowa izoterma[C]
ip=40;          % początkowa entalpia [kJ/kg]
di=20;          % krok entalpii [kJ/kg]
ik=500;         % końcowa entalpia [kJ/kg]
opis=1;         % włącz opis =1 wyłącz opis = 0
fs=8;          % rozmiar czcionki opisu wykresu
%-----
% wektor wilgotności względnej
fz=(fip:dfi:1)';
% wektor wilgotności bezwzględnej
Yz=linspace(Yp,Yk,nY);
% obliczam temperaturę wrzenia cieczy pod zadany ciśnieniem
Tb=fzero(@psTb,100);
g1=(cp(Tb).*Tb*(sY./si).*(Yz./(si./cg(Tb)))+cg(Tb).*Tb./si)';
% obliczam linie wilgotności względnej
```

```

for i=1:length(fz)
    fi=fz(i);
    t0=-80;
    for j=1:length(Yz)
        Y=Yz(j);
        yt(j,i)=fzero(@cis,t0);
        t0=floor(yt(j,i));
        yfi(j,i)=cp(yt(j,i)).*yt(j,i).*(sY./si).*(
            Y./sY)+yt(j,i)./(si./cg(yt(j,i)));
    end
end
%-----Budowa wykresu-----
figure(1)
% procedura rysująca linie wilgotności względnej
[~,kg]=size(yfi);
for i=1:kg
    ypg=yfi(:,i);
    nrg=find(ypg <= g1);
    plot(Yz(nrg),ypg(nrg),'g')
    hold on
    [w1,~]=size(nrg);
    str = num2str(fz(i));
    if opis == 1
text(Yz(nrg(w1)),yfi(nrg(w1),i),str, 'FontSize',fs, ...
    'HorizontalAlignment','center', 'EdgeColor','g', 'Back-
    groundColor',[1 1 1]);
    end
end
end
[w1,k1]=size(yfi);
% obliczam izotermy dla powietrza wilgotnego
Tz=(Tp:dT:Tk)';
for i=1:length(Tz)
    T=Tz(i);
    for j=1:length(Yz)
        Y=Yz(j);
        yt(j,i)=cp(T).*T*(sY./si).*(Y./((si./cg(T)))+
            cg(T).*T./si;
    end
end
end
% procedura rysująca i opisująca izotermy
[~,k2]=size(yt);
for i=1:k2
    ypz=yt(:,i);

```

```

nr=find(ypz >= yfi(:,k1));
[wn,~]=size(nr);
plot(Yz(nr),ypz(nr),'r','LineWidth',1)
str = num2str(Tz(i));
if opis == 1
    yy=ypz(nr(round(0.5*wn))); %jeśli pionowa współrzędna
    tagu
    if yy<yfi(w1,1) %nie wychodzi za ramkę
        text(Yz(1,nr(round(0.5*wn))),yy,str, 'FontSize',fs,...
            'HorizontalAlignment','center', 'EdgeCol-
            or','r','BackgroundColor',[1 1 1]);
    end
end
end
plot(Yz,cp(Tb).*Tb*(sY./si).*(Yz./(si./cg(Tb)))+cg(Tb).*Tb./si,'
    r-.' )
hold on
% obliczam entalpie dla powietrza wilgotnego
iz=(ip:di:ik)';
for k=1:length(iz)
    i=iz(k);
    for j=1:length(Yz)
        Y=Yz(j);
        yi(j,i)=-hv0.*(sY./si).*Y+i./si;
    end
end
end
%procedura rysująca izoentalpy
[~,k3]=size(yi);
for i=1:k3
    yiz=yi(:,i);
    nr=find(yiz >= yfi(:,k1));
    plot(Yz(nr),yiz(nr),'b')
end

axis([0 Yk 0 yfi(w1,1)])
pres=[num2str(P0) ' kPa'];
title(['Wykres Molliera. Ukł. woda-powietrze. P=' pres])
xlabel('Y [kg/kg]'); ylabel('i [kJ/kg]')
text(0.85*Yk,20,'t=const','EdgeColor','r')
text(0.85*Yk,10,'\phi=const','EdgeColor','g');
set(gca,'XGrid','on','fontsize',6)

```



```

%Funkcje opisujące właściwości
%Aby wykonać wykres dla innego układu gaz-ciecz należy
%jedynie wymienić te funkcje.

%-----
function y=cis(T)
% równanie do przeliczania wilgotności bezwzględnej na względną
global P0 fi Y
M=18.02; Mg=28.9645;
y=((M./Mg).*((fi*ps(T))./(P0-fi.*ps(T))))-Y;

%-----
function y=psTb(T)
% równanie do obliczania temperatury wrzenia w [C]
global P0
y=ps(T)-P0;

%-----
function y=ps(T)
% prężność pary [kPa] od temperatury [C]
a1=16.376953; a2=3878.8223; a3=229.862;
y=exp(a1-a2/(a3+(T)));

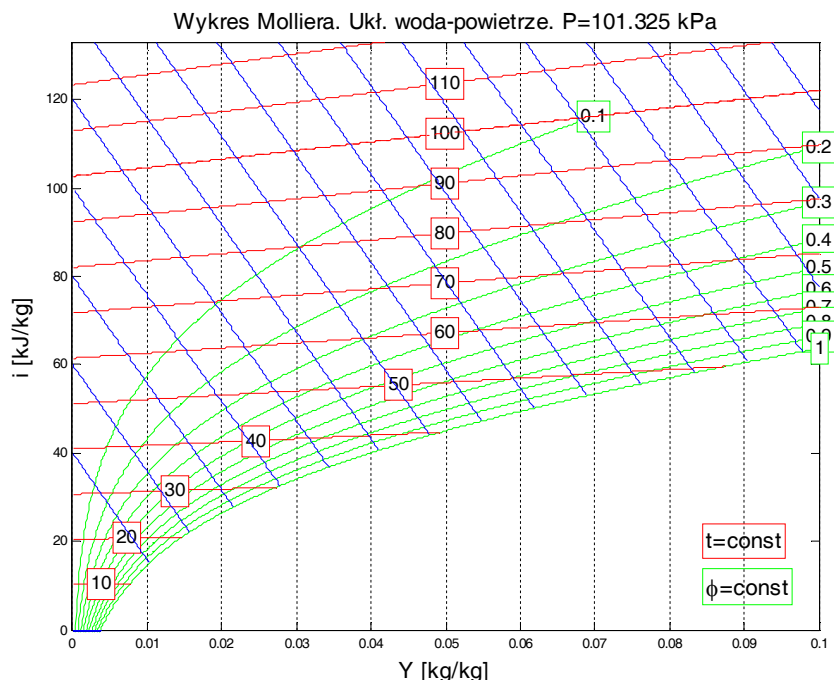
%-----
function y=hvp(T)
% ciepło parowania [kJ/kg] od temperatury [C]
a1=52.053; a2=647.13; a3=0.321;
y=(a1.*(1-(T+273.15)./a2).^a3).*(1000./18.015);

%-----
function y=cp(T)
% ciepło właściwe pary [kJ/(kg*K)] od temperatury [C]
a1=33.933; a2=-8.4186e-3; a3=2.9906e-5; a4=-1.7825e-8;
a5=3.6934e-12;
y=(a1+a2.*(T+273.15)+a3.*(T+273.15).^2+a4.* ...
(T+273.15).^3+a5.*(T+273.15).^4).*(1./18.015);

%-----
function y=cg(T)
% ciepło właściwe powietrza [kJ/(kg*K)] od temperatury [C]
a1=1.02287;a2=-0.0512e-3; a3=0.181871e-6; a4=-0.05122e-9;
y=(a1+a2.*(T+273.15)+a3.*(T+273.15).^2+a4.*(T+273.15).^3);

```

Wyniki 4.1



Rys. 4.33. Wykres Molliera dla układu woda-powietrze

Uzyskany wykres można modyfikować, zmieniając ciśnienie ogólne lub składniki układu. Można też dodatkowo nanieść nań izotermy wilgotnego termometru i/lub adiabatycznego nasycenia. Łatwo też nanieść na wykres macierz wyników obliczonych trajektorii procesu w postaci zależności np. Y od t .

PROJEKT 4.2

Temat: Trójwymiarowy wykres równowagi ciecz-para dla układu dwuskładnikowego

Toolbox: -

Równowaga ciecz-para dla układu dwuskładnikowego jest tradycyjnie przedstawiana jako tzw. „rybka” w dwuwymiarowym układzie współrzędnych temperatura od ułamka molowego lub ciśnienie od ułamka molowego. Jednak najlepiej pokazać tę równowagę na wykresie 3D.

W tym projekcie wygenerowano wykres 3D dla układu metanol (1) –woda (2). Wykres jest wykonany we współrzędnych temperatura (x), ułamek molowy składnika 1 (y), ciśnienie (p) i składa się z dwu powierzchni – powierzchni wrzącej cieczy i powierzchni kondensującej pary (punkt rosy). Powierzchnię cieczy wrzącej wyznacza się dla wszystkich węzłów siatki t, x_1 , sumując udziały prężności pary obu składników

$$P(x_1, t) = \gamma_1 x_1 p_{s1}(t) + \gamma_2 x_2 p_{s2}(t), \quad (4.9)$$

gdzie P – ciśnienie ogólne, kPa, x – ułamek molowy, g – współczynnik aktywności, p_s – prężność pary nasyconej, kPa, t – temperatura, °C. Współczynniki aktywności policzono metodą Wilsona, prężności par nasyconych i inne właściwości pobrano z książki *Pakowski, Głębowski, 2000*.

Otrzymujemy macierz powierzchni cieczy wrzącej (zmienna CC w programie).

Prężność pary składnika 1 w kondensującej parze dla danej temperatury i ułamka molowego y_1 oblicza się z prawa Raoult'a

$$y_1 P(z_1, t) = \gamma_1 z_1 p_{s1}(t). \quad (4.10)$$

W równaniu tym z_1 to taki ułamek molowy w cieczy, dla którego ułamki molowe w parze y_1 leżą w tych samych węzłach siatki co ułamki molowe w cieczy użyte do konstrukcji powierzchni wrzenia.

Znalezienie z_1 wymaga rozwiązania równania 4.10 dla kolejnych wartości $z_1 = y_1 = x_1$, gdzie x_1 jest wektorem wartości x_1 w węzłach siatki współrzędnych. Otrzymujemy macierz powierzchni kondensującej pary (zmienna PP w programie). Otrzymane macierze wykreślamy za pomocą polecenia `surf` lub `surface` – polecenie `surface` umożliwia lokalne przypisanie współczynnika przezroczystości `alpha` do danej powierzchni. Dodatkowo poleceniem `contour3` nanosimy warstwicę stałego ciśnienia. Powstały wykres rzutowany na powierzchnię x_1, t daje „rybkę” dla $p = \text{const.}$, a na powierzchnię x_1, p dla $t = \text{const.}$

P4_2.m

```
%Projekt 4.2
%Równowaga ciecz-para układu metanol(1)-woda(2)
function rekaw
clear all; close all; clc;
n=20;
m=20;
x1=linspace(0,1,n);
x1(1)=0.0001; x1(n)=0.9999;
t =linspace(40,110,m);
```

```

x=[x1;1-x1];

%powierzchnia cieczy wrzącej
for j=1:m
    for i=1:n
        CC(i,j)=sum(Wilson(x(:,i),t(j)).*x(:,i).*ps(t(j)),1);
    end
end

%powierzchnia punktu rosy
y=x;
for j=1:m
    for i=1:n
        z=xx(y(:,i),t(j));
        a=Wilson(z,t(j));
        b=ps(t(j));
        PP(i,j)=a(1).*b(1).*z(1)./y(1,i);
    end
end

contour3(t,x1,CC,[50 100 150 200 250],'y');
h1=surface(t,x1,CC);
hold on
contour3(t,x1,PP,[50 100 150 200 250],'y');
surf(t,x1,PP);
alpha(h1,0.5)
shading interp
colormap autumn
xlabel('t [\circ C]'); ylabel('x [-]'); zlabel('p [kPa]');

function x=xx(y,t)
%znajduje wartość ułamka molowego w cieczy,
%który w tej samej temperaturze daje takie
%samo ciśnienie ogólne
x0=[0.5];
F=@(z) LHS(y(1),z,t);
x=fzero(F,x0);
x=[x; 1-x];

function z=LHS(y1,x1,t)
y=[x1; 1-x1];
g=Wilson(y,t);
psat=ps(t);

```

```

sumP=sum(g.*y.*psat);
z=y1*sumP-g(1)*x1*psat(1);

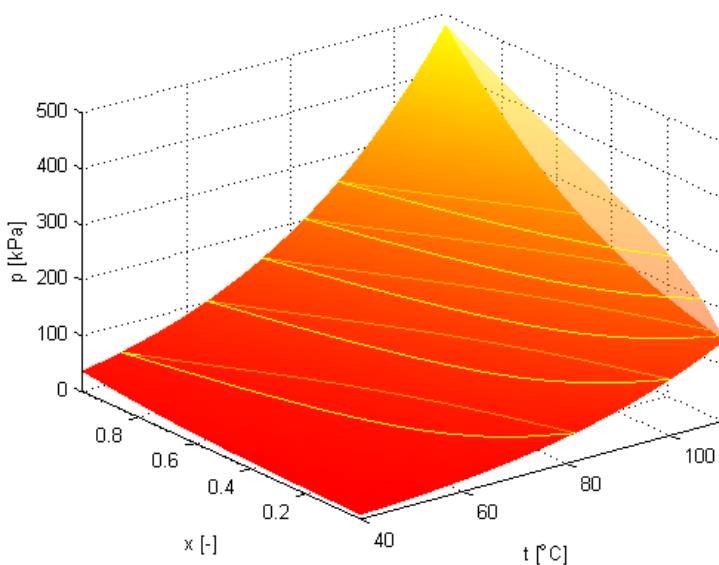
function p=ps(t)
c=[45.6171 -3.2447e3 -1.3988e1 6.6365e-3 -1.0507e-13;
    29.8605 -3.1522e3 -7.3037e0 2.4247e-9 1.8090e-6];
T=t+273.15;
w1=c(1,1)+c(1,2)./T+c(1,3)*log10(T)+c(1,4)*T+c(1,5)*T.^2;
w2=c(2,1)+c(2,2)./T+c(2,3)*log10(T)+c(2,4)*T+c(2,5)*T.^2;
p(1)= 0.133322368*10.^w1;
p(2)= 0.133322368*10.^w2;
p=p';

function d=rho(t)
r=[ 0.27197 0.27192 0.23310 512.58;
    0.34710 0.27400 0.28571 647.13];
T=t+273.15;
d(1)=1e3*r(1,1)*r(1,2).^(-(1-T./r(1,4)).^r(1,3));
d(2)=1e3*r(2,1)*r(2,2).^(-(1-T./r(2,4)).^r(2,3));

function gamma=Wilson(x,t)
T=t+273.15;
R=8.314;
RT=R*T;
Mol1=32.042;
Mol2=18.01;
lambda1211=449.59198;
lambda2122=1965.9157;
ro=rho(t);
vm1=Mol1./ro(1);
vm2=Mol2./ro(2);
Lam(1)=vm2./vm1*exp(-lambda1211./RT);
Lam(2)=vm1./vm2*exp(-lambda2122./RT);
c1=Lam(1)./(x(1)+Lam(1)*x(2));
c2=Lam(2)./(x(2)+Lam(2)*x(1));
gamma(1)=exp(-log(x(1)+Lam(1)*x(2))...
    +x(2)*(c1-c2));
gamma(2)=exp(-log(x(2)+Lam(2)*x(1))...
    -x(1)*(c1-c2));
gamma=gamma';

```

Wyniki 4.2



Rys. 4.34. Równowaga cieczeniowa dla układu metanol-woda

4.4. Wizualizacja obiektów 3D

MATLAB oferuje bogatą pulę możliwości prezentowania i wizualizacji danych i obiektów graficznych w przestrzeni trójwymiarowej. Mogą one pochodzić z obliczeń symulacyjnych i mieć formę wykresów przestrzennych jak na Rys. 4.30, mogą być mapami płaskimi bądź przestrzennymi, mogą mieć również postać obiektu złożonego z wielościanów stworzonych przez użytkownika.

Obiekty 3D można odpowiednio oświetlić, nadać im połysk i odpowiednio ustawić wirtualną kamerę, która „fotografuje” obiekt, dając obraz widziany w oknie graficznym. Niektóre z tych pojęć użyto już wcześniej w przykładach sekcji 4.3. Poniżej przedstawiono wybrane najważniejsze funkcje używane w wizualizacji 3D.

Tablica 4.6. Funkcje pomocne przy tworzeniu obiektów przestrzennych

<code>line(x,y,z)</code>	rysuje prostą łączącą kolejne punkty zawarte w wektorach x, y, z
<code>fill3(x,y,z,c)</code>	rysuje wielokąty o wierzchołkach zawartych w wektorach x, y, y i kolorze o specyfikacji zawartej w wektorze c.
<code>patch(x,y,z,c)</code>	rysuje wielokąty o wierzchołkach zawartych w wektorach x, y, y i kolorze o specyfikacji zawartej w wektorze c.
<code>patch('prop',val...)</code>	rysuje powierzchnię w oparciu o własności, takie jak: 'vertices' – macierz współrzędnych x,y,z wszystkich wierzchołków, 'faces' – macierz numerów wierzchołków tworzących ściany, 'FaceColor' – macierz kolorów ścian itd.
<code>surface(x,y,z,c)</code>	rysuje powierzchnię o węzłach określonych wektorami x, y i z i kolorze o specyfikacji zawartej wektorze c.
<code>[x,y,z]=cylinder(r)</code>	tworzy macierz współrzędnych kartezjańskich walca o promieniu r, jeśli r jest wektorem to walec będzie miał zmienny promień zgodny z zawartością wektora r.
<code>[x,y,z]=sphere(n)</code>	tworzy macierz współrzędnych kartezjańskich kuli o promieniu 1 ze środkiem o współrzędnych [0,0,0] i n równoleżnikach oraz n południkach.
<code>[x,y,z] = ellipsoid (xc,yc,zc,xr,yr,zr,n)</code>	tworzy macierz współrzędnych kartezjańskich elipsoidy obrotowej o długościach półosi [xr,yr,zr] ze środkiem o współrzędnych [xc,yc,zc] i n równoleżnikach oraz n południkach.
<code>rotate(h,oś,alfa)</code>	obróć obiekt o uchwycie h, wokół osi określonej wektorem oś (np. [1 0 0] to oś x), o kąt alfa (stopni).
<code>view(az,el)</code>	określa dwa kąty: azymut (Az) i wysokość (El), pod którymi widzimy obiekt. Aby wybrać i ustalić najlepsze położenie obiektu można obracać obiekt w oknie Figure opcją Rotate 3D. Az i El wyświetlają się wtedy w lewym dolnym narożniku ekranu.

Tablica 4.7. Funkcje sterujące powierzchnią obiektów

<code>material opcja</code>	Określa własności powierzchni materiału wszystkich obiektów <code>surface</code> lub <code>patch</code> w danym oknie. Do wyboru są opcje: <code>shiny</code> , <code>dull</code> i <code>metal</code> .
<code>material([ka,kd,ks])</code>	jw. gdzie <code>ka</code> – siła światła zewnętrznego, <code>kd</code> – rozproszonego, <code>ks</code> – odbitego ($0 \leq k \leq 1$).
<code>alpha(k)</code>	ustawia przezroczystość wszystkich obiektów w danym oknie, gdzie $0 \leq k \leq 1$. $k = 1$ oznacza nieprzezroczyste ściany, $k = 0$ – przezroczyste ściany. jw., ale dotyczy tylko obiektu o uchwycie <code>h</code> .
<code>alpha(h,k)</code> <code>shading opcja</code>	określa sposób cieniowania powierzchni złożonej z fasetek (okienek siatki przestrzennej). Do wyboru <code>flat</code> – koloruje fasetki i linie tym samym kolorem, <code>faceted</code> – koloruje fasetki wg mapy kolorów, ale siatka pozostaje czarna, <code>interp</code> – wygładza kolory między fasetkami.
<code>colormap opcja</code>	ustawia paletę barw wszystkich obiektów 3D w oknie. Dostępnych jest 13 wbudowanych palet: <code>jet</code> , <code>hot</code> , <code>cool</code> , <code>copper</code> , <code>spring</code> , <code>summer</code> , <code>autumn</code> , <code>winter</code> itd.

Kolejne funkcje służą do ustawiania fotoaparatu (kamery), który można utożsamiać z okiem obserwatora sceny widocznej na ekranie. Funkcje tych będziemy głównie używać w animacjach opisanych w sekcji 4.5.

Tablica 4.8. Funkcje do manipulacji fotoaparatem

<code>camdolly(dx,dy,dz)</code>	odpowiada przesunięciu nieruchomej kamery na wózku o <code>dx</code> , <code>dy</code> i <code>dz</code> od poprzedniego położenia.
<code>camorbit(dth,dfi)</code>	obraca kamerę o kąt <code>dth</code> w kierunku poziomym i <code>dfi</code> w kierunku pionowym, jej oś widzenia pozostaje skierowana na cel.
<code>campos([x y z])</code> <code>campan(dth,dfi)</code>	współrzędne punktu, w którym znajduje się kamera obraca oś optyczną kamery o kąty <code>dth</code> i <code>dfi</code> (zob. <code>camorbit</code>).
<code>camroll(dth)</code> <code>camtarget([x y z])</code>	obraca kamerę o kąt <code>dth</code> wokół jej osi pionowej współrzędne punktu, w który wycelowana jest oś obiektywu kamery.
<code>camzoom(k)</code>	zmienia ogniskową kamery o współczynnik <code>k</code> , dla $k > 1$ obraz rośnie, dla $k < 1$ obraz maleje.

Uwaga: funkcje kamery (fotoaparatu) są dostępne interaktywnie w pozycji menu okna **Figure** o nazwie View/Camera Toolbar.

Tablica 4.9. Funkcje do manipulacji oświetleniem

<code>light('opcja',wartość,...)</code>	definiuje źródło światła przez wybór opcji. Ważne opcje: 'Position' – [x,y,z] współrzędne źródła światła, 'FaceLighting' – sposób oświetlenia (zob. lighting), 'AmbientStrength' – siła źródła (0-1), 'Style' – 'local' światło punktowe, 'infinite' – światło oddalone.
<code>lighting</code>	włącza oświetlenie obiektów wg jednej z opcji: none – brak, flat – płaskie, stosowane dla obiektów o płaskich ścianach, gouraud i phong – stosowane do oświetlania obiektów krzywoliniowych.
<code>camlight('opcja')</code>	ustawia źródło światła związane z kamerą. Opcje 'headlight' – światło na kamerze, 'right' – z prawej, 'left' – z lewej.
<code>camlight(az,el)</code>	jw., ale źródło światła ma współrzędne az, el (zob. view – Tabl. 4.7).

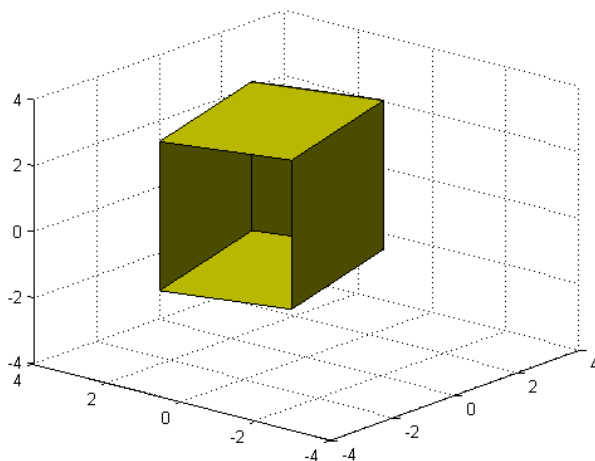
Przykład 4.8. Ilustracja tworzenia i oświetlania obiektu typu patch w grafice 3D

```
%Przykład 4.8 Ilustracja obiektu typu patch w 3D
clear all; close all; clc
[a b]=pudlo(1,1,0,3,2,3);
a=a*1.5;
h=patch('Vertices',a,'Faces',b,'FaceColor','y');
axis ([-4 4 -4 4 -4 4])
grid on
%Obróć obiekt
zdir = [0 0 1]; %wybór osi z jako osi obrotu
center = [0 0 0]; %środek obiektu
angle=15; %kąt obrotu [deg]
rotate(h,zdir,angle,center)
%opcje grafiki
light('Position',[0 1 1],'Style','infinite');
view(-50,24)
material shiny
lighting flat
```

Wierzchołki i ściany bryły zawarte są w pliku **pudlo.m**

```
function [verts,faces]=pudlo(x,y,z,a,b,c)
%x,y,z - położenie centrum
%a,b,c - rozmiary pudła
ap=a/2; bp=b/2; cp=c/2;
verts=[ x+ap y+bp z+cp; x+ap y+bp z-cp; x+ap y-bp z-cp;
        x+ap y-bp z+cp; x-ap y+bp z+cp; x-ap y+bp z-cp; x-ap y-bp z-cp;
        x-ap y-bp z+cp; x-ap y-bp z+cp; x+ap y-bp z+cp;
        x+ap y-bp z-cp; x-ap y-bp z-cp; x-ap y+bp z-cp;
        x+ap y+bp z-cp; x+ap y-bp z-cp; x-ap y-bp z-cp;
        x-ap y-bp z+cp; x+ap y-bp z+cp; x+ap y+bp z-cp;
        x-ap y+bp z-cp; x-ap y+bp z+cp; x+ap y+bp z+cp];
faces=[ 1 2 3 4
        %5 6 7 8 %Ta ściana jest usunięta
        9 10 11 12
        13 14 15 16
        17 18 19 20
        21 22 23 24];
```

Rezultat 4.8. Ilustracja tworzenia obiektu typu patch

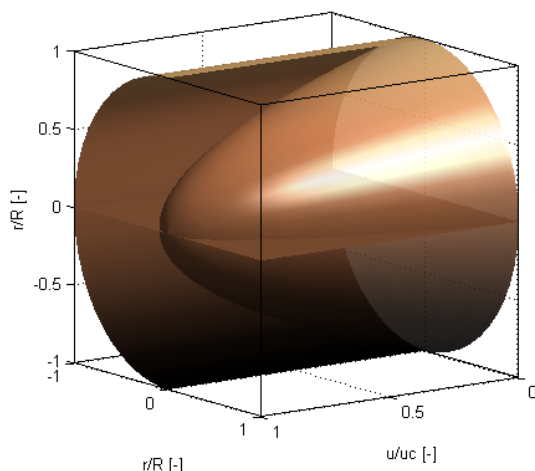


Rys. 4.35. Rysunek 3D uzyskany za pomocą polecenia **patch** i oświetlony

Przykład 4.9. Grafika 3D z wykorzystaniem obiektów typu surf i patch

```
%Przykład 4.9 Profil prędkości przepływu w rurze
clear all, close all; clc
n=50; umax=0.5; R=1;
[x y z]=cylinder(1,n);
%usunięcie połowy cylindra
for i=1:n
    if y(i)>0
        x(i)=NaN;
    end
end
%rysunek połowy cylindra
h1=surfc(x,y,z);
u=0:0.01:umax;
%wykres paraboli
[x1 y1 z1]=cylinder(R*(1-u/umax).^0.5,50);
hold on
h2=surf(x1,y1,z1);
%rotacja wykresu
zdir = [0 1 0]; %wybór osi z jako osi obrotu
center = [0 0 0]; %środek obiektu
angle=90; %kąt obrotu [deg]
rotate(h1,zdir,angle,center)
rotate(h2,zdir,angle,center)
view(144,12)
%płaszczyzna symetrii
xp=[1 1 0 0];
yp=[-1 1 1 -1];
zp=[0 0 0 0];
patch(xp,yp,zp)
%opis
xlabel('u/uc [-]');
ylabel('r/R [-]');
zlabel('r/R [-]');
box on
%ustawienia graficzne 3D
colormap copper
shading interp
alpha(h2,0.9);
camzoom(0.95)
light('Position',[0 1 1],'Style','infinite');
material shiny
lighting gouraud
```

Rezultat 4.9



Rys. 4.36. Grafika 3D utworzona z obiektów typu surf i patch

4.5. Animacja

Animację można wykorzystać do sugestywnego przekazywania wyników symulacji odbiorcom, jest więc zalecana w prezentacjach multimedialnych. MATLAB umożliwia animację zarówno wykresów dwu- jak i trójwymiarowych. Już za pomocą kilku prostych metod można ożywić statyczne wykresy. Pokażemy tu kilka przykładów.

Najprostszy wykres animowany tworzy funkcja `comet` użyta zamiast `plot` na wykresach 2D. `Comet` rysuje wykres poruszającym się znacznikiem (kometą) ciągnącą za sobą ogon. Widać wtedy w jakiej kolejności powstawał wykres. Po wykonaniu wykresu pozostaje on statyczny. Wystarczy napisać te linijki

```
x=linspace(0,2*pi,1000);  
comet(x,sin(x));
```

aby na ekranie zobaczyć powstający na oczach widza rysunek funkcji sinus.

Jednakże typowym, opartym o zasadę kina, sposobem animacji jest rysowanie kolejnych obrazków (klatek), i pokazywanie ich jedna za drugą z odpowiednią szybkością, tak aby wywołać wrażenie ruchu.

Do tego celu służy funkcja `drawnow`. Powoduje ona, że poprzedni wykres (bez osi, siatki itd.) zostanie wymazany, a na jego miejscu umieszczony następny. Ilustrują to kolejne przykłady. Rezultaty należy obejrzeć w czasie rzeczywistym, nie są tu więc pokazywane. Uwaga: w przypadku konieczności wcześniejszego przerwania pętli rysującej wykres można to zrobić za pomocą klawiszy CTRL+C.

Przykład 4.10. Animacja krzywej 2D – dane z pliku

W projekcie 8.1 (zob. rozdz. 8) symulowano przemieszczanie się frontu desorpcji w złożu materiału ziarnistego, przez które przepuszczane jest powietrze, i wygenerowano plik z danymi **Wyniki_desorp.mat**. Zawiera on w formie macierzy strukturalnej trzy pola: `ax` – wektor współrzędnych rysunku, `h` – wektor kolejnych wysokości złoża licząc od wlotu gazu i `allX` – macierz zawartości składnika na danej wysokości złoża w danym czasie. O sposobie zapisu zmiennych w plikach typu `mat` można przeczytać w Rozdz. 5. Pole `allX` zostało przepisane do macierzy `X`.

Rysując kolejne wiersze macierzy `X` sekwencyjnie, uzyskamy efekt animacji. Wykonuje to pętla `for`. W pętli tej do odnawiania wykresu użyto funkcji `drawnow` i uchwytu funkcji `plot`. Nadając poprzez funkcję `set` jej własności `'YData'` wartości kolejnych wierszy macierzy `X` uzyskujemy animację poklatkową. Do regulacji szybkości wyświetlania klatek służy funkcja `pause`. Im mniejszy jej argument tym szybciej tworzony jest wykres.

```
%Przykład 4.10
%Animacja fali chromatograficznej desorpcji
clear all; close all; clc;
%Załaduj i odczytaj dane z pliku
%wytworzonego w Projekcie 8.1
Z=load('Wyniki_desorp');
ax=Z.ax; X=Z.allX; H=Z.h;
[w,k]=size(X);
h=plot(H,X(1,:), 'LineWidth',2);
axis(ax); grid on
xlabel('H [m]'); ylabel('X [kg/kg]');
title('Animacja krzywej desorpcji');
for i=2:w
    set(h,'YData',X(i,:))
    drawnow
    pause(0.01)
end
```

Rezultat 4.10

Aby zobaczyć wyniki, należy przepisać i uruchomić program.

W analogiczny sposób można animować krzywą generowaną bezpośrednio przez program jak w tym przykładzie.

Przykład 4.11. Animacja krzywej 2D

```
%Przykład 4.11 Animacja pojedynczej krzywej
clear all; close all; clc
x=-5:.02:5;
y=sin(x);
h=plot(x,y);
axis([-5 0.02 0 0.13])
for t=2:.01:700
    drawnow
    y=sin(x-2*t).^2.*exp(-t);
    set(h,'YData',y)
    pause(0.05)
end
```

Rezultat 4.11

Aby zobaczyć wyniki, należy przepisać i uruchomić program.

Ten sam sposób może posłużyć do animacji funkcji na wykresie 3D. Ilustruje to przykład 4.11.

Przykład 4.12. Animacja powierzchni 3D

```
%Przykład 4.12 Powierzchnia animowana
clear all; close all; clc
Z = peaks;          % dane z funkcji wbudowanej
hSurf = surf(Z); % wykres powierzchniowy
axis tight;         % dopasowanie skali i jej unieruchomienie

numer = text(10,40,5,'0','FontSize',14,...
            'EdgeColor',[0 0 0]); % wyświetla numer iteracji na
            wykresie
for j = 1:1000
```

```

% animacja danych
set(hSurf, 'ZData', sin(2*pi*j/20)*exp(-j/300)*Z)
% zmiana numeru iteracji na wykresie
set(numer, 'String', num2str(j))
% częstość wyświetlania klatek
pause(0.05)
end

```

Uwaga: szereg efektów animacji można uzyskać obracając obiekt (np. funkcją `rotate`), przesuwając kamerę lub źródło światła względem obiektu. Opcje te częściowo opisano w poprzedniej sekcji.

Aby wytworzone animacje stały się dostępne poza środowiskiem MATLABa, należy je zapisać w formacie dostępnym w przeglądarkach plików np. w formacie avi. Jeden ze sposobów zapisu sekwencji animacji do pliku pokazuje Przykład 4.12. Kolejne generowane klatki są zapisywane w macierzy `F` za pomocą funkcji `getframe`, a następnie zapisane w pliku za pomocą `movie2avi`. Uwaga: wygenerowany plik otwiera się bez problemu w systemach 32-bitowych. W 64-bitowym Windows 7 należy zainstalować odpowiedni codec umożliwiający odczyt pliku. Wybór kompresji pliku odbywa się przez podanie w `movie2avi` następującej sekwencji `'Compression', <nazwa codeca>`. Można też wpisać `'Compression', 'none'`, a następnie dokonać kompresji programem zewnętrznym. Pozostawiamy to dociekliwym Czytelnikom.

Przykład 4.13. Zapisywanie animacji w formacie przenośnym avi

```

%Przykład 4.13
%Zapis animacji do pliku avi
clear all; close all; clc
u=linspace(0,2*pi,50);
v=linspace(-2,2,50);
[U,V]=meshgrid(u,v);
h1=surf(cos(U),sin(U),V);
hold on
h2=surf(0.5*cos(U),V,0.5*sin(U));
axis equal; shading interp; camlight('right')
material shiny; colormap bone
del=90/500;
for i=1:40
rotate(h1,[1 0 0],i*del);
drawnow
pause(0.05)

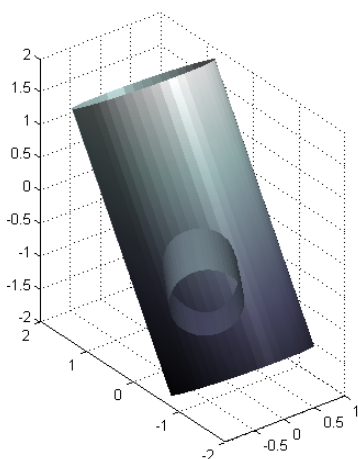
```

```

F(i)=getframe(gca);
end
for i=1:40
rotate(h2,[0 0 1],i*del);
drawnow
pause(0.05)
F(40+i)=getframe(gca);
end
movie2avi(F,'rotacje.avi');

```

Rezultat 4.12



Rys. 4.37. Ostatnia klatka filmu wygenerowanego przez program

LITERATURA

Pakowski Z., 1986, Computer drawing of enthalpy-humidity charts, Hungarian Journal of Industrial Chemistry, Veszprem, vol.14, pp. 225-236.

Pakowski Z. Głębowski M., „Symulacja procesów inżynierii chemicznej”, Wyd. PŁ, Łódź 2001.

5. ANALIZA DANYCH DOŚWIADCZALNYCH

5.1. Zapisywanie i wczytywanie danych

Podstawowe informacje o operacjach na katalogach i plikach zawarto w rozdz. 2.5. Ponieważ w obróbce danych import i eksport danych pomiędzy środowiskiem MATLABa a innymi programami jest szczególnie istotny przedstawiono tu dodatkowo kilka ważnych metod służących do tego celu wraz z przykładami.

5.1.1. Eksport danych

- Polecenie `save`

Do celów wewnętrznych MATLABa formatem plików stosowanym do importu i eksportu danych jest format **.mat**. Pliki tego typu służą do zapisu i odczytu wszystkich bądź wybranych zmiennych z pamięci roboczej. Polecenie to umożliwia również zapis zmiennych w formacie tekstowym. Sterowanie działaniem polecenia odbywa się przez wybór odpowiedniej opcji, co ilustruje przykład 5.1.

Przykład 5.1. Eksport danych w formacie .mat i .txt

p5_1.m

%Przykład 5.1 Eksport danych

```
xx=linspace(1,10,100);  
yy=xx.^3+2.*xx-6;  
save('DG_wszystko');  
save('DG_tylko_xx','xx');  
save('DG_ascii_h.txt','xx','-ascii');  
xxt=xx';  
save('DG_ascii_v.txt','xxt','-ascii');  
A=[xx' yy'];  
save('DG_tablica.txt','A','-ascii');
```

Rezultat 5.1

Program generuje pięć plików z danymi. Kolejne pliki zawierają:

- **DG_wszystko.mat** – zawiera wszystkie zmienne `xx`, `yy`, `A`, `xxt`.
- **DG_tylko_xx.mat** – zawiera jedynie zmienną `xx`.
- **DG_ascii_h.txt** – zawiera wektor `xx` w formacie poziomym
- **DG_ascii_v.txt** – zawiera wektor `xx` w formacie pionowym
- **DG_tablica.txt** – zawiera tablicę, w której kolumna pierwsza zawiera wektor `xx`, a druga wektor `yy`.

W nazwach zmiennych można stosować maskę *, np. aby zapisać wszystkie zmienne, których nazwa zaczyna się na literę w, należy podać nazwę 'w*'. Utworzone przez **save** pliki w formacie ASCII (tekstowe) dają się odczytać dowolnym edytorem tekstów. Należy pamiętać, aby rozszerzenie **.txt** umieścić w nazwie pliku. Pliki z rozszerzeniem **.mat** mogą być przeczytane wyłącznie przez MATLAB. Do istniejącego pliku **.mat** można dopisać nowe zmienne, wstawiając jako opcję **'-append'**. Np. polecenie

```
save('DG_tylko_xx','yy','-append');
```

spowoduje dopisanie zmiennej yy do istniejącego pliku. Polecenie **save** nie zapisuje macierzy komórkowych.

- Polecenie **dlmwrite**

Zapisywanie danych poleceniem **save** w formacie ASCII używa dwu spacji jako separatora elementów macierzy. Aby zapisać dane w formacie tekstowym z innym separatorem, należy użyć polecenia **dlmwrite('nazwa pliku', zmienna, 'opcja 1', parametrl,...)**.

Jako opcji można używać:

delimiter	znak rozdzielający elementy macierzy. Jako parametr umieszcza się typowe znaki jak ' ' (spacja), ';' (średnik), '-' (myślnik). Domyślny znak to ',' (przecinek). Można też użyć znaków specjalnych jak '\t' (tabulator).
roffset	pozostaw puste wiersze od góry pliku, jako parametr wstawia się liczbę wierszy.
coffset	pozostaw puste kolumny od lewej, parametr to liczba pustych kolumn.
newline	oznaczenie sposobu tworzenia nowej linii po każdym wierszu macierzy, dla komputerów typu PC używa się parametru 'pc'.
precision	format liczbowy, parametr to liczba miejsc znaczących lub łańcuch formatujący (zob. objaśnienie poniżej).

Łańcuch formatujący to ciąg znaków zaczynający się od znaku % określającego format zapisu lub wydruku zmiennej. Najbardziej przydatne są następujące formaty:

- format całkowity %d lub %i, np. '%10d', 12 zapisuje liczbę 12 poprzedzoną ośmioma spacjami, '%d', 3.14 daje liczbę 3.

- format zmiennoprzecinkowy %f np. '%8.5f' oznacza format o łącznej liczbie znaków 8 (wliczając kropkę dziesiętną) i 5 znakach po przecinku, i tak '%8.5',12.41 drukuje 12.41000, a '%8.1f',12.41 zapisuje 12.4 poprzedzone pięcioma spacjami.
- format zmiennoprzecinkowy %g – skrócona forma formatu %f bez poprzedzających zer np. '%0.3g',pi daje 3.14 (%0.3 oznacza liczbę cyfr znaczących równą 3).
- format wykładniczy %e lub %E np. '%8.3e',pi drukuje 3.142e+000. Symbol e lub E oznacza 10 podniesione do odpowiedniej potęgi.

Uwaga łańcuchy formatujące mogą zawierać dodatkowy tekst np. 'Temperatura [C] %5.2f', 23.1 daje Temperatura [C] 23.10.

Przykład 5.2. Zapis danych poleceniem dlmwrite

p5_2.m

%Przykład 5.2 Eksport danych - dlmwrite

```
clear all; clc;
a=linspace(0.1,10,100); b=log(a);
A=[a' b'];
dlmwrite('DL_tablica1.txt',A);
dlmwrite('DL_tablica2.txt',A,'newline','pc');
dlmwrite('DL_tablica3.txt',A,'delimiter',';', 'newline','pc');
dlmwrite('DL_tablica4.txt',A,'delimiter','
','newline','pc','coffset',3);
dlmwrite('DL_tablica5.txt',A,'delimiter','
','newline','pc','precision',8);
dlmwrite('DL_tablica6.txt',A,'delimiter','\t','newline','pc',
'precision','%8.5f');
```

Rezultat 5.2

Program tworzy sześć plików z tablicą A w różnych formatach. Jedynie plik **DL_tablica6.txt** zawiera odpowiednio sformatowane dane w dwu kolumnach oddzielonych tabulatorem. Takie dane są łatwe do przeniesienia do innych programów.

W MATLABie istnieją jednak narzędzia pozwalające eksportować dane w specjalnym formacie innych programów.

- Polecenie **xlswrite**

Pozwala na zapis danych w formacie programu MS Excel®. Należy użyć polecenia **xlswrite('nazwa pliku', 'zmienna', 'nazwa arkusza', 'zakres')**. Nazwa pliku musi posiadać rozszerzenie **.xls**. W komputerze musi być zainstalowany

Excel. Wynik działania polecenia można podstawić pod dwie zmienne określające status operacji i komunikat informacyjny. Status operacji równy 1 oznacza udaną operację.

Przykład 5.3. Zapis danych poleceniem xlswrite

p5_3.m

%Przykład 5.3 Eksport danych - xlswrite

```
clear all; clc;
t=linspace(10,100,90);
E=4270; % [kJ/kmol]
t0=272.15; % [C]
k0=1.5e-5; % [1/s]
R=8.314; % [kJ/(kmol K)]
k=k0*exp(-E./(R*(t(:)+t0)));

A=[t' k];
xlswrite('DX_tablica1.xls',A);
xlswrite('DX_tablica2.xls',A,'Dane_kin','B5:C25');
[stat
msg]=xlswrite('DX_tablica2.xls',{'Temp,C','k,1/s'},'Dane_kin','B
4:C4');
disp(stat);
disp(msg);
```

Rezultat 5.3

W wyniku programu zostały utworzone dwa pliki. W pierwszym wpisano dwie kolumny macierzy A do komórek zaczynających się od A1 w arkusz o nazwie 'Dane_kin' do pliku **DX_tablica1.xls**.

W drugim początkowo wpisano tę macierz do zakresu komórek B5:C25, a więc jedynie 20 wierszy macierzy A. W kolejnej operacji dopisano do tego samego arkusza wiersz z nagłówkami.

Uwaga: wpisując łańcuchy jako elementy macierzy, posłużono się macierzą upakowaną. W przeciwnym razie MATLAB wstawia poszczególne znaki łańcuchów do osobnych komórek. Komunikaty wydrukowane podczas wykonywania operacji mają postać:

```
Warning: Added specified worksheet.
> In xlswrite>activate_sheet at 294
In xlswrite>ExecuteWrite at 249
In xlswrite at 207
In data_gen3 at 12
1
```

```
message: ''  
identifier: ''
```

- Polecenia do otwierania, odczytywania, zapisywania i zamykania plików

Znanym z innych języków programowania sposobem transferu danych do i z pliku dyskowego jest następująca sekwencja: otwarcie pliku do odczytu/zapisu, odczytywanie/zapisywanie danych w pliku w miarę potrzeb oraz zamknięcie pliku po użyciu. Można wtedy odczytywać/zapisywać dane po wierszu, przeszukiwać plik, dopisywać dane do istniejącego pliku, korygować je itd. Potrzebne są do tego następujące instrukcje (podano tylko najważniejsze opcje z list argumentów):

`fopen('nazwa', 'zezwozenie')`

otwiera plik w celu wykonania na nim operacji w zależności od opcji 'zezwozenie'. Lista opcji jest następująca:

'r' otwarcie pliku do odczytu

'w' usunięcie zawartości istniejącego pliku lub utworzenie nowego i otwarcie go do zapisu

'a' otwarcie pliku w celu dopisania elementów na jego końcu

'r+' otwarcie pliku do odczytu i zapisu

'w+' usunięcie zawartości istniejącego pliku lub utworzenie nowego i otwarcie go do odczytu i zapisu

'a+' otwarcie pliku w celu dopisania elementów na jego końcu i odczytu

Wartość funkcji można podstawić pod zmienną `id`, która będzie stanowić identyfikator (numer) pliku w dalszych odwołaniach.

`fseek(id,offset,początek)`

ustawienie znacznika w pozycji `offset` od początku. Typowym oznaczeniem początku jest 'bof' – begining of file

`fread(id,rozmiar)`

wczytuje dane o podanym rozmiarze z otwartego pliku binarnego. Rozmiar może być skalarem lub mieć postać tablicy `[m,n]`

`fwrite(id, A, precyzja)`

zapisuje do otwartego pliku binarnego zawartość macierzy `A`. Precyzja to łańcuch określający format zapisu (liczby całkowite, rzeczywiste, znaki). Uwaga: elementy macierzy są zapisywane jako wektor, w porządku kolumnowym – najpierw pierwsza kolumna, potem druga itd. Podczas odczytywania elementy tego wektora są układane w kolumny zadanej struktury, a jeśli nie jest określona, podawane są jako wektor.

`fprintf(id,format,A)`

zapisuje dane w pliku tekstowym, konwertuje dane przechowywane w części rzeczywistej macierzy A kolumnowo na łańcuchy znakowe i zapisuje je w pliku tekstowym o podanym identyfikatorze, opcja format określa rodzaj konwersji, szerokość pola i liczbę cyfr znaczących każdej wymienionej w funkcji macierzy.

Rodzaj konwersji:

- `%d` zapis liczb całkowitych
- `%f` liczby rzeczywiste w formacie stałoprzecinkowym
- `%e` liczby rzeczywiste w formacie zmiennoprzecinkowym
- `%g` automatyczny dobór krótszego formatu `%e` lub `%f`
- `%c` zapis pojedynczego znaku
- `%s` zapis łańcucha znakowego.

Znak specjalny:

- `\b` cofnięcie o jeden znak
- `\f` nowa strona
- `\n` nowy wiersz
- `\r` powrót karetki (przesunięcie kursora do początku wiersza)
- `\t` tabulacja
- `\“` znak apostrofu
- `\\` znak lewego ukośnika
- `%%` znak procentu

`fclose(id)`

zamyka otwarty plik o numerze id.

`ferror(id)`

zwraca komunikat błędu dla ostatniej wykonywanej operacji na pliku o numerze id.

Przykład 5.4. Pisanie do i czytanie z pliku binarnego

 **p5_4.m**

```
%Przykład 5.4 Zapis/odczyt pliku binarnego
clear all; clc;
%Utworzenie nowego pliku i zapisanie w nim
%macierzy liczb losowych 2x5
O=rand(2,5)
id=fopen('ones.bin','w');
fwrite(id,O,'double');
fclose(id);
```

```

%otwarcie istniejącego pliku i odczytanie
%z niego macierzy 2x3
id1=fopen('ones.bin','r');
A=fread(id1,[2,3],'double')
fclose(id1);

%otwarcie istniejącego pliku i dopisanie
%do niego macierzy A z przeciwnym znakiem
id2=fopen('ones.bin','a');
fwrite(id2,-A,'double');
fclose(id2);

id3=fopen('ones.bin','r');
%Odczytanie z pliku macierzy 4x5
B=fread(id3,[4,5],'double')
%ustawienie znacznika odczytu na początku pliku
%i odczytanie całej jego zawartości w jednej kolumnie
fseek(id3,0,'bof');
C=fread(id3,'double')
fclose(id3);

```

Rezultat 5.4

```

O =
    0.9106    0.2638    0.1361    0.5797    0.1450
    0.1818    0.1455    0.8693    0.5499    0.8530

A =
    0.9106    0.2638    0.1361
    0.1818    0.1455    0.8693

B =
    0.9106    0.1361    0.1450   -0.2638
    0.1818    0.8693    0.8530   -0.1455
    0.2638    0.5797   -0.9106   -0.1361
    0.1455    0.5499   -0.1818   -0.8693

C =
    0.9106
    0.1818
    0.2638
    0.1455
    0.1361

```



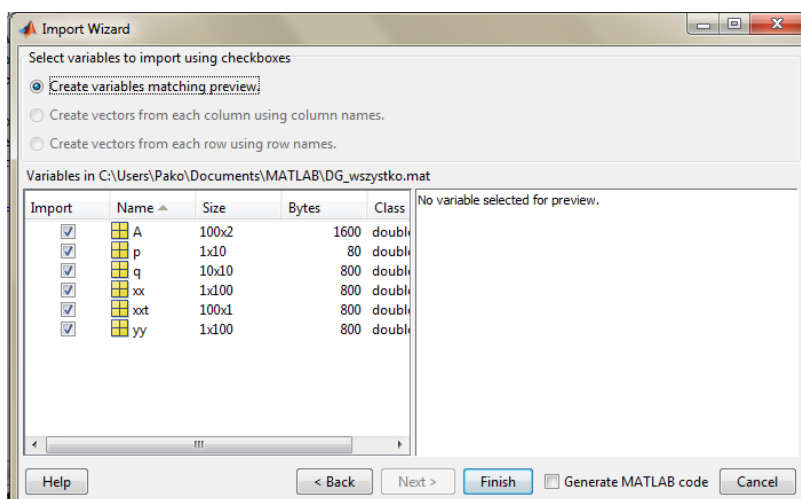
```

0.8693
0.5797
0.5499
0.1450
0.8530
-0.9106
-0.1818
-0.2638
-0.1455
-0.1361
-0.8693

```

5.1.2. Import danych

- Import Wizard



Rys. 5.1. Okno Import Wizard

MATLAB posiada uniwersalne narzędzie do wczytywania danych o nazwie **Import Wizard**. Do wczytywania danych przystępujemy, wciskając trzecią z lewej ikonę (Import data) w pasku narzędziowym okna **Workspace**. Pojawia się wtedy okno **Import Data**, w którym dokonujemy wyboru pliku z danymi. Po wybraniu pliku pojawia się okno **Import Wizard** w postaci przedstawionej na Rys. 5.1. W oknie tym można zbadać zawartość pliku przed jego wczytaniem oraz, w przypadku plików typu **.mat**, dokonać wyboru zmiennych do wczytania. **Import Wizard** obsługuje wszystkie typy plików danych dostępne w MATLABie.

- Polecenie `load`

Polecenie `load` obsługuje pliki typu **.mat**. Dwa podstawowe typy wywołania to:

```
s=load(<nazwa pliku>)
load(<nazwa pliku>)
```

Pierwsza ładuje wyniki wczytywania do macierzy strukturalnej `s`. Druga wczytuje dane bez ich podstawiania do macierzy. Dodatkowo po nazwie pliku można umieścić listę zmiennych do załadowania – w przeciwnym razie polecenie ładuje całą zawartość pliku.

Przed wczytaniem można sprawdzić zawartość pliku **.mat** za pomocą instrukcji `whos -file <nazwa pliku>`. Użycie `load` ilustruje przykład poniżej.

Przykład 5.5. Wczytanie danych poleceniem load

 **p5_5.m**

```
%Przykład 5.5 Import danych - load
clear all; clc;
whos -file 'DG_wszystko.mat';
load('DG_wszystko.mat');
disp(p(1:5));
clear all;
load('DG_wszystko.mat','xx');
disp(xx(1:5));
s=load('DG_wszystko.mat');
disp(s(1).xx(1:5));
```

Rezultat 5.5

Name	Size	Bytes	Class	Attributes
A	100x2	1600	double	
p	1x10	80	double	
q	10x10	800	double	
xx	1x100	800	double	
xxt	100x1	800	double	
yy	1x100	800	double	
0.7060	0.0318	0.2769	0.0462	0.0971
1.0000	1.0909	1.1818	1.2727	1.3636
1.0000	1.0909	1.1818	1.2727	1.3636

Wyniki te zawierają:

- zawartość pliku **DG_wszystko.mat** uzyskane poleceniem `whos -file`,
- pięć pierwszych elementów zmiennej `p` wczytanej poleceniem `load`,
- pięć pierwszych elementów zmiennej `xx` wczytanej poleceniem `load` z podaną nazwą zmiennej,
- pięć pierwszych elementów zmiennej `xx` odczytanej z macierzy strukturalnej.

- Polecenie `importdata`

Polecenie to jest podstawowym sposobem odczytywania plików tekstowych zawierających znaki kodu ASCII. Pozwala czytać pliki gdzie kolumny oddzielane są różnymi znakami oraz posiadają nagłówki. Typowe polecenie ma postać

```
Z=importdata('nazwa','separator',liczba_wierszy_naglowka)
```

gdzie pierwszy argument to nazwa pliku z rozszerzeniem, drugi to separator kolumn danych w pliku np. ' ' – spacje, ',' – przecinek, '\t' – tabulator, trzeci to liczba wierszy nagłówka. Zmienna `Z` przyjmuje postać tablicy strukturalnej o jednym rekordzie, w którego polach znajdują się elementy wczytanej tablicy z danymi. Ilustruje to przykład 5.6.

Przykład 5.6. Wczytanie danych poleceniem `importdata`

p5_6.m

```
%Przykład 5.6 Czytanie danych z pliku tekstowego
clear all; clc
z=importdata('dane_GAB.txt',' ',2);
%wczytana macierz jest tablicą strukturalną
%o jednym rekordzie, w jego polach znajdują się elementy
% wczytanej tabelki
t=z.textdata(1)
n=z.colheaders
a=z.data
```

Rezultat 5.6

```
t =
    '      Dane sorpcji chitozanu'
n =
    'X[kg/kg]'    'aw[-]'    'T[C]'
a =
    0.0822    0.2340    25.0000
```

0.0932	0.3290	25.0000
0.1164	0.4430	25.0000
0.1213	0.5360	25.0000
0.1474	0.6540	25.0000
0.2263	0.7650	25.0000
0.3289	0.9040	25.0000
0.0806	0.2160	35.0000
0.0884	0.3180	35.0000
0.1133	0.4360	35.0000
0.1131	0.5150	35.0000
0.1412	0.6280	35.0000
0.1818	0.7430	35.0000
0.2638	0.8940	35.0000
0.0639	0.1970	45.0000
0.0856	0.3080	45.0000
0.1000	0.4300	45.0000
0.1029	0.4970	45.0000
0.1078	0.6010	45.0000
0.1703	0.7260	45.0000
0.2333	0.8850	45.0000

- Polecenie **dlmread**

Wczytuje i podstawia pod macierz dwuwymiarową dane z pliku tekstowego zapisanego z dowolnymi separatorami. Zawartość pliku tekstowego jest traktowana jako macierz dwuwymiarowa. Na przykład polecenie `A=dlmread('dane.txt','t','A1:C3')` odczytuje z pliku o nazwie **dane.txt**, gdzie elementy macierzy są oddzielane tabulatorami, zakres 3×3 (można używać zapisu Excela jak tu, lub wektorowego [1 1 3 3], przy czym dwie pierwsze liczby oznaczają współrzędne górnego-lewego narożnika, a dwie ostatnie współrzędne dolnego-prawego narożnika).

- Polecenie **xlsread**

Polecenie `A=xlsread('nazwa','arkusz','zakres')` odczytuje zakres danych np. B1:D20 z arkusza o podanej nazwie znajdującego się w pliku o podanej nazwie zapisanego w formacie MS Excel®.

- Polecenie **textscan**

Odczytuje dane z pliku tekstowego otwartego wcześniej poleceniem **fopen** i nadaniu mu numeru id. Przykładowe wywołanie to `C=textscan(id,'format')`

gdzie format to oznaczenie liczby o dowolnym formacie lub łańcucha, a C to macierz komórkowa, pod którą zostaną podstawione wyniki odczytywania. Szczegóły formatowania są dostępne w pomocy programu.

Przedstawiono tu najczęściej stosowane metody zapisywania i odczytywania danych z plików. W MATLABie istnieje wiele innych metod odczytywania i zapisu danych, w tym liczbowych, tekstowych, mieszanych, obrazów, filmów i nagrań audio. W przypadkach nieopisanych powyżej należy przestudiować pozycje Exporting data i Importing data w pomocy programu.

5.2. Obróbka wstępna danych

Wczytane dane należy wstępnie opracować przed poddaniem ich aproksymacji, wygładzaniu czy innej obróbce. Przede wszystkim dane należy posortować wg rosnącej wartości zmiennej niezależnej x . Sortowanie wierszy macierzy i wektorów wg rosnącej wartości zmiennej w wybranej kolumnie dokonuje polecenie `sortrows(n)`, gdzie n jest numerem kolumny, wg której sortujemy.

Posortowane dane należy przedstawić graficznie, aby zorientować się w trendzie i rozrzucie danych. Na tym etapie należy usunąć punkty drastycznie odbiegające od pozostałych (ang. *outliers*), jeśli ich źródłem są przypadkowe szumy, błędy pomiarowe lub typograficzne. Wystarczającym kryterium usunięcia punktu jest przypadek, gdy wartość zmiennej zależnej y odbiega od średniej o więcej jak 2 odchylenia standardowe. Technikę usuwania punktów odbiegających ilustruje poniższy przykład.

Przykład 5.7. Obróbka wstępna danych

p5_7.m

%Przykład 5.7 Obróbka wstępna danych

```
clear all; clc;
A=importdata('Dane_w1su.txt');
%wykres danych surowych
xx=A(:,1);
yy=A(:,2);
figure(1)
plot(xx,yy);
xlabel('x');
ylabel('y');
title('Dane nieposortowane');
```

```

%wykres danych posortowanych
A=sortrows(A,1);
xx=A(:,1);
yy=A(:,2);
figure(2)
plot(xx,yy,'-o');
xlabel('x');
ylabel('y');
title('Dane posortowane');

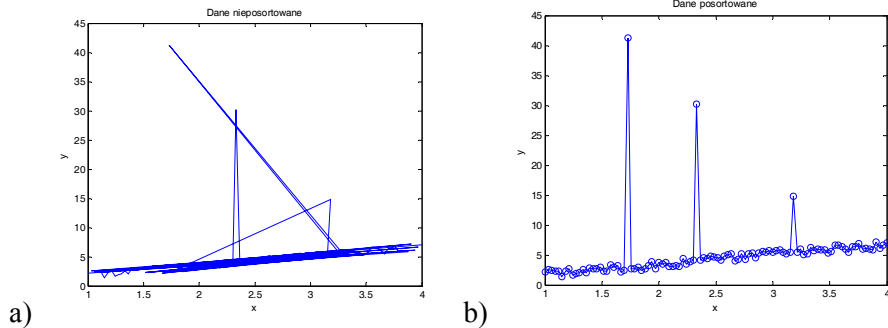
%dopasowanie linii prostej
lin=@(c,x) c(2)+c(1)*x;
c=polyfit(xx,yy,1);
figure(3)
plot(xx,yy,'o',xx,lin(c,xx));
xlabel('x');
ylabel('y');
title('Prosta dopasowana do wszystkich punktów');

%usunięcie punktów odbiegających
clear c;
myy=mean(yy);
syy=std(yy);
zp=abs(yy-myy)>2*syy; %zaznaczenie punktów odbiegających
A(zp,:)=[]; % usunięcie ich z macierzy danych
nyy=A(:,2);
nxx=A(:,1);
%ponowne dopasowanie linii prostej
c=polyfit(nxx,nyy,1);
figure(4)
plot(nxx,nyy,'o',nxx,lin(c,nxx));
xlabel('x');
ylabel('y');
title('Prosta po odrzuceniu punktów odbiegających');

```

Rezultat 5.7

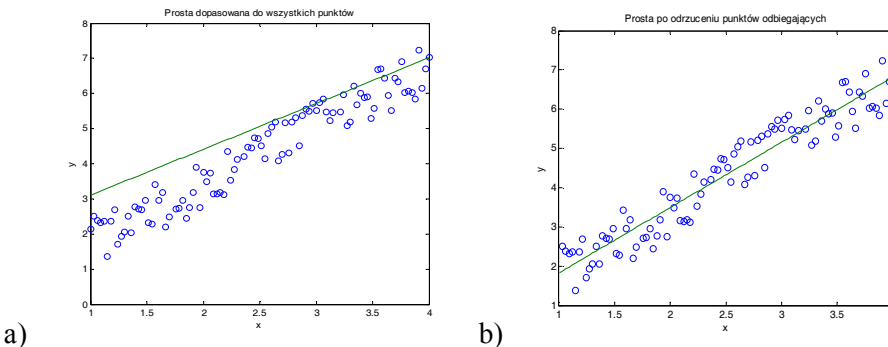
Dane surowe i posortowane wyglądają następująco:



Rys. 5.2. Wykres danych: a) surowych, b) posortowanych

Jak widać MATLAB łączy punkty w kolejności ich występowania w macierzy danych co zaciemnia wykres.

Dopasowaną prostą bez usuwania punktów odbiegających oraz po usunięciu punktów odbiegających pokazano poniżej:



Rys. 5.3. Dopasowana prosta: a) bez usuwania punktów odbiegających, b) po usunięciu punktów odbiegających

5.3. Aproksymacja

Aproksymacja polega na dopasowaniu do punktów doświadczalnych takiej funkcji ciągłej, która minimalizuje kryterium dopasowania. Należy więc do zagadnień optymalizacyjnych. Dopasowane równanie funkcji zastępuje zbiór punktów i pozwala np. na obliczenie pochodnej lokalnej, całki w danym przedziale lub przewidywanie wartości funkcji dla danego argumentu. Funkcje dobiera się empirycznie bądź na podstawie teorii opisującej proces, który był źródłem punktów doświadczalnych.

5.3.1. Metoda najmniejszych kwadratów

Typowym kryterium dopasowania jest minimum sumy kwadratów odchyłeń funkcji aproksymującej od punktów doświadczalnych, która dla funkcji jednej zmiennej jest dana zależnością:

$$SSE = \sum_i (y_{obl}(x_i) - y_i)^2, \quad (5.1)$$

gdzie x_i, y_i – punkty doświadczalne, $y_{obl}(x)$ funkcja dopasowana. Funkcja SSE jest funkcją celu dla narzędzia poszukującego minimum (optymalizator) w przestrzeni m -wymiarowej, gdzie m – liczba współczynników funkcji aproksymującej. Jakość dopasowania krzywej do punktów określa tzw. wskaźnik dopasowania R^2 dany wzorem:

$$R^2 = 1 - \frac{SSE}{\sum_i (y_i - y_m)^2}, \quad (5.2)$$

gdzie y_m to średnia arytmetyczna wszystkich wartości y_i . Wskaźnik dopasowania powinien być możliwie bliski 1. Wartości powyżej 0.8 są akceptowalne, a powyżej 0.9 oznaczają dobre dopasowanie.

Inną często stosowaną statystyką jest pierwiastek ze skorygowanego średniego błędu kwadratowego RMSE. Oblicza się go następująco:

$$RMSE = \sqrt{\frac{SSE}{n - N}}, \quad (5.3)$$

gdzie n jest liczbą punktów doświadczalnych, a N liczbą współczynników dopasowywanej krzywej.

Przy dobrym dopasowaniu wartość RMSE powinna być możliwie bliska zero.

Wśród spotykanych zagadnień aproksymacyjnych należy rozróżnić aproksymację funkcji jednej zmiennej od aproksymacji funkcji wielu zmiennych. Pierwsza z nich obejmuje aproksymację linią prostą, wielomianem, funkcją wykładniczą, funkcją logarytmiczną itd. Z uwagi na strukturę narzędzi MATLABA te ostatnie zalicza się do jednej klasy – aproksymacji dowolną funkcją.

Aproksymacja funkcji wielu zmiennych wymaga prezentacji wyników w przestrzeni 3D lub wyższych i wymaga specjalnych narzędzi.

Na zakończenie przestroga: znalezione minimum niekoniecznie jest minimum globalnym, a więc takim gdzie funkcja celu przyjmuje wartość najniższą z możliwych. Czasem możemy znaleźć minimum lokalne. Zależy to od doboru

punktu startowego i przeszukiwanego przedziału, a także sprawności optymalizatora.

5.3.2. Ocena statystyczna jakości parametrów funkcji aproksymującej

Z uwagi na statystyczny rozrzut punktów doświadczalnych wartości współczynników leżą w pewnym przedziale ufności, którego granice należy wyznaczyć. Ogólna metoda postępowania (dla funkcji jednej zmiennej) jest następująca (Cady i Trapp, 1999). Dla funkcji dopasowanej $f(x)$ o N współczynnikach będącej aproksymacją n punktów doświadczalnych x, y wykonujemy:

a) wyznaczenie wektora pochodnych funkcji aproksymacyjnej po każdym ze współczynników

$$D(x) = \begin{bmatrix} df/da_1 \\ df/da_2 \\ \dots \\ df/da_N \end{bmatrix} \quad (5.4)$$

gdzie f – funkcja aproksymacyjna, a – wektor jej współczynników,

b) obliczenie macierzy sum iloczynów tych pochodnych

$$p_{j,k} = \sum_i D_j(x_i) D_k(x_i) \quad (5.5)$$

dla $j, k = 1..N + 1$, $i = 1..n$

c) obliczenie skorygowanego odchylenia standardowego

$$s_{yx} = \sqrt{\frac{SSE}{n - N}} = RMSE, \quad (5.6)$$

d) obliczenie odchylen standardowych współczynników

$$(s_a)_j = s_{yx} \sqrt{(p^{-1})_{j,j}}, \quad (5.7)$$

e) wyznaczenie granic przedziału ufności współczynników

$$\Delta a_l = a_l \pm qt_{\alpha,d}(s_a)_l, \quad (5.8)$$

gdzie $l = 1..N$, $qt_{\alpha,d}$ – odwrotna kumulatywna funkcja t Studenta na poziomie istotności α i o liczbie stopni swobody $d = n - N$.

Przykładowo obliczenia takie są przeprowadzone w Projekcie 5.2.

5.3.3. Aproksymacja liniowa

Wykorzystując zasady algebry liniowej współczynniki prostej, uzyskuje się z rozwiązania układu równań liniowych

$$\begin{aligned} ax_1 + b &= y_1 \\ ax_2 + b &= y_2 \\ &\dots\dots\dots \\ ax_n + b &= y_n \end{aligned} \tag{5.9}$$

który sprowadza się do postaci

$$p \cdot X = y, \tag{5.10}$$

gdzie p to wektor współczynników $[a \ b]'$ a X to macierz $n \times 2$, która w pierwszej kolumnie zawiera wektor x , a w drugiej jedynki. Rozwiązanie tego nadokreślonego układu równań (zob. rozdz. 6.2.2) uzyskuje się za pomocą operatora $\backslash$

$$p = y \backslash X \tag{5.11}$$

Ilustruje to przykład:

Przykład 5.8.

 **p5_8.m**

```
% Przykład 5.8 Aproksymacja liniowa 1
```

```
% dane
```

```
x=[1 2 3 4 5 6];
```

```
y=[0.99 2.01 3.1 3.9 4.9 6.1];
```

```
% dopasowanie prostej metodą MNK
```

```
n=length(x);
```

```
A=[x' ones(n,1)];
```

```
p=A\y'
```

Rezultat 5.8

```
p =
```

```
1.0006
```

```
-0.0020
```

5.3.4. Aproksymacja wielomianem

W podstawowej wersji MATLAB oferuje kilka poleceń umożliwiających operacje z użyciem wielomianów. Definiując wielomian, korzysta się tu z wektorowego zapisu współczynników wielomianu uporządkowanych według malejących potęg zmiennej niezależnej np. dla przykładowego wielomianu

$$f(x) = 3x^4 + 5x^3 - 2x^2 + 5x - 1 \quad (5.12)$$

% zdefiniowanie wielomianu wektorem

```
p=[3 5 -2 5 -1]
```

MATLAB oferuje szereg funkcji wykonujących operacje na wielomianach:

`polyval(p,argument)`

oblicza wartość funkcji opisanej wielomianem dla zadanego argumentu, `p` – wektor współczynników wielomianu, `argument` – wartość lub wektor.

`roots(p)`

oblicza pierwiastki wielomianu o współczynnikach zawartych w wektorze `p`.

`polyder(p)`

wyznacza wektor stanowiący pochodną wielomianu zdefiniowanego w wektorze `p`.

`poly([x1 x2 x3])`

generuje współczynniki wielomianu o zadanych miejscach zerowych (inaczej `poly(x)`), gdzie `x` to wektor złożony ze składowych `x1`, `x2` i `x3`.

`conv(p1,p1)`

wyznacza współczynniki wielomianu będącego wynikiem mnożenia wielomianów `p1` i `p2`.

`deconv(p1,p2)`

wyznacza iloraz i resztę z dzielenia wielomianu `p1` przez `p2`.

`polyfit(x,y,n)`

interpolacja wielomianem stopnia `n`.

`polyeig(p0,p1,...,pp)`

oblicza wartości własne wielomianu.

`dot(p1,p2)`

oblicza iloczyn skalarny.

Do aproksymacji wielomianem wg MNK służy funkcja `polyfit`.

`[a]=polyfit(x,y,n)`

funkcja aproksymująca dane doświadczalne wielomianem `n`-tego rzędu, zwraca współczynniki wielomianu, `x`, `y` – wektory punktów doświadczalnych.

Wykorzystanie tej funkcji dla dopasowania równania linii prostej przedstawia przykład 5.9.

Przykład 5.9.

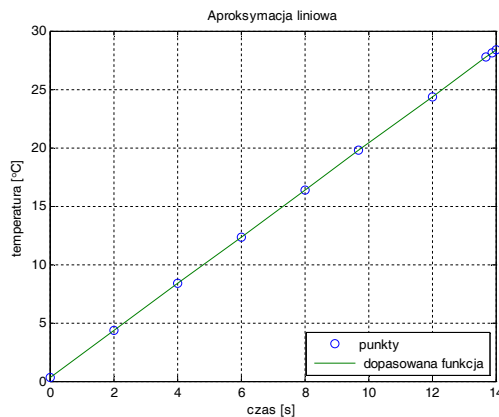
 **p5_9.m**

```
% Przykład 5.9 Aproksymacja liniowa 2
clear all; close all; clc
% punkty doświadczalne
y=[0.34; 4.34; 8.34; 12.34; 16.34; 19.74; 24.34; 27.74; 28.14;
28.34]; %[C]
x=[0.0; 2.0; 4.0; 6.0; 8.0; 9.7; 12.0; 13.7; 13.9; 14.0]; %
[m^3/hr]
a = polyfit(x,y,1)           %znajduje współczynniki wielomianu
yh = polyval(a,x);          %oblicza wartości wielomianu
plot(x,y,'o',x,yh),title('Aproksymacja liniowa')
xlabel('czas [s]'),ylabel('temperatura [\text{C}]')
grid on
legend('punkty','dopasowana funkcja',4)
```

Rezultat 5.9

Otrzymano następujące współczynniki dopasowanej prostej $y=a(1)x+a(2)$

```
a =
    2.0000    0.3400
```



Rys. 5.4. Efekt dopasowania prostej do punktów doświadczalnych

W przykładzie 5.10 pokazano dopasowanie wielomianów wyższych stopni.

Przykład 5.10.

Aproksymacja wielomianami

 **p5_10.m**

```
%Przykład 5.10 Aproksymacja wielomianami
clear all; close all; clc
% Aproksymacja wielomianami wyższych rzędów
x = 0:0.1:1;
y = [-0.447 1.978 3.28 6.16 7.08 7.34 7.66 9.56 9.48 9.30 11.2];
%-----
a2 = polyfit(x,y,2);
disp('współczynniki wielomianu st 2');
disp(a2);
%-----
a3 = polyfit(x,y,3);
disp('współczynniki wielomianu st 3');
disp(a3);
%-----
a10 = polyfit(x,y,10);
disp('współczynniki wielomianu st 10');
disp(a10);
%-----
% instrukcja polyval oblicza wartość wielomianu w danym punkcie
% polyval(wektor współczynników, argumenty)
xi = linspace(0,1,101); %dzieli przedział na 101 elementów
yi2 = polyval(a2,xi);
yi3 = polyval(a3,xi);
yi10 = polyval(a10,xi);
% wykres punktów doświadczalnych i dopasowanego wielomianu
figure(1)
plot(x,y,'o',xi,yi2,'--',xi,yi3,xi,yi10,'g','LineWidth',2);
xlabel('x'), ylabel('y');
title('Aproksymacja wielomianowa');
legend('punkty doświadczalne','wielomian 2 st','wielomian 3
st','wielomian ... 10 st',4);
%-----
grid on
```

Rezultat 5.10

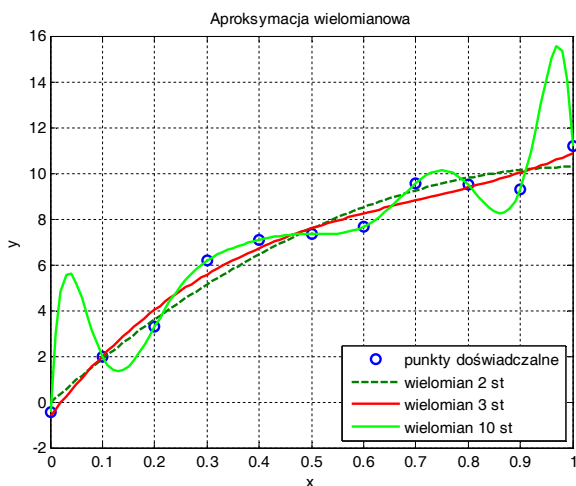
Program znalazł następujące współczynniki wielomianów (przypominamy, że są one podawane od najwyższej potęgi do zerowej).

```

współczynniki wielomianu st 2
-9.8108    20.1293    -0.0317
współczynniki wielomianu st 3
 16.0758  -33.9245   29.3246   -0.6104

współczynniki wielomianu st 10
1.0e+006 *
-0.4644
 2.2965
-4.8773
 5.8233
-4.2948
 2.0211
-0.6032
 0.1090
-0.0106
 0.0004
-0.0000

```



Rys. 5.5. Efekt dopasowania wielomianów różnych rzędów do punktów doświadczalnych

Na załączonym wykresie widać, że nie zawsze wielomian wyższego stopnia lepiej opisuje punkty doświadczalne. W tym przypadku najlepiej opisuje je wielomian drugiego stopnia, natomiast przy aproksymacji wielomianem dziesiątego stopnia występuje tzw. efekt Rungego. Powoduje on poprawę przybliżenia w środku przedziału, ale znaczne pogorszenie na jego krańcach.

PROJEKT 5.1

Tytuł: Aproksymacja danych zależności ciepła właściwego wody
 od temperatury wielomianem trzeciego stopnia
Toolbox: -

Dane wartości ciepła właściwego wody są zmierzone w zależności od temperatury. Należy je przybliżyć wielomianem 3-go stopnia.

P5_1.m

```
%PROJEKT 5.1
%Aproksymacja danych zależności ciepła właściwego
%wody od temperatury wielomianem
clear all; close all; clc
format short e
% dane eksperymentalne
% temperatura [C]
t=[0 20 40 60 80 100 120 140 150 160 180 200 220 240 250 260 280
300]';
% przeliczenie jednostki na [K]
T=t+273.15;
% ciepło właściwe wody [kJ/kg*K]
cc=[4.220 4.183 4.178 4.191 4.199 4.216 4.233 4.258 4.271...
4.283 4.396 4.501 4.605 4.731 4.857 4.982 5.234 5.694]';

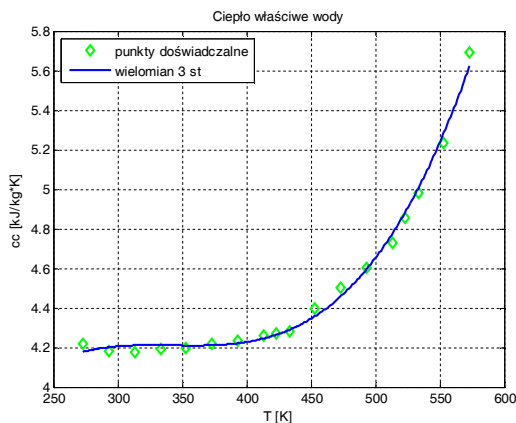
% dopasowanie wielomianu 3st.
a=polyfit(T,cc,3)
Tob=min(T):1:max(T);
cob=polyval(a,Tob);
SSE=sum((polyval(a,T)-cc).^2);
SST=sum((cc-mean(cc)).^2);
R2=1-SSE/SST

%wykres
plot(T,cc,'gd',Tob,cob,'LineWidth',2)
legend('punkty doświadczalne', 'wielomian 3 st',2)
title('Ciepło właściwe wody');
xlabel('T [K]'); ylabel('cc [kJ/kg*K]');
grid on
```

Wyniki 5.1

Uzyskano następujące współczynniki wielomianu i współczynnik dopasowania R^2

$a =$
1.1759e-007 -1.2095e-004 4.1374e-002 -4.9559e-001
 $R^2 =$
9.9438e-001



Rys. 5.6. Dopasowanie wielomianu 3-go stopnia do punktów doświadczalnych

5.3.5. Aproksymacja dowolną funkcją

Jednak nie zawsze chcemy, aby równanie aproksymujące miało postać wielomianu, niejednokrotnie funkcja opisująca jakies zjawisko jest oparta na teorii tego zjawiska. W celu dopasowania takiej funkcji można posłużyć się narzędziami do poszukiwania minimum funkcji celu, sformułowanej jako suma kwadratów odchyłeń wg równania 5.1. Można do tego wykorzystać funkcję `fminsearch` ze skrzynki narzędziowej **Optimization Toolbox**. Działanie tej funkcji jest omówione bliżej w rozdz. 9.

W skrzynce **Statistics Toolbox** znajdują się następujące funkcje pozwalające aproksymować dane dowolną funkcją nieliniową:

```
[a,r,J]=nlinfit([x y],z,@fun,a0)
```

funkcja dopasowuje współczynniki funkcji nieliniowej, wykorzystując kryterium minimum sumy kwadratów odchyłeń.

`aci = nlparci(a,r,J)`

funkcja zwraca zakresy współczynników, przy poziomie ufności 95%

`apred = nlpredci(@fun,[x y],a,r,J)`

funkcja zwraca wartości funkcji w punktach węzłowych.

Oznaczenia:

`[x, y]`

tablica zbudowana z dwóch wektorów danych doświadczalnych, które stanowią argumenty szukanej funkcji.

`z`

wektor danych doświadczalnych.

`@fun`

nazwa funkcji, w której zdefiniowane jest równanie.

`a0`

wektor punktów startowych.

`a`

wektor poszukiwanych współczynników.

`r`

zwrócone residuum.

`J`

zwrócony Jakobian.

Wykorzystanie tych funkcji ilustruje projekt 5.2

PROJEKT 5.2

Tytuł: Dopasowanie równania Watsona do danych zależności ciepła parowania od temperatury

Toolbox: **Optimization Toolbox**

Dane ciepła parowania neoprenu od temperatury są zawarte w pliku **cparow.txt**. Zależność ciepła parowania od temperatury opisuje równanie Watsona postaci

$$\Delta h_v(T) = \Delta h_{v0} \left(1 - \frac{T}{T_{cr}} \right)^n, \quad (5.13)$$

gdzie Δh_{v0} [kJ/kg], n – współczynniki równania, T_{cr} – temperatura krytyczna [K]. Należy estymować współczynniki tego równania dla neoprenu i znaleźć ich zakres w przedziale ufności 95%.

P5_2.m

```
%PROJEKT 5.2
```

```
%Aproksymacja danych zależności ciepła parowania
```

```
%od temperatury równaniem Watsona
```

```
clear all; clc
```

```
global x y t0 Tcr N
```

```

t0=273.15;
Tcr=525.0;
Dane=load('cparow.txt');
x=Dane(:,1);
y=Dane(:,2);
n=length(Dane);
format short e
adol=[1000 0.2];
agor=[8000 0.5];
a0= [4500 0.35];
N=2;
a1=lsqcurvefit(@fun,a0,x,y);
xobl=linspace(min(x),max(x),100);
yobl=fun(a1,xobl);
plot(x,y,'*',xobl,yobl,'-');
xlabel('T [C]'); ylabel('hv [kJ/kg]');
title('Ciepło parowania neoprenu');
%--Obliczenie R2
SSE=sum((fun(a1,x)-y).^2);
SST=sum((y-mean(y)).^2);
R2=1-SSE/SST;
%--obliczenie przedziałów ufności współczynników
alfa=1-0.95; %poziom istotności
t95=tinv(1-alfa/2,n-N); %wartość testu t
syx=sqrt(SSE./(n-N));
p=pfun(a1,x);
p=p^(-1);
s=syx*sqrt(p);
s=diag(s,0);
range=eye(N);
range(:,1)=a1'-s.*t95;
range(:,2)=a1'+s.*t95;
disp('Metoda 1');
disp(a1');
disp(range);
%--dla porównania wartości obliczone przez inne funkcje
[a,r,J]=nlinfit(x,y,@fun,a0);
aci=nlparci(a,r,J);
disp('Metoda 2');
disp(a');
disp(aci);

```

Pozostałe funkcje to

Dfun.m

```
function y=Dfun(a,x)
global t0 Tcr
X=x+t0;
y=[(1-(X./Tcr)).^a(2) a(1)*log(1-X./Tcr).*(1-(X./Tcr)).^a(2)];
```

pfun.m

```
function y=pfun(a,x)
global N
d=Dfun(a,x);
p=ones(N);
for i=1:N
    for j=1:N
        p(i,j)=sum(d(:,i).*d(:,j));
    end;
end;
y=p;
```

Wyniki 5.2

Uzyskano następujące współczynniki i ich zakresy:

- Na podstawie funkcji `lsqcurvefit` i równań 5.3-5.7:

Metoda 1

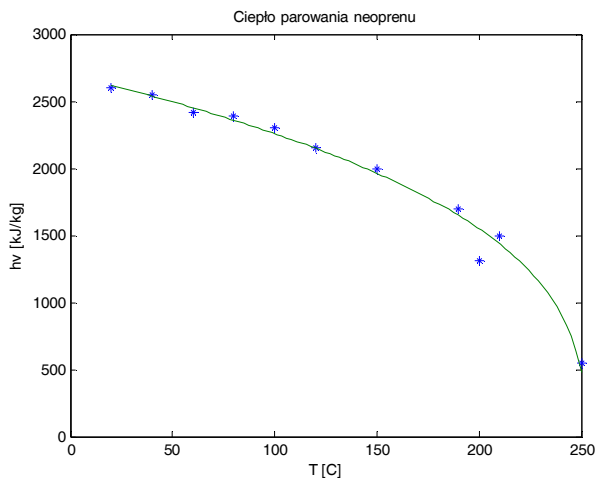
3.4898e+003	
3.5088e-001	
3.2315e+003	3.7482e+003
2.9966e-001	4.0209e-001

- Na podstawie funkcji `nlinfit` i `nlparci`

Metoda 2

3.4898e+003	
3.5088e-001	
3.2315e+003	3.7482e+003
2.9966e-001	4.0209e-001

Otrzymane obiema metodami zakresy przedziału ufności 95% są więc identyczne. Wykres dopasowanego równania przedstawia Rys 5.7.



Rys. 5.7. Wykres dopasowanego równania Watsona

PROJEKT 5.3

Tytuł: Dopasowanie równania izoterm sorpcji metodą minimalizacji sumy kwadratów odchyleń

Toolbox: **Optimization Toolbox**

Do punktów uzyskanych z eksperymentu dopasujemy funkcję aproksymującą metodą najmniejszych kwadratów, korzystając z **Optimization Toolbox**. Zebrane dane doświadczalne zawierają temperaturę, zawartość wody i odpowiadającą im aktywność wody (*Pakowski i in., 2005*). Dane zapisano do pliku tekstowego, oddzielając poszczególne kolumny znakiem tabulacji. Tak przygotowane dane są zawarte w pliku **Dane_izot.txt**. Wiadomo, że zebrane dane najlepiej opisać funkcją dwóch zmiennych: zawartości wilgoci i temperatury. Jako funkcję aproksymującą wybrano zmodyfikowane równanie Hendersona III postaci:

$$a_w = 1 - \exp\left(-a_1(a_2 + T)X^{a_3}\right). \quad (5.14)$$

W celu estymacji parametrów **a** powyższego równania wykorzystano metodę najmniejszych kwadratów. Do poszukiwania minimum funkcji celu danej rów-

naniem 5.1 wykorzystano narzędzie `fminsearch` ze skrzynki **narzędziowej Optimization Toolbox**.

Wywołanie tej funkcji wygląda następująco:

```
fminsearch(@fun,a0,options),
```

gdzie:

@fun	nazwa funkcji oraz pliku, w której zdefiniowane jest równanie,
a0	punkty startowe,
options	opcje optymalizatora.

P5_3.m

```
%PROJEKT 5.3
%Dopasowanie równania izotermy sorpcji
%metodą minimalizacji funkcji celu
clear all; close all; clc
%rozwiązanie równania Henderson III 311
global X T A
z=importdata('Dane_izot.txt','\t',4);
r=z.data;
% podstawienie odpowiednio zmiennych
T=r(:,2);      % temperatura [C]
A=r(:,4);      % aktywność wody [-]
X=r(:,3);      % wilgotność [kg/kg]
% wstępne oszacowanie współczynników
a0=[0.1,0,0.1];
% w opcjach optymalizatora włączono wyświetlanie:
%iteracji, diagnostyki, dokładności zmiennej i funkcji
options=optimset('Display','iter','Diagnostics','on','TolX',1e-5,'TolFun',1e-6);
% wywołanie optymalizatora
a=fminsearch(@Henderson,a0,options)
%---opracowanie wyników
xi=linspace(min(X),max(X),200);
ti=linspace(min(T),max(T),200);
[Xi,Ti]=meshgrid(xi,ti);
Ak=1-(exp(-a(1)*(a(2)+Ti).*(Xi.^a(3))));
% wykres 3D
figure(1);
surf(Xi,Ti,Ak)
alpha(0.7)
```

```

shading interp;
hold on;
plot3(X,T,A,'o')
xlabel('zawartość wilgoci [kg/kg]');
ylabel('temperatura [\text{C}]');
zlabel('aktywność wody [-]');

```

Plik **Henderson.m** zawiera funkcję celu, jest to suma kwadratów odchyleń wartości obliczonych i eksperymentalnych.

Henderson.m

```

function ff=Henderson(a)
% dobór współczynników w równaniu Hendersona III
global X T A
f=1-(exp(-(a(1)*(a(2)+T).*(X.^a(3))));
ff=sum(A-f);

```

Wyniki 5.3

Pokazano część tabeli iteracji, komunikaty końcowe i dobrane współczynniki a.

Iteration	Func-count	min f(x)	Procedure
0	1	18.4017	
1	4	18.3859	initial simplex
2	6	18.2993	expand
3	7	18.2993	reflect
4	9	18.1966	expand
.			
310	562	0.739185	contract inside
311	564	0.739185	contract inside
312	566	0.739185	contract inside
313	568	0.739185	contract inside
314	570	0.739185	contract inside
315	572	0.739185	contract inside
316	577	0.739185	shrink

Optimization terminated:

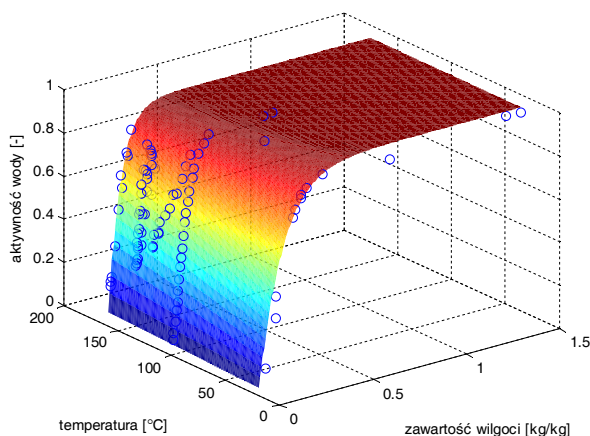
the current x satisfies the termination criteria using OP-
TIONS.TolX of 1.000000e-005
and F(X) satisfies the convergence criteria using OP-
TIONS.TolFun of 1.000000e-006

a =

```

0.0603 125.3750 1.0640

```



Rys. 5.8. Dopasowana powierzchnia równowagi sorpcyjnej wg równania Hendersona III

PROJEKT 5.4

Tytuł: Dopasowanie równania izotermy sorpcji jako funkcji jednej zmiennej, z wyborem równania aproksymacyjnego
 Toolbox: **Statistics Toolbox**

Należy dopasować równanie izotermy desorpcji wody z ciała stałego do punktów uzyskanych z eksperymentu. Do wyznaczenia izotermy desorpcji zastosowano aparaturę umożliwiającą pomiar ubytku masy próbek zawieszonych w termostatowanych pojemnikach nad roztworami nasyconymi odpowiednich soli wytwarzających znaną wilgotność względną (aktywność wody). Ważenie następuje aż do ustalenia się równowagi. Z tak przeprowadzonego pomiaru mamy dane wartości wilgotności względnej i masy próbek w równowadze. Zebrane dane zapisano do pliku tekstowego, wpisując w kolejnych wierszach zawartość wilgoci i aktywność wody i oddzielając je znakiem tabulacji. Tak przygotowane dane zostały zapisane w pliku **data.txt** w bieżącym katalogu. Do rozwiązania problemu wykorzystamy funkcje **nlinfit**, **naparci**, **nlpredci**, które są dostępne w **Statistics Toolbox**.

Równanie izotermy sorpcji musi zostać zdefiniowane jako funkcja w m-pliku, z wykorzystaniem szukanych współczynników. Utworzono 16 takich funkcji np. w następującym m-pliku zdefiniowano równanie Chena-Claytona:

ChC.m

```
function y=ChC(a,x)
global T
y=exp(a(1)*(T.^a(2)).*exp(a(3)*(T.^a(4)).*X));
```

Wykorzystano równania izoterm sorpcji podane w Tablicy 5.1.

Tablica 5.1. Równania izoterm sorpcji niezależne od temperatury

Anderson & Hall	$X = \frac{a_1 a_2 a_3 a_w}{(1 - a_3 a_4 a_w)(1 + (a_2 - a_4))a_3 a_w}$
BET	$X = a_1 \left(\frac{a_2 a_w}{(1 - a_w)(1 + (a_2 - 1)a_w)} \right)$
Bradley	$\ln \left(\ln \left(\frac{1}{a_w} \right) \right) = \ln(a_1) + X \ln(a_2)$
Caurie	$\ln(X) = \ln(a_1) - a_2 a_w$
Filonienko	$X = a_1 \left(\frac{a_w}{a_2 - a_w} \right)^{a_3}$
GAB	$X = \frac{a_1 a_2 a_3 a_w}{(1 - a_3 a_w)[1 + (a_2 - 1)a_3 a_w]}$
Hailwood & Horrobin	$\frac{a_w}{X} = a_1 + a_2 a_w + a_3 a_w^2$
Harkins&Jura	$\ln(a_w) = a_1 - \frac{a_2}{X^2}$
Huttig	$\frac{X}{a_1} = \frac{a_2 a_w}{1 + a_2 a_w} (1 + a_w)$
Jaafar & Michałowski	$X = \frac{a_1 a_2 a_w}{(1 - a_3 a_w)[1 + (a_2 - a_3)a_w]}$

Kühn

$$X = \frac{a_1}{\ln(a_w)} + a_2$$

Rounsley

$$X = \frac{a_1 a_2 a_w (1 - a_w^{a_3})}{(1 - a_w) [1 + (a_2 - 1) a_w]}$$

Sips

$$X = a_1 \left(\frac{a_2 a_w}{1 + (a_2 - 1) a_w} \right)$$

Peleg

$$X = a_1 \exp(a_2 \ln(a_w)) + a_3 a_w^{a_4}$$

Oznaczenia: a_w – aktywność wody [-], X – zawartość wilgoci [kg/kg], a – współczynniki.

P5_4.m

%PROJEKT 5.4

%Dopasowanie równania izotermy sorpcji niezależnego od temperatury

clear all; close all; clc;

disp('-----')

disp('Lp. wywołanie nazwa równania liczba parametrów')

disp('-----')

disp(' 1 @ANHL Anderson&Hall 4 parametrowe')

disp(' 2 @BET BET 3 parametrowe')

disp(' 3 @BRAD Bradley 2 parametrowe')

disp(' 4 @CAU Caurie 2 parametrowe')

disp(' 5 @FIL Filonenko 3 parametrowe')

disp(' 6 @GAB GAB 3 parametrowe')

disp(' 7 @HALH Hailwood&Horrobin 3 parametrowe')

disp(' 8 @HAJ Harkins&Jura 2 parametrowe')

disp(' 9 @HUTT Huttig 2 parametrowe')

disp(' 10 @JAM Jaafar&Michalowski 3 parametrowe')

disp(' 11 @KUH Kuhn 2 parametrowe')

disp(' 12 @OSW Oswin 2 parametrowe')

disp(' 13 @ROU Rounsley 3 parametrowe')

disp(' 14 @SIP Sips 3 parametrowe')

disp(' 15 @PEL Peleg 4 parametrowe')

disp(' 16 @MIZ Mizrahi 2 parametrowe')

disp('-----')

nazwa=input('Wprowadź nazwę wywołania zgodną z listą = ');

disp('Wprowadź startowe współczynniki równania');

```

a0=input('jako wektor kolumnowy np. [1;1;1] = ');
warning off all
format short e
% wczytanie danych z pliku tekstowego
load data.txt
X=data(:,1); Ak=data(:,2);
n=100;
% wywołanie solvera
[a,r,J]=nlinfit(Ak,X,nazwa,a0);
aci = nlparci(a,r,J);
[apred,delta] = nlpredci(nazwa,Ak,a,r,J);
% standardowy błąd bezwzględny
blad=sum(abs(X-apred)./X)./length(X);
odp=[Ak apred delta];
disp('Dopasowanie wykonano dla przedziału ufności 95%')
disp('Współczynniki równania od a1 do an');
disp(a)
disp('Zakres współczynników a');
disp(aci)
disp('wartości funkcji w punktach')
disp('doświadczalna   dopasowana      delta      residuum')
disp([Ak apred delta r])
% wizualizacja wyników
Akob=linspace(0.01,max(Ak),n);
Xob=nazwa(a,Akob);
figure(1)
plot(Ak,X,'rd',Akob,Xob,'Linewidth',2);
grid on
xlabel('aw,-');
ylabel('X, kg/kg');
title('Dopasowana izoterma sorpcji');
legend('punkty dośw','dop. rów.',2);

```

Wyniki 5.4

Poniżej pokazano dialog w oknie **Command Window** po uruchomieniu programu:

Lp.	wywołanie	nazwa równania	liczba parametrów
1	@ANHL	Anderson&Hall	4 parametrowe
2	@BET	BET	3 parametrowe
3	@BRAD	Bradley	2 parametrowe

4	@CAU	Curie	2 parametrowe
5	@FIL	Filonenko	3 parametrowe
6	@GAB	GAB	3 parametrowe
7	@HALH	Hailwood&Horrobin	3 parametrowe
8	@HAJ	Harkins&Jura	2 parametrowe
9	@HUTT	Huttig	2 parametrowe
10	@JAM	Jaafar&Michałowski	3 parametrowe
11	@KUH	Kuhn	2 parametrowe
12	@OSW	Oswin	2 parametrowe
13	@ROU	Rounsley	3 parametrowe
14	@SIP	Sips	3 parametrowe
15	@PEL	Peleg	4 parametrowe
16	@MIZ	Mizrahi	2 parametrowe

Wprowadź nazwę wywołania zgodną z listą = @GAB
Wprowadź startowe współczynniki równania
jako wektor kolumnowy np. $[1;1;1] = [1;1;1]$

Dopasowanie wykonano dla przedziału ufności 95%
Współczynniki równania od a_1 do a_n

7.6491e-002
2.4440e+000
9.4824e-001

Zakres współczynników a

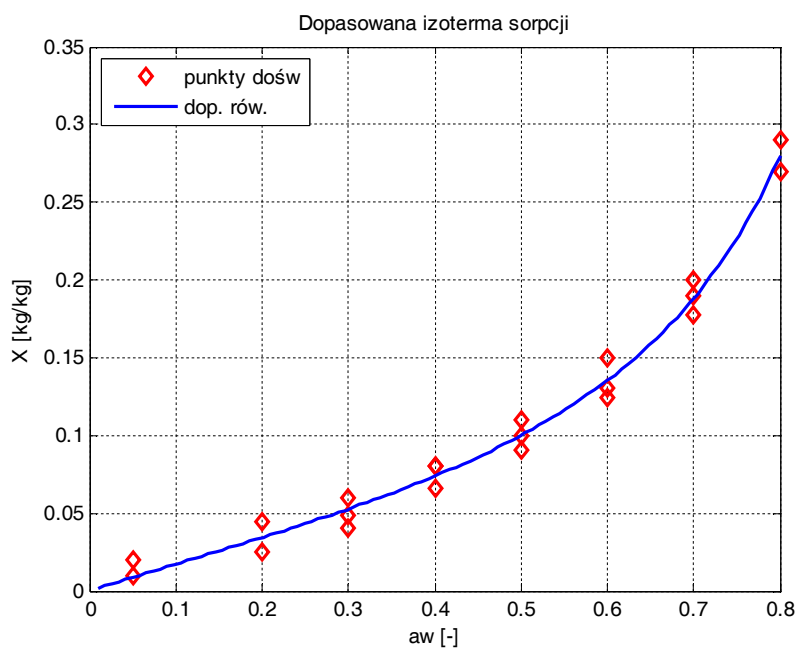
4.9694e-002 1.0329e-001
6.6830e-001 4.2197e+000
8.7045e-001 1.0260e+000

Wartości funkcji w punktach

doswiadczalna	dopasowana	delta	residuum
5.0000e-002	8.7083e-003	3.4248e-003	1.2917e-003
2.0000e-001	3.4345e-002	7.4104e-003	-9.3453e-003
3.0000e-001	5.2683e-002	7.1210e-003	-1.2683e-002
4.0000e-001	7.3811e-002	6.0187e-003	6.1893e-003
5.0000e-001	1.0005e-001	5.7165e-003	-4.8630e-005
6.0000e-001	1.3546e-001	7.1528e-003	1.4541e-002
7.0000e-001	1.8844e-001	8.3778e-003	1.1560e-002
8.0000e-001	2.8035e-001	1.4156e-002	9.6460e-003

5.0000e-002	8.7083e-003	3.4248e-003	1.1292e-002
2.0000e-001	3.4345e-002	7.4104e-003	1.0655e-002
3.0000e-001	5.2683e-002	7.1210e-003	7.3174e-003
4.0000e-001	7.3811e-002	6.0187e-003	6.1893e-003
5.0000e-001	1.0005e-001	5.7165e-003	9.9514e-003
6.0000e-001	1.3546e-001	7.1528e-003	-5.4591e-003
7.0000e-001	1.8844e-001	8.3778e-003	1.5596e-003
8.0000e-001	2.8035e-001	1.4156e-002	-1.0354e-002
3.0000e-001	5.2683e-002	7.1210e-003	-4.4826e-003
4.0000e-001	7.3811e-002	6.0187e-003	-7.7107e-003
5.0000e-001	1.0005e-001	5.7165e-003	-9.5486e-003
6.0000e-001	1.3546e-001	7.1528e-003	-1.1459e-002

Rysunek pokazuje dopasowaną izotermę sorpcji i punkty doświadczalne.



Rys. 5.9. Efekt dopasowania równania izotermi GAB sorpcji do punktów doświadczalnych

5.3.6. Aproksymacja funkcji wielu zmiennych

PROJEKT 5.5	
Tytuł:	Dopasowanie do danych eksperymentalnych równania izoterm sorpcji jako funkcji dwu parametrów, z wyborem równania aproksymacyjnego.
Toolbox:	Statistics Toolbox

Przebieg izoterm sorpcji zależy tutaj od temperatury. Uwzględnienie temperatury wprowadza do równania równowagi sorpcyjnej drugą zmienną niezależną. Do aproksymacji danych sorpcyjnych zależnych od temperatury używa się równań, spośród których wybrano te pokazane w Tablicy 5.2. Kolekcja równań pochodzi z pracy *Pakowski, 2000*.

Tablica 5.2. Równania izoterm sorpcji zależne od temperatury

Aguerre	$\ln(a_w) = \frac{a_1 a_2 a_w}{\frac{1}{a_3} - \frac{1}{T} - 1}$
Chung&Pfof	$\ln(a_w) = -\frac{a_1}{RT} \exp(-a_2 X)$
Day&Nelson	$1 - a_w = \exp\left(-a_1 T^{a_2} X^{a_3 T^{a_4}}\right)$
Day&Nelson II	$1 - a_w = \exp\left[-a_1 (a_5 T + a_6)^{a_2} (100 X)\right]^{a_3 (a_5 T + a_6)^{a_4}}$
Karlsson &Soininen	$a_w = \exp\left[T \exp(-a_1 X - a_2 \sqrt{X}) - \exp(-a_4 X - a_5 \sqrt{X} - a_6)\right]$
Mod. Halsey II	$a_w = \exp\left[-a_1 \exp(a_2 - a_3 T) X^{a_4}\right]$
Mod. Chung &Pfof	$a_w = \exp\left(-\frac{a_1}{a_2 + T} \exp(-a_3 X)\right)$
Modified Halsey I	$a_w = \exp\left[-X^{a_3} \exp(a_1 - a_2 T)\right]$
Modified Henderson I	$\ln(1 - a_w) = a_1 T + a_2 X^{a_3}$

Mod. Henderson II	$a_w = a_3 \left(1 - \exp(a_1 T X^{a_2}) \right)$
Mod. Henderson III	$a_w = 1 - \exp(-a_1 (a_2 + T) X^{a_3})$
Mod. Henderson IV	$\ln(1 - a_w) = -a_1 X^{a_2} T^{a_3}$
Chen&Clayton	$a_w = \exp(a_1 T^{a_2} \exp(a_3 T^{a_4 X}))$
Strohman & Yoerger	$\ln(a_w) = a_1 \ln(p_s) T \exp(a_2 X) + a_3 \exp(a_4 X)$

Oznaczenia: a_w – aktywność wody, T – temperatura, °C, X – zawartość wilgoci, kg/kg, p_s – prężność pary nasyconej, kPa, a – współczynniki.

Dopasuj równanie równowagi, uwzględniające zależność a_w od temperatury i zawartości wilgoci, do punktów uzyskanych z eksperymentu, korzystając z funkcji dostępnych w **Statistics Toolbox**. Dane te zawarte są w pliku **data.txt**.

P5_5.m

```
%PROJEKT 5.5
%Dopasowanie równania izotermy sorpcji zaleznego od temperatury
clear all; close all; clc;
disp('Równowaga sorpcyjna - funkcje 2 zmiennych aw=f(T,X)')
disp('-----')
disp('Lp.      wywołanie      nazwa równania      liczba parametrów')
disp('-----')
disp('1      @AG      Aguerre et.al.      3 parametrowe')
disp('2      @CHP      Chung&Pfof      2 parametrowe')
disp('3      @DNI      Day&Nelson      4 parametrowe')
disp('4      @DNII     Day&Nelson II     6 parametrowe')
disp('5      @HEN      Henderson      2 parametrowe')
disp('6      @KS      Karlsson Soininnen  6 parametrowe')
disp('7      @HII      Modif. Halsey II    4 parametrowe')
disp('8      @CHPM     Modified Chung&Pfof      3 parametrowe')
disp('9      @HALI     Modified Halsey I      3 parametrowe')
disp('10     @HENI     Modified Henderson I      3 parametrowe')
disp('11     @HENII    Modified Henderson II     3 parametrowe')
disp('12     @HENIII   Modified Henderson III    3 parametrowe')
disp('13     @HENIV    Modified Henderson IV     3 parametrowe')
disp('14     @CHC     Chen&Clayton      4 parametrowe')
disp('15     @SYI     Strohman& Yoerger I  4 parametrowe')
disp('-----')
```

```

nazwa=input('Wprowadź nazwę wywołania zgodną z listą = ');
a0=input('Wprowadź oszacowanie współczynników równania jako wektor kolumnowy np. [1;1;1] = ');
warning off all
load data.txt
format short e
T=data(:,1); X=data(:,2); Ak=data(:,3);
n=100;
%-----
[a,r,J]=nlinfit([T X],Ak,nazwa,a0);
aci = nlparci(a,r,J);
[apred,delta] = nlpredci(nazwa,[T X],a,r,J);
% Tworze siatke dla wykresu 3D
Ti=linspace(min(T),max(T),n);
Xi=linspace(min(X),max(X),n);
[Temp,Wilg]=meshgrid(Ti,Xi);
%-----
P=nazwa(a,[T X]);
% standardowy błąd bezwzględny
blad=sum(abs(Ak-P)./Ak)./length(Ak);
disp('Dopasowanie wykonano w przedziale ufności 95%');
disp('Współczynniki równania oraz ich zakresy');
disp('          a          amin          amax');
disp([a aci]);
disp('Maksymalny błąd bezwzględny')
disp(blad)
r1 = reshape(Temp,n*n,1);
r2 = reshape(Wilg,n*n,1);
tmp = nazwa(a,[r1 r2]);
yw = reshape(tmp,n,n);
clear r1 r2 tmp;
figure(1)
surf(Temp,Wilg,yw);
shading interp
alpha(.5)
hold on
plot3(T,X,Ak,'d')
title('Dopasowanie powierzchni równowagi')
xlabel('T, C')
ylabel('X, kg/kg')
zlabel('aw, -')

```

Wyniki 5.5

Po uruchomieniu pliku z programem użytkownik proszony jest o wybór równania i wprowadzenie oszacowania jego parametrów. Dialog ten pokazano poniżej.

Równowaga sorpcyjna - funkcje 2 zmiennych $aw=f(T,X)$

Lp.	wywołanie	nazwa równania	liczba parametrów
1	@AG	Aguerre et.al.	3 parametrowe
2	@CHP	Chung&Pfof	2 parametrowe
3	@DNI	Day&Nelson	4 parametrowe
4	@DNII	Day&Nelson II	6 parametrowe
5	@HEN	Henderson	2 parametrowe
6	@KS	Karlsson Soininnen	6 parametrowe
7	@HII	Modif. Halsey II	4 parametrowe
8	@CHPM	Modified Chung&Pfof	3 parametrowe
9	@HALI	Modified Halsey I	3 parametrowe
10	@HENI	Modified Henderson I	3 parametrowe
11	@HENII	Modified Henderson II	3 parametrowe
12	@HENIII	Modified Henderson III	3 parametrowe
13	@HENIV	Modified Henderson IV	3 parametrowe
14	@CHC	Chen&Clayton	4 parametrowe
15	@SYI	Strohman& Yoerger I	4 parametrowe

Wprowadź nazwę wywołania zgodną z listą = @CHP

Wprowadź prawdopodobne współczynniki występujące w równaniu jako wektor kolumnowy np. $[1;1;1] = [1;1]$

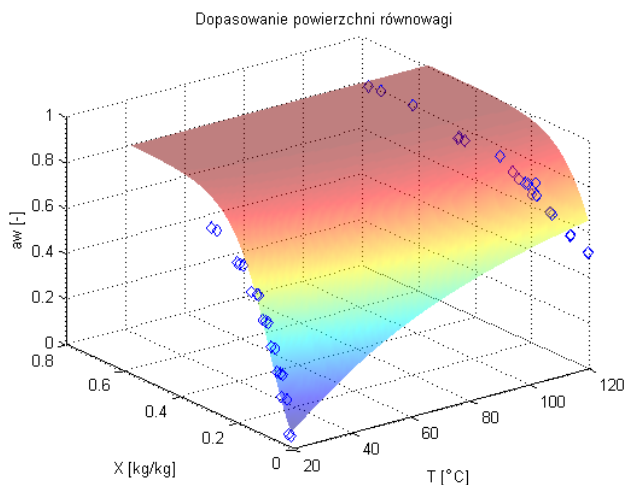
Dopasowanie wykonano w przedziale ufności 95%

Współczynniki równania oraz ich zakresy

a	amin	amax
2.4364e+004	2.0334e+004	2.8394e+004
1.3207e+001	1.1252e+001	1.5162e+001

Maksymalny błąd bezwzględny

1.4285e-001



Rys. 5.10. Dopasowanie powierzchni równowagi sorpcyjnej do punktów doświadczalnych

PROJEKT 5.6

Tytuł: Dopasowanie równania GAB o współczynnikach będących funkcjami temperatury opisanymi wielomianami drugiego stopnia.
 Toolbox: **Statistics Toolbox**

Projekt dotyczy sorpcji wody z fazy gazowej, w różnych temperaturach, przez folie chitozanowe grubości ok. 30 μm , otrzymane przez wylanie określonej ilości roztworu chitozanu, o stopniu deacetylacji $\text{DD} = 78.504\%$ w 1-procentowym kwasie octowym, na płytki z pleksi (Balcerzak, 2005). Wielkość adsorpcji X , czyli masa wody zaadsorbowanej przez jednostkową masę adsorbentu (folii chitozanowej) zależy od temperatury T oraz aktywności wody a_w . Dane doświadczalne otrzymano w następujący sposób: suche próbki ważono na wadze laboratoryjnej, po czym umieszczano w eksykatorach, na perforowanych sitach, w otoczeniu o znanej aktywności wody. Po okresie ustalenia się równowagi adsorpcyjnej, który trwał do 4h, próbki wyjmowano z eksykatorów i niezwłocznie ważono. Wartości adsorpcji wody dla folii chitozanowych X [g H_2O /g s.m.] otrzymywano z następującej zależności:

$$X = \frac{m_1 - m_0}{m_0}, \quad (5.15)$$

gdzie: m_0 – masa suchej próbki, g, m_1 – masa próbki po adsorpcji wody, g.
 Proces sorpcji wody przeprowadzono w trzech temperaturach: 25°C, 35°C oraz 45°C. Do opisu zjawiska sorpcji wody przez folie chitozanowe posłużono się równaniem izotermy GAB:

$$\frac{X}{X_m} = \frac{c k a_w}{(1 - k a_w) [1 + (c - 1) k a_w]} \quad (5.16)$$

Parametry równania GAB: c , K , X_m przedstawiono jako funkcje temperatury:

$$\begin{aligned} c &= a_1 T^2 + a_2 T + a_3 \\ k &= a_4 T^2 + a_5 T + a_6 \\ X_m &= a_7 T^2 + a_8 T + a_9. \end{aligned} \quad (5.17)$$

Analizowany zakres danych doświadczalnych to:

$$T \in \langle 25, 45 \rangle \quad a_w \in \langle 0.2, 0.9 \rangle$$

P5_6.m

```
%PROJEKT 5.6
%Dopasowanie izotermy sorpcji GAB o współczynnikach
%zależnych od temperatury dla chitozanu:
clear all; close all; clc;
global T

% wczytanie danych z pliku tekstowego
z=importdata('dane_GAB.txt',' ',2);
dane=z.data;
X=dane(:,1); aw=dane(:,2); T=dane(:,3);
%punkt startowy:
a0=[0.01 1 4.84 0.0001 0.01 0.1 0.00001 0.001 0.02075];
%regresja nieliniowa-funkcja pozwalająca znaleźć współczynniki
wielomianów
%opisujących parametry równania GAB: c, k i Xm
a=nlinfit([T aw],X,@GAB,a0);

%wykres 2D izoterm sorpcji z dobranymi parametrami a1...a9:
figure(1)
plot(aw(1:7,1),X(1:7,1),'b*')
```

```

hold on
plot(aw(8:14,1),X(8:14,1),'g*')
hold on
plot(aw(15:21,1),X(15:21,1),'r*')
hold on
xlabel('aktywność wody: aw [-]')
ylabel('zawartość wody: X [kg H2O/kg s.m.]')
title('Izotermy sorpcji wody przez folie chitozanowe wg modelu GAB')
%izoterma dla T=25C:
awx=linspace(0.2,0.9,100);
c=a(1).*25.^2+a(2).*25+a(3);
k=a(4).*25.^2+a(5).*25+a(6);
Xm=a(7).*25.^2+a(8).*25+a(9);
Xz=(c.*k.*awx.*Xm)./((1-k.*awx).*(1+(c-1).*k.*awx));
plot(awx,Xz,'b')
grid on
hold on

%izoterma dla T=35C:
c=a(1).*35.^2+a(2).*35+a(3);
k=a(4).*35.^2+a(5).*35+a(6);
Xm=a(7).*35.^2+a(8).*35+a(9);
Xz=(c.*k.*awx.*Xm)./((1-k.*awx).*(1+(c-1).*k.*awx));
plot(awx,Xz,'g')
hold on

%izoterma dla T=45C:
c=a(1).*45.^2+a(2).*45+a(3);
k=a(4).*45.^2+a(5).*45+a(6);
Xm=a(7).*45.^2+a(8).*45+a(9);
Xz=(c.*k.*awx.*Xm)./((1-k.*awx).*(1+(c-1).*k.*awx));
plot(awx,Xz,'r')
legend('25 \circC','35 \circC','45 \circC',2)
hold on
%wykres 3D izoterm sorpcji: punkty-dane doświadczalne, siatka-
wartości z
%modelu;
figure(2)
plot3(T,aw,X,'ko')
hold on
T=25:1:45;
aw=min(aw):0.025:max(aw);

```

```

[Tw,Aw]=meshgrid(T,aw);
c=a(1).*Tw.^2+a(2).*Tw+a(3);
k=a(4).*Tw.^2+a(5).*Tw+a(6);
Xm=a(7).*Tw.^2+a(8).*Tw+a(9);
Xw=(c.*k.*Aw.*Xm)./((1-k.*Aw).*(1+(c-1).*k.*Aw));
surf(Tw,Aw,Xw)

%edycja wykresu 3D
grid on
ylabel('aktywność wody: aw [-]')
zlabel('wielkość adsorpcji: X [kg H2O/kg s.m.]')
xlabel('temperatura: T [C]')
title('X = f(aw,T)')
alpha(0.5)
colormap(hsv)

```

Wyniki 5.6

Współczynniki wielomianów:

```

a1 = -5.4844e+005
a2= 1.3620e+007
a3 = 1.0436e+005
a4 = 2.8105e-004
a5 = -2.1700e-002
a6 = 1.2485e+000
a7 = -3.6251e-005
a8 = 2.1819e-003
a9 = 3.5629e-002

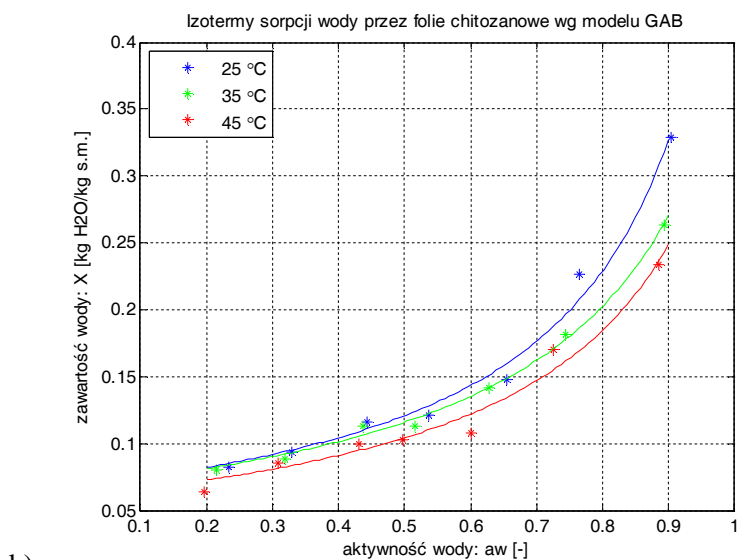
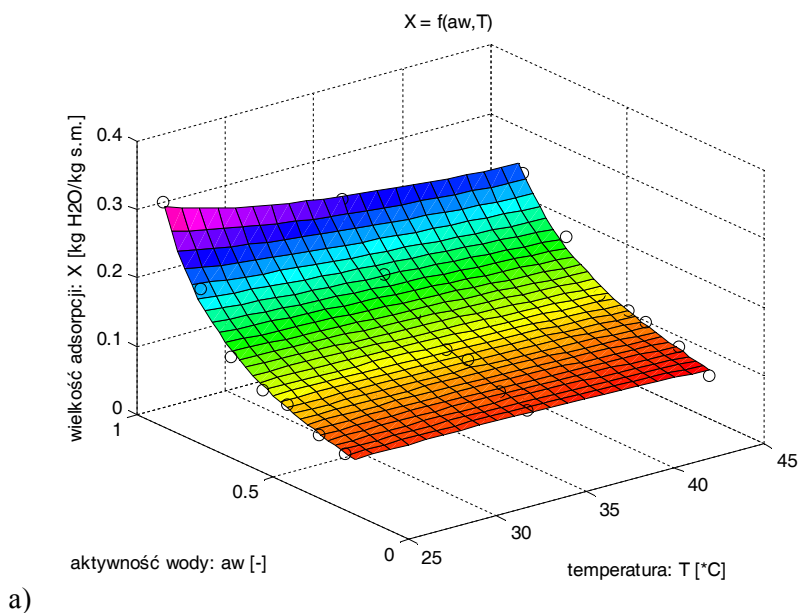
```

Współczynniki korelacji:

```

T = 25°C:      R2 = 0,994
T = 35°C:      R2 = 0,998
T = 45°C:      R2 = 0,986

```

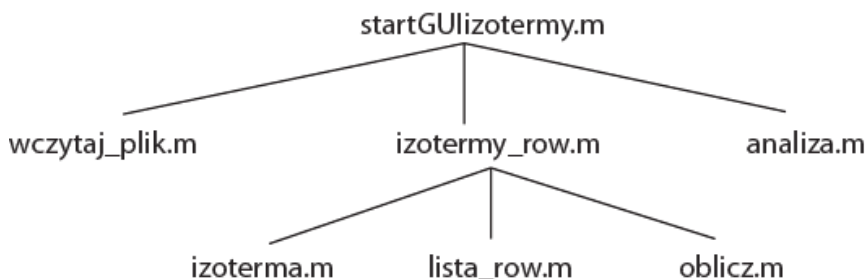


Rys. 5.11. Dopasowana izoterma sorpcji GAB T: a) powierzchnia równowagi, b) krzywe równowagi dla trzech temperatur

PROJEKT 5.7

Tytuł: Dopasowanie równania izotermy sorpcji z wykorzystaniem graficznego interfejsu użytkownika (GUI)
 Toolbox: **Statistics Toolbox**

Środowisko MATLABa daje możliwość stworzenia profesjonalnie wyglądającego graficznego interfejsu użytkownika, ułatwiającego obsługę programu. W katalogu projektu znajduje się program **startGULizotermy.m** z utworzonym graficznym interfejsem. Posługuje się on wykazem równań izoterm sorpcji z Projektów 5.4 i 5.5, jednakże korzystanie z nich odbywa się poprzez system menu – okienek z przyciskami do wyboru opcji i polami do wprowadzania danych. Zależność poszczególnych skryptów zarządzających opcjami jest pokazana na Rys. 5.12.



Rys. 5.12. Schemat połączeń pomiędzy przyciskami GUI w projekcie

Przykładowe dane zawarte są w plikach **dane1.txt** i **dane2.txt**. Główny plik programu ma następującą postać:

P5_7.m

%PROJEKT 5.7

%Wykorzystanie okna Graphic User Interface w programie dopasowującym

%równanie izotermy sorpcji do danych doświadczalnych

%kreuję okno graficzne w którym definiuję rozmiar papieru oraz wyłączam

%domyślne menu okna Figure

```
figure1=figure('PaperSize',[200 200],'MenuBar','none');
```

% kreuję menu z odpowiednimi opcjami

```
f1 = uimenu('Label','Dane');
```

```
    uimenu(f1,'Label','Wczytaj plik z danymi', 'Callback','wczytaj_plik');
```

```
f2 = uimenu('Label','Dopasuj równanie');
```

```

    uimenu(f2,'Label','Izotermy sorp-
cji','Callback','izotermy_row');
f3 = uimenu('Label','Wyniki');
    uimenu(f3,'Label','Wartości współczynni-
ków','Callback','wspolczynniki');
    uimenu(f3,'Label','Analiza statystycz-
na','Callback','analiza');
%tekst informacyjny wprowadzony w oknie
uicontrol('Style','text','String','Dopasowanie izotermy sorp-
cji do danych doświadczalnych',...
    'FontSize',16,'FontWeight','bold',...
    'BackgroundColor',[1 1 0.8],...
    'Position',[370 300 180 110])

```

Jak widać kreuje on jedynie główne okno programu o nazwie **Figure 1**. Zarządzanie pozostałymi plikami odbywa się przez menu stworzone w oknie **Figure 1**. Po uruchomieniu skryptu ukaze się puste okno **Figure 1** (Rys. 5.13) z menu zawierającym opcje **Dane**, **Dopasuj równanie**, **Wyniki**. W pierwszym etapie należy wczytać dane doświadczalne z pliku tekstowego, matlabowego lub excelowskiego. W tym celu wybieramy z menu **Dane** opcję *Wczytaj plik z danymi*. Następnie wybieramy z menu **Dopasuj równanie** opcję *Izotermy sorpcji*. Po wyborze na ekranie pojawi się okno, w którym mamy możliwość wyboru równania izotermy sorpcji – Rys. 5.14. Po wyborze jest konieczne podanie przybliżonych wartości współczynników występujących w równaniu, które posłużą jako wartości początkowe w procesie dopasowywania, lub zatwierdzenie sugerowanych przez program. Po ich wprowadzeniu wciskamy przycisk **OBLICZ**. Efektem będzie uzyskanie wykresu z naniesionymi danymi doświadczalnymi i z dopasowanym równaniem – Rys. 5.15, natomiast wartości współczynników i ich zakresy przy 95% poziomie ufności uzyskamy po wybraniu odpowiedniej opcji z menu **Wyniki** – Rys. 5.16.

Poniżej zostaną przedstawione pozostałe najważniejsze skrypty wykorzystane w tym projekcie.

wczytaj_plik.m

```

% Otwarcie okna wyboru pliku
% [nazwa_pliku, sciezka] =uigetfile({'*.m;*.fig;*.mat;*.mdl',...
% 'Wszystkie pliki Matlaba (*.m, *.fig, *.mat, *.mdl)'; '*.m',
% 'M-files (*.m)'; ...
% '*.fig','Figures (*.fig)'; '*.mat','MAT-files (*.mat)';
% '*.mdl','Models (*.mdl)'; ...
% '.*', 'All Files (*.*)'}, 'Otwórz plik') ;

```

```
[nazwa_pliku, sciezka] =uigetfile({'*.txt','Text Files
(*.txt)'} , 'Otwórz plik') ;
% załadowanie danych z pliku
dane=load(nazwa_pliku);
% przydzielenie zmiennym danym
X =dane(:,1);    % wilgotność
Ak=dane(:,2);    % aktywność wody
```

izotermy_row.m

```
figure(3)
% Wprowadzenie tekstu - opisu, informacji
% ss= 'Anderson&Hall      4 parametrowe'+...
%      '|BET              3 parametrowe'+...
%      '|Bradley          2 parametrowe'+...
%      '|Caurie           2 parametrowe'+...
%      '|Filonenko        3 parametrowe'+...
%      '|GAB              3 parametrowe'+...
%      '|Hailwood&Horrobin 3 parametrowe'+...
%      '|Harkins&Jura      2 parametrowe'+...
%      '|Huttig           2 parametrowe'+...
%      '|Jaafar&Michalowski 3 parametrowe'+...
%      '|Kuhn             2 parametrowe'+...
%      '|Oswin            2 parametrowe'+...
%      '|Rounsley         3 parametrowe'+...
%      '|Sips             3 parametrowe'+...
%      '|Peleg            4 parametrowe';
uicontrol('Style','text','String','Wybierz równanie z li-
sty',...
    'FontSize',12,'FontWeight','bold',...
    'Position',[10 380 350 25])
% Wstawienie listy wyboru
hpop = uicontrol('Style','popup',...
    'String','Anderson&Hall      4 parametrowe|BET
3 parametrowe|Bradley          2 parametrowe|Caurie
2 parametrowe|Filonenko        3 parametrowe|GAB
3 parametrowe|Hailwood&Horrobin 3 parametrowe|Harkins&Jura
2 parametrowe|Huttig           2 parametrowe|
we|Jaafar,Michalowski 3 parametrowe|Kuhn             2 para-
metrowe|Oswin                2 parametrowe|Rounsley      3
parametrowe|Sips              3 parametrowe|Peleg
4 parametrowe',...
    'FontSize',12,'FontWeight','bold',...
    'Position',[10 320 350 50],...
```



```

        'Callback', 'izoterma');
% przycisk uruchamiający listę równań stworzoną w html
uicontrol('Style', 'pushbutton', 'String', 'LISTA RÓWNAŃ',...
    'FontSize',10,'FontWeight','bold',...
    'Position', [400 340 130 30], 'Callback', 'lista_row');
% uruchamia proces obliczeniowy
uicontrol('Style', 'pushbutton', 'String', 'OBLICZ',...
    'FontSize',10,'FontWeight','bold',...
    'Position', [400 50 130 30], 'Callback', 'oblicz');

```

wspolczynniki.m

```

% zestawienie wyników
aodp=[a aci];
% przedstawienie wyników w tabeli
cnames1 = {'wartości współczynników','zakresy współczynników'};
t1 = uitable('Data',aodp,'ColumnName',cnames1,...
    'Position',[10 320 355 63]);

```

analiza.m

```

% przedstawienie danych w postaci tabelarycznej
cnames = {'wartości danych doświadczalnych','wartości funkcji
dopasowanej','delta','residuum'};
t = uitable('Data',odp,'ColumnName',cnames,'Position',[10 10 355
290]);

```

izoterma.m

```

% pobranie danych z menu rozwijalnego z równaniami izoterm
val = get(hpop,'Value');
% podstawienie pod zmienną nazwa, nazwy funkcji która zawiera
równanie danej
% postaci, wykorzystano funkcję wyboru switch
switch val
    case 1
        nazwa=@ANHL; na=4; cztery;
    case 2
        nazwa=@BET; na=3; trzy;
    :
    case 15
        nazwa=@PEL; na=4; cztery;
    otherwise
        disp('Unknown method.')
end

```

lista_row.m

% otwarcie strony html w przeglądarce wbudowanej w matlab
web lista.htm

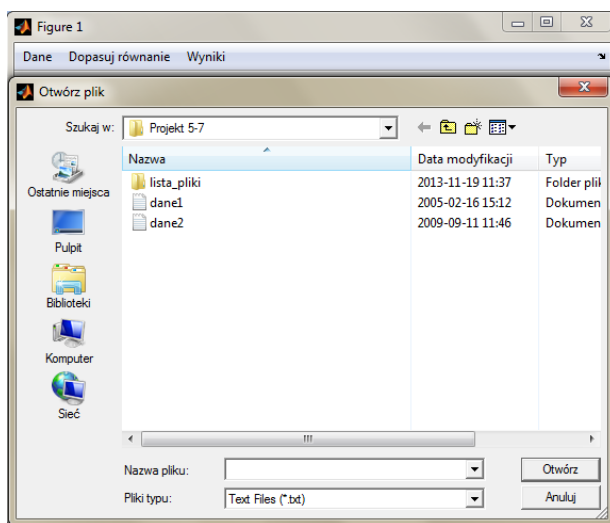
oblicz.m

```
% zdefiniowanie liczby punktów startowych
if na==2
    a0=[C1;C2];
elseif na==3
    a0=[C1;C2;C3];
elseif na==4
    a0=[C1;C2;C3;C4];
end

% wywołanie solvera
[a,r,J]=nlinfit(Ak,X,nazwa,a0);
aci = nlparci(a,r,J);
[apred,delta] = nlpredci(nazwa,Ak,a,r,J);
n=100;
odp=[Ak apred delta r];
%wizualizacja wyników
Akob=linspace(0.01,max(Ak),n);
Xob=nazwa(a,Akob);
figure(4)
hold all
plot(Ak,X,'rd',Akob,Xob,'Linewidth',2);
grid on
xlabel('Ak'); ylabel('X');
title('Dopasowanie izotermy sorpcji');
legend('punkty dosw','dop. rów.',2);
close(3)
```

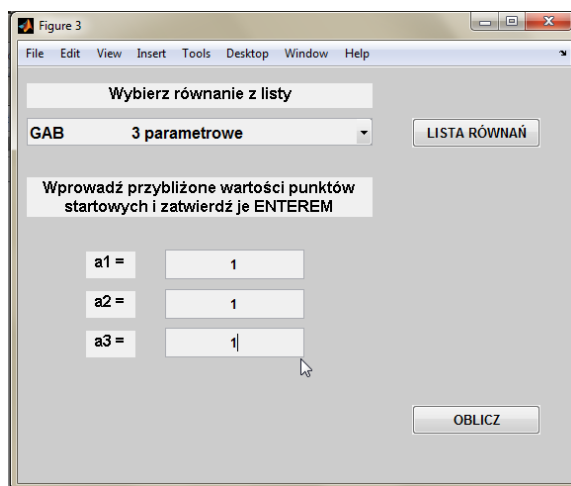
Wyniki 5.7

Przed rozpoczęciem pracy należy w katalogu, w którym znajduje się program, umieścić plik z danymi. Plik powinien mieć dwie kolumny danych: w pierwszej X, kg/kg w drugiej odpowiadające wartości a_w , -. Poniżej pokazane zostaną wyniki dopasowania dla pliku danych **dane1.txt**. Pierwszym oknem programu jest menu główne **Figure 1** – Rys. 5.13.



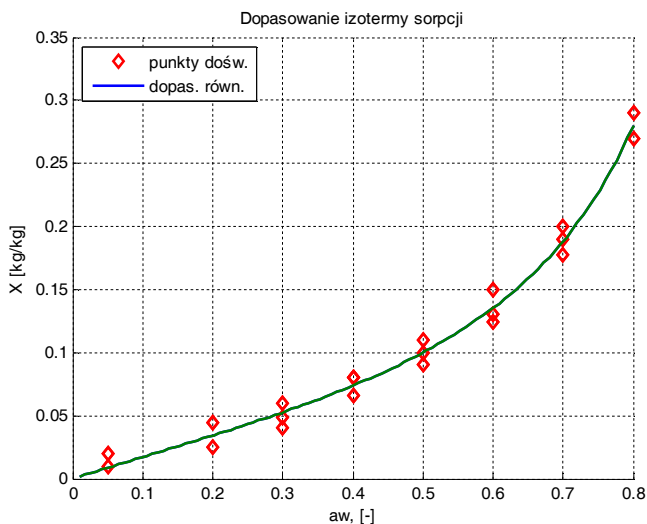
Rys. 5.13. Okno menu głównego - wybór pliku z danymi

Wyboru równania dokonuje się w kolejnym oknie Rys. 5.14.



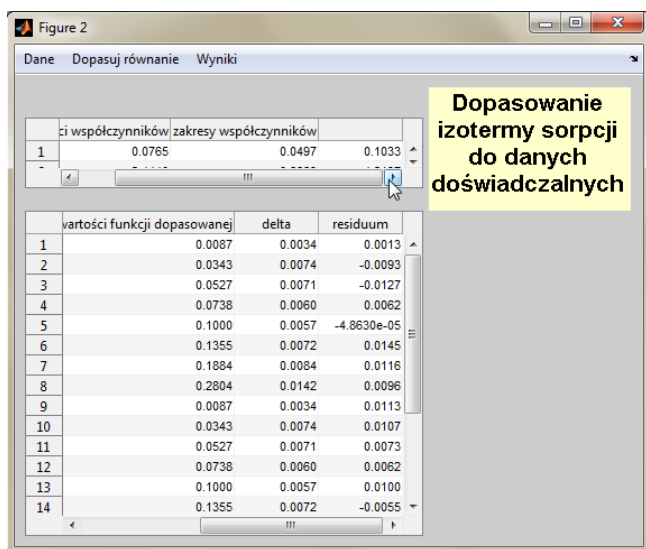
Rys. 5.14. Okno wyboru izotermy sorpcji

W oknie tym można obejrzeć listę równań (wybierz przycisk **LISTA RÓWNAŃ**) i wybrać jedno z rozwijanej listy. Po wyborze można wpisać w pola współczynników wartości proponowane (im lepsze podamy przybliżenie tym program lepiej dopasuje równanie) lub zatwierdzić klawiszem Enter każdy z sugerowanych przez program współczynników.



Rys. 5.15. Dopasowana izoterma sorpcji

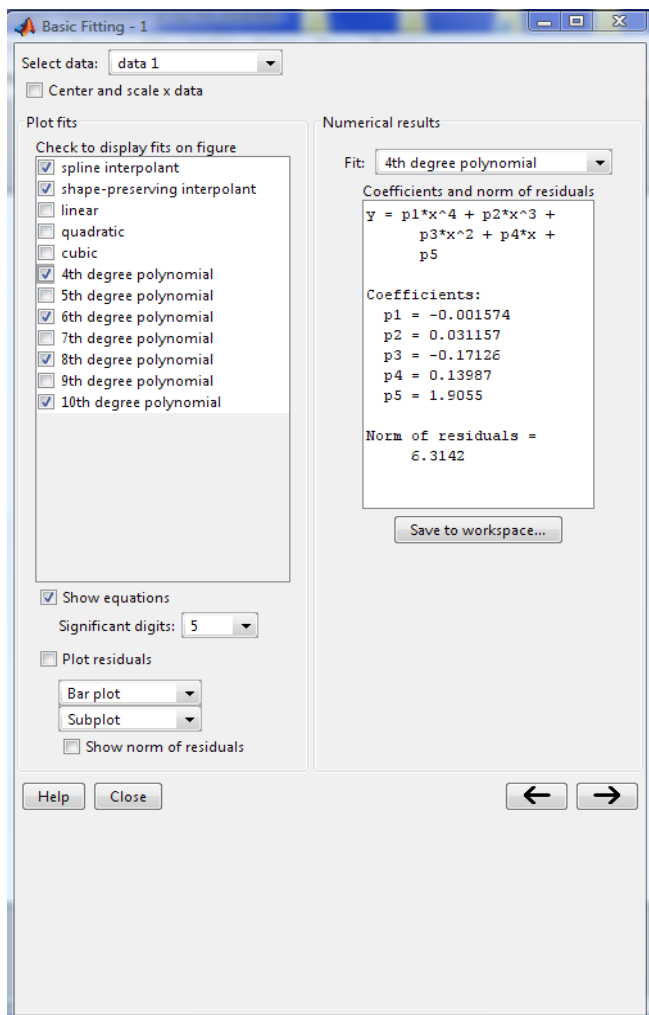
W kolejnym oknie odczytamy współczynniki dopasowanego równania oraz odczytamy wartości residuów dopasowania.



Rys. 5.16. Efekt wyboru opcji Wyniki z menu głównego

5.3.7. Wykorzystanie narzędzia Basic Fitting

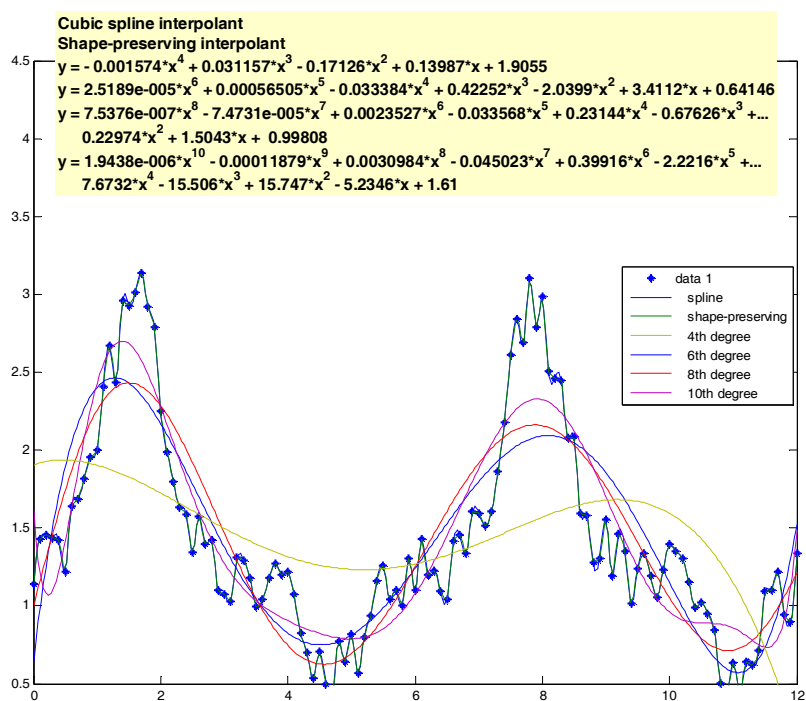
MATLAB udostępnia narzędzie **Basic Fitting** dostępne w oknie graficznym **Figure** w pozycji **Tools** menu głównego tego okna. Okno **Basic Fitting** dotyczy danych w aktualnym oknie **Figure** i dlatego ma numer identyczny z numerem okna **Figure**. Wśród krzywych aproksymacyjnych oferowanych przez **Basic Fitting** podstawową jest wielomian.



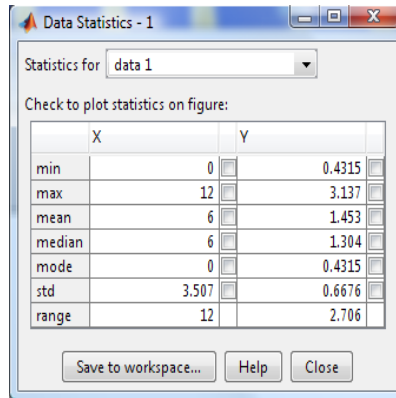
Rys. 5.17. Okno Basic Fitting z możliwością wyboru funkcji aproksymującej

Prowadząc eksperymenty, otrzymujemy serię punktów doświadczalnych charakteryzujących zjawisko. Zdarza się często, że mechanizmy jakimi rządzą się zjawiska są nierozpoznane i nie ma możliwości opisanie danego zjawiska teoretycznym modelem matematycznym. W takiej sytuacji do opisanie przebiegu zmian wykorzystać można podejście empiryczne, tzn. opisać zjawisko za pomocą popularnego równania empirycznego np. wielomianu.

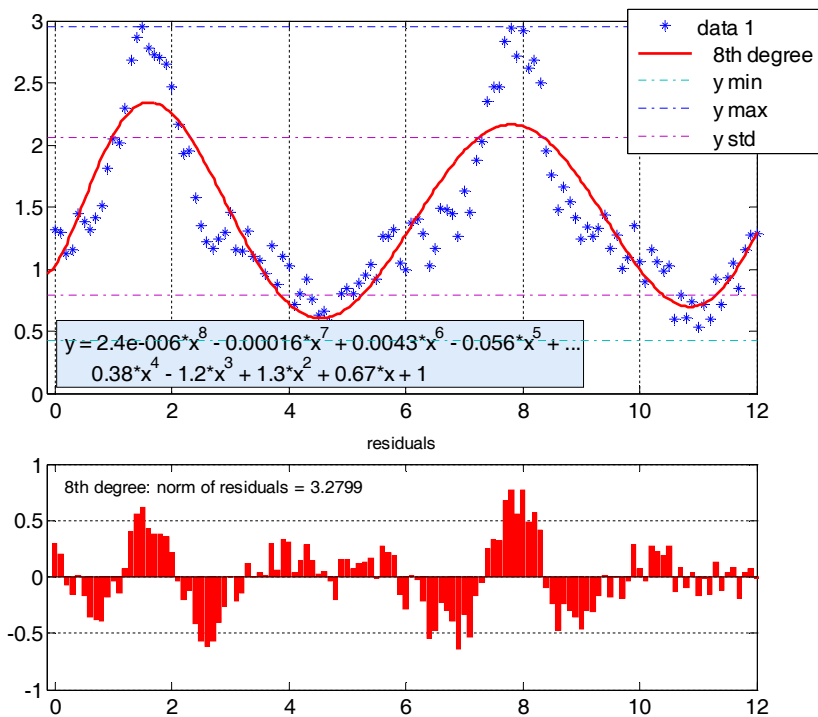
Opcja taka o nazwie **Basic Fitting** jest dostępna w oknie zawierającym wykres w menu **Tools**. Po wybraniu tej opcji, obok okna z wykresem pojawi się okno **Basic Fitting** (Rys. 5.17).



Rys. 5.18. Efekt dopasowania krzywej do danych doświadczalnych z pomocą opcji **Basic Fitting** dostępnej w menu **Tools** okna wykresu



Rys. 5.19. Okno **Data Statistics**



Rys. 5.20. Efekt użycia opcji **Basic Fitting**

Okno **Basic Fitting**, pokazane na Rys. 5.17, zawiera menu rozwijalne **Select data** znajdujące się w górnej części okna, w którym wybieramy dane do opisanego równaniem. Poniżej po lewej stronie znajduje się okno z możliwością wyboru równania, za pomocą którego opiszemy dane. Po prawej stronie w oknie ukaże się postać równania, które wybraliśmy oraz wartości współczynników. Istnieje możliwość zapisania wartości tych współczynników do pamięci roboczej w postaci macierzy. Poniżej znajdują się opcje, za pomocą których możemy wyświetlić postać równania na wykresie, decydujemy również o liczbie wyświetlanych cyfr po przecinku. Na Rys. 5.18 przedstawiono efekty, jakie można uzyskać za pomocą przedstawionego narzędzia.

W oknie **Basic Fitting** istnieje możliwość przedstawienia w formie graficznej residuów Rys. 5.20. Dodatkowo w menu **Tools** istnieje opcja **Data Statistics**, która uruchamia okno z podstawowymi danymi statystycznymi – rys 5.19.

Korzystanie z narzędzia **Basic Fitting** jest interaktywne podobnie jak korzystanie z narzędzia **Curve Fitting Tool**, które udostępnia zaawansowane metody aproksymacji i obróbki danych.

5.3.8. Korzystanie z narzędzia Curve Fitting Tool

Zaawansowane narzędzia do obróbki danych zawiera skrzynka **Curve Fitting Tool**. Pozwala ona dokonać wygładzenia, interpolacji i aproksymacji danych różnymi metodami.

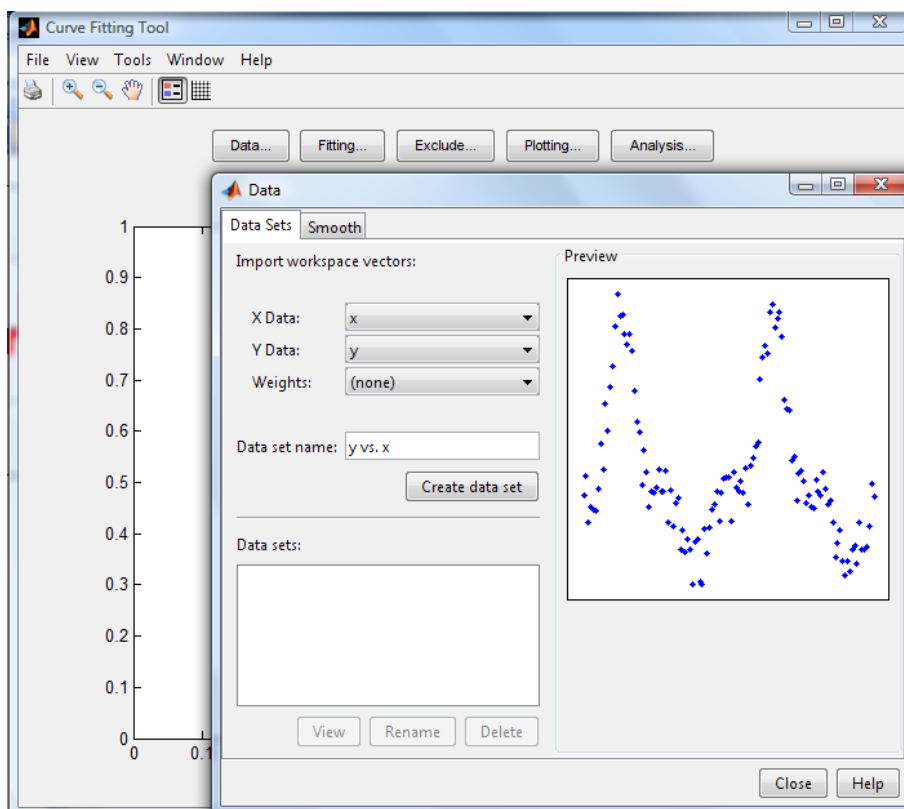
Dane do obróbki w postaci zmiennej (zmiennych) niezależnej i zmiennej zależnej powinny znajdować się w pamięci roboczej (**Workspace**). Można je tam umieścić przez wprowadzenie z klawiatury bądź wczytanie z pliku, jak podano w sekcji 5.1.2. Skrzynkę **Curve Fitting Tool** uruchamia się poleceniem `cftool`.

Po uruchomieniu ukaże się okno Rys. 5.21, w którym jest zamieszczony pusty wykres a nad nim przyciski:

Data	wprowadzenie danych do obliczeń
Fitting	dopasowanie równania do danych doświadczalnych
Exclude	wycięcie zbędnych danych
Plotting	rysowanie wykresów
Analysis	analiza pobranych danych

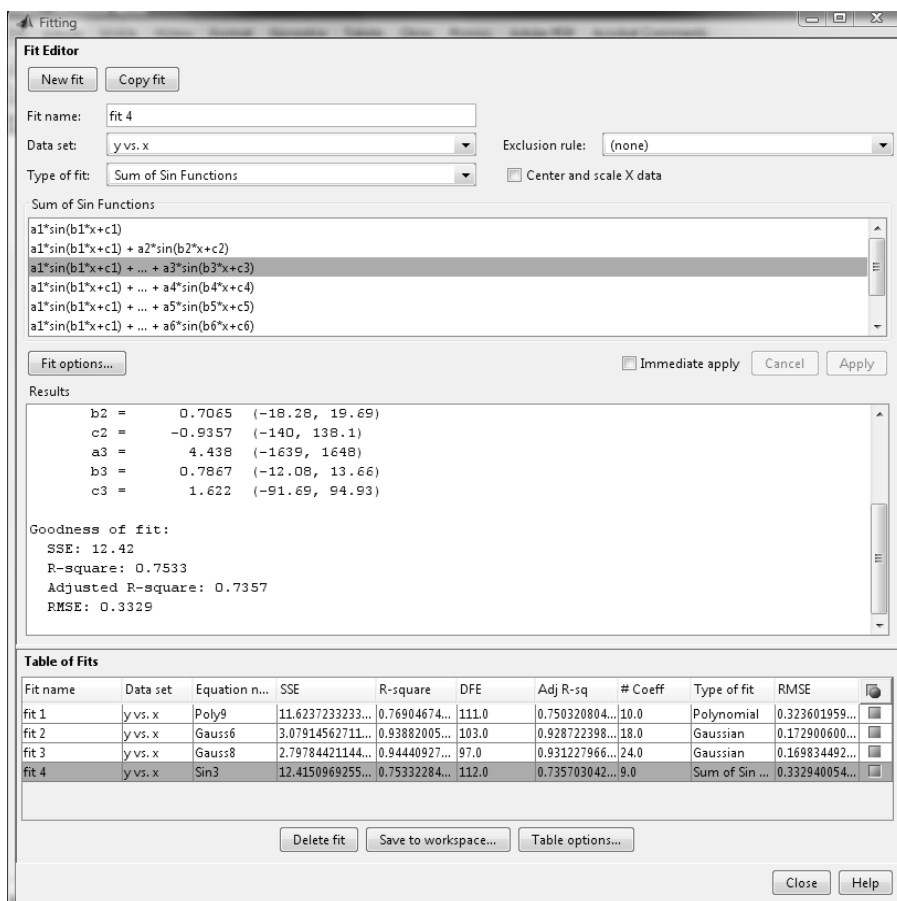
Przycisk **Data** otwiera okno o tej samej nazwie, które zawiera opcje importu danych w postaci wektorów z pamięci programu do procedury obliczeniowej.

Czynność ta jest maksymalnie ułatwiona poprzez zastosowanie graficznego interfejsu wyboru. Program daje również możliwość wczytania wag dla danych doświadczalnych. Po zatwierdzeniu wyboru poleceniem **Create data set** w prawej części okna pojawi się podgląd danych, które wybraliśmy. Podgląd danych jest o tyle użyteczny, że daje możliwość skorygowania błędnego wyboru przed finalnym zatwierdzeniem i zamknięciem okna.



Rys. 5.21. Okno Curve Fitting Tool wraz z oknem uruchomionym przyciskiem Data

Aproksymacji danych dokonuje się w oknie **Fitting** pokazanym na Rys. 5.22. Aproksymację zaczynamy od wciśnięcia przycisku **New fit** otwierającego nową sesję, której możemy nadać nazwę w polu **Fit name**. W polu **Data set** wybieramy dane na których będą prowadzone obliczenia, a w polu **Type of fit** wybieramy typ/rodzaj funkcji aproksymującej dane doświadczalne.



Rys. 5.22. Okno uruchamiane przyciskiem Fitting

W tym polu mamy możliwość wyboru jedenastu opcji, pokazanych poniżej.

- *Custom Equations* – daje możliwość wprowadzenia dowolnego równania zdefiniowanego przez użytkownika
- *Exponential*

$$y = a \exp(bx)$$

$$y = a \exp(bx) + c \exp(dx)$$

(5.18)

- *Fourier*

$$y = a_0 + a_1 \cos(xw) + b_1 \sin(xw) + \dots$$

$$a_8 \cos(8xw) + b_8 \sin(8xw) \quad (5.19)$$

- *Gaussian*

$$y = a_1 \exp\left(-\left(\frac{x-b_1}{c_1}\right)^2\right) + \dots + a_8 \exp\left(-\left(\frac{x-b_8}{c_8}\right)^2\right) \quad (5.20)$$

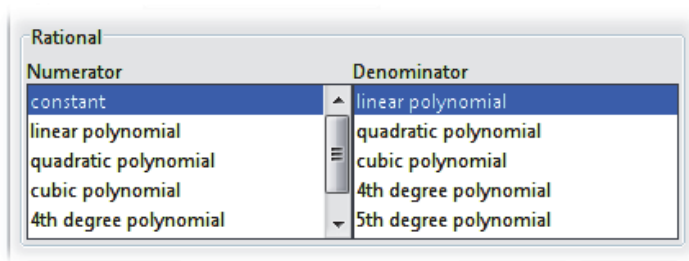
- *Interpolant* – jedna z trzech metod interpolacji danych doświadczalnych: *nearest neighbor*, *cubic spline*, *shape preserving*.
- *Polynomial* – wielomiany od pierwszego do dziewiątego stopnia
- *Power* – funkcja potęgowa

$$y = ax^b \quad (5.21)$$

$$y = ax^b + c$$

- *Rational*

Funkcje ułamkowe, których licznik i mianownik pokazane są na Rys. 5.23



Rys. 5.23. Możliwości wyboru opcji Rational

- *Smoothing Spline*

Krzywa sklejana o zmiennym parametrze wygładzania.

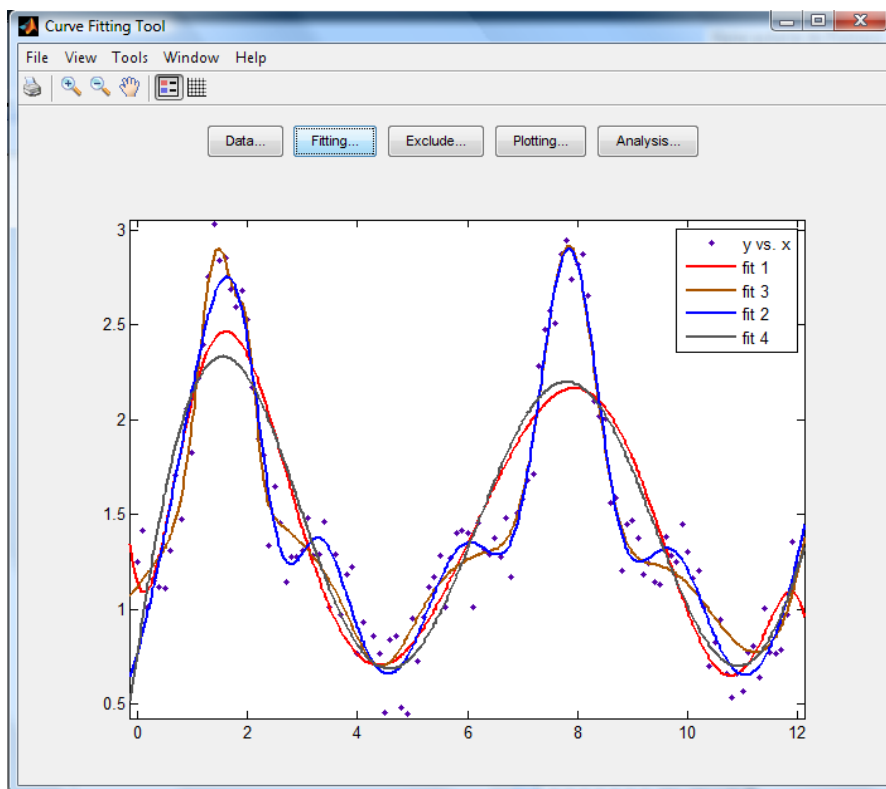
- *Sum of sin Function*

$$y = a_1 \sin(b_1 x + c_1) + \dots + a_8 \sin(b_8 x + c_8) \quad (5.22)$$

- *Weibull*

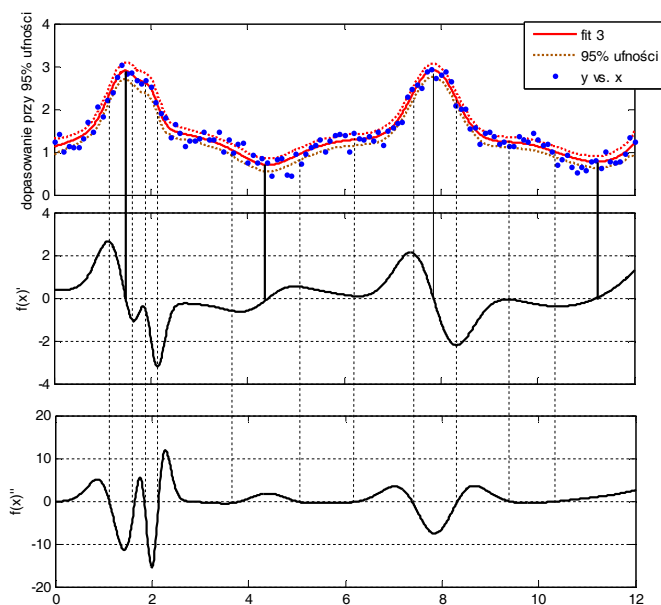
$$y = abx^{(b-1)} \exp(-ax^b) \quad (5.23)$$

Po dokonaniu wyboru funkcji należy ją zatwierdzić, wciskając przycisk **Apply**. Po akceptacji otrzymamy wyniki w okienku **Results** oraz wykres dopasowanego równania do punktów doświadczalnych w oknie głównym CFT – Rys. 5.24. W okienku **Results** przedstawiona jest postać funkcji jaką wybrano, wartości współczynników oraz parametry informujące o jakości dopasowania.



Rys. 5.24. Wykres punktów doświadczalnych z naniesioną dopasowaną funkcją

Następnym krokiem jest przejście do opcji **Analysis**, w której mamy możliwość doboru kroku argumentów funkcji, przewidywania wartości funkcji przy zadanym przedziale ufności oraz obliczenie pierwszej i drugiej pochodnej. Wyniki przedstawia Rys. 5.25.



Rys. 5.25. Wykres dopasowania funkcji do danych doświadczalnych, przedstawienie pierwszej i drugiej pochodnej, zaznaczenie minimów, maksimów i punktów przebiegu funkcji

Należy dodać, że dla danych 3D istnieje analogiczne narzędzie **Surface Fitting Tool**, które uruchamia się poleceniem `sftool`, którego tu nie omówiono.

5.4. Interpolacja danych

Interpolacja wykorzystywana jest często podczas przedstawiania graficznego danych doświadczalnych. Interpolacja polega na wyznaczeniu wartości funkcji w innych punktach niż w węzłach interpolacji, tzn. należy znaleźć taką funkcję $f(x)$ interpolującą dane, która w punktach węzłowych przyjmie wartości tych danych. W MATLABie interpolację przeprowadza się za pomocą funkcji:

- Funkcje interpolujące

`interp1(x, y, xi, 'metoda interpolacji')`

interpolacja danych jednowymiarowych.

`interp1q(x, y, xi)`

interpolacja taka jak `interp1` tylko szybsza.

`interp2(x, y, z, xi, yi, 'metoda interpolacji')`

interpolacja danych dwuwymiarowych.

`interp3(x,y,z,v,xi,yi,zi,'metoda interpolacji')`

interpolacja danych trójwymiarowych.

`interpN(x1,x2,x3...,v, z1, z2, z3,...z, 'metoda interpolacji')`

interpolacja danych wielowymiarowych.

- Metody interpolacji

`'nearest'` interpolacja sąsiedzka

`'linear'` interpolacja liniowa

`'cubic'` interpolacja wielomianem trzeciego rzędu

`'pchip'` to samo co `'cubic'`

`'spline'` interpolacja wielomianami sklejanymi trzeciego rzędu.

Zastosowanie różnych metod interpolacji danych funkcji jednej zmiennej przedstawia Przykład 5.11, a funkcji dwu zmiennych Przykład 5.12.

Przykład 5.11. Interpolacja jednowymiarowa

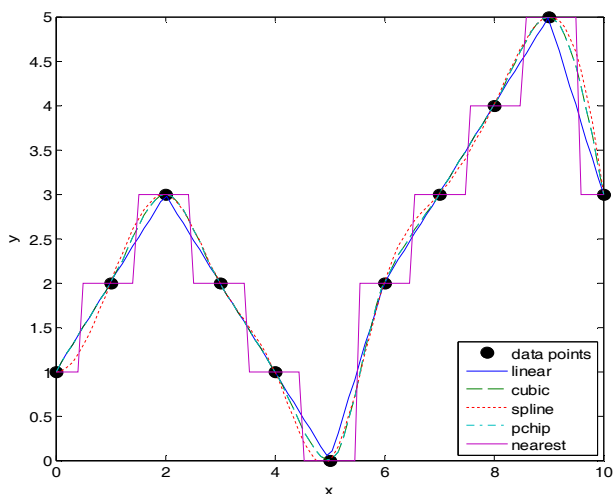
 p5_11.m

```
%Przykład 5.11 Interpolacja jednowymiarowa
clear all; close all; clc
% dane doświadczalne w dwóch wektorach x i y
x=[0; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10];
y=[1; 2; 3; 2; 1; 0; 2; 3; 4; 5; 3];
% wektor wartości pomocniczych,
% dla których zostanie przeprowadzone obliczenie
xi=linspace(min(x),max(x),100);
% interpolacja z opcją linear
y1=interp1(x,y,xi,'linear');
% interpolacja z opcją cubic
y2=interp1(x,y,xi,'cubic');
% interpolacja z opcją spline
y3=interp1(x,y,xi,'spline');
```

```
%interpolacja z opcją pchip
y4=interp1(x,y,xi,'pchip');
%interpolacja z opcją nearest
y5=interp1(x,y,xi,'nearest');

%wizualizacja wyników
plot(x,y,'k.',xi,y1,xi,y2,xi,y3,xi,y4,xi,y5)
legend('data
points','linear','cubic','spline','pchip','nearest',4)
xlabel('x'); ylabel('y')
```

Rezultat 5.11



Rys. 5.26. Efekt interpolacji jednowymiarowej punktów doświadczalnych różnymi metodami

Punkty użyte w Przykładzie 5.12 zostały wygenerowane w siatce 10x10, a interpolowane w siatce 30x30.

Przykład 5.12. Interpolacja dwuwymiarowa

 **p5_12.m**

```
%Przykład 5.12 Interpolacja dwuwymiarowa
clear all; close all; clc
```

```

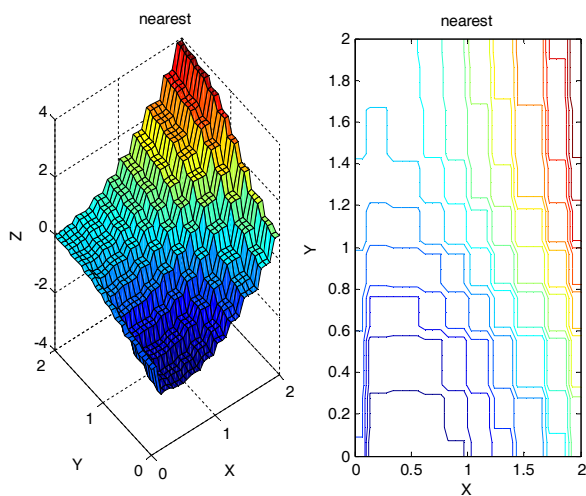
% wygenerowane punkty
x=linspace(0,2,10);
y=x;
[X,Y]=meshgrid(x,y);
Z=X.^2-exp(X.^2-Y.^2);
% siatka dla interpolacji
xi=linspace(0,2,30); yi=xi;
[Xi,Yi]=meshgrid(xi,yi);

% interpolacja dwuwymiarowa
Zi1=interp2(X,Y,Z,Xi,Yi,'nearest');
Zi2=interp2(X,Y,Z,Xi,Yi,'linear');
Zi3=interp2(X,Y,Z,Xi,Yi,'cubic');

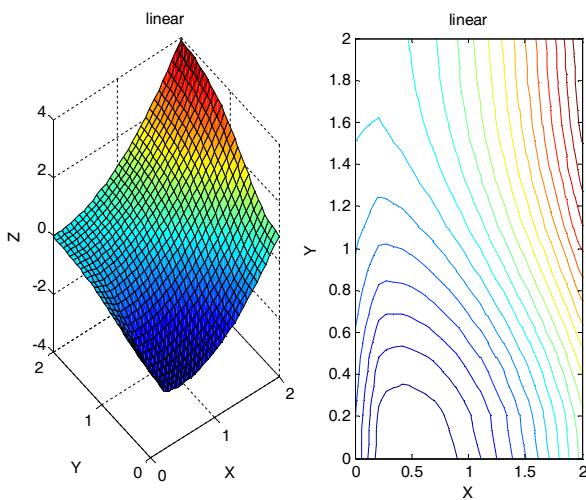
% wizualizacja wyników
figure(1)
subplot(1,2,1)
surf(Xi,Yi,Zi1)
xlabel('X'); ylabel('Y'); zlabel('Z');
title('nearest')
subplot(1,2,2)
contour(Xi,Yi,Zi1,20)
xlabel('X'); ylabel('Y'); zlabel('Z');
title('nearest')
figure(2)
subplot(1,2,1)
surf(Xi,Yi,Zi2)
xlabel('X'); ylabel('Y'); zlabel('Z');
title('linear')
subplot(1,2,2)
contour(Xi,Yi,Zi2,20)
xlabel('X'); ylabel('Y'); zlabel('Z');
title('linear')
figure(3)
subplot(1,2,1)
surf(Xi,Yi,Zi3)
xlabel('X'); ylabel('Y'); zlabel('Z');
title('cubic')
subplot(1,2,2)
contour(Xi,Yi,Zi3,20)
xlabel('X'); ylabel('Y'); zlabel('Z');
title('cubic')

```

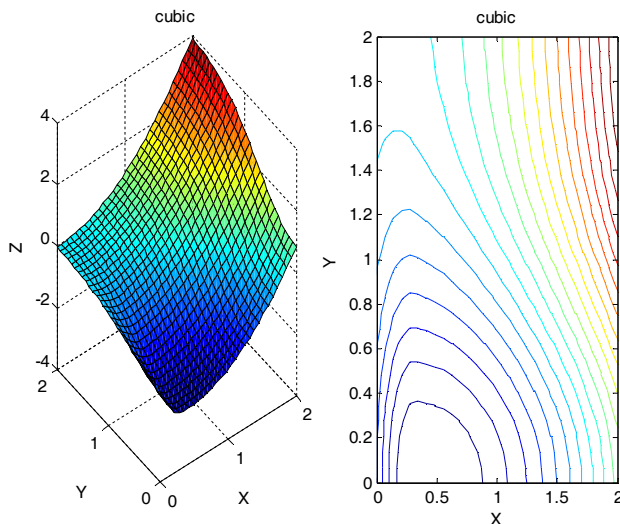

Rezultat 5.12



Rys. 5.27. Efekt interpolacji dwuwymiarowej punktów metodą 'nearest'



Rys. 5.28. Efekt interpolacji dwuwymiarowej punktów metodą 'linear'



Rys. 5.29. Efekt interpolacji dwuwymiarowej punktów metodą 'cubic'

Interpolacja danych funkcji wielu zmiennych odbywa się za pomocą `interp3`. Wywołanie funkcji jest podobne do `interp2` z tą różnicą, że do wizualizacji wyników trzeba użyć odpowiedniego wykresu `slice`.

5.5. Wygładzanie danych

Wygładzenie danych polega na takim poprowadzeniu funkcji dyskretnej przez dane punkty, aby oddawała ona charakter zmienności danych, eliminując nałożony nań szum statystyczny. Na podstawie tej funkcji otrzymuje się nowe, pozbawione szumu, wartości zmiennej zależnej dla wszystkich wartości doświadczanych zmiennej niezależnej.

W wersji podstawowej MATLABa zaleca się korzystanie w tym celu ze wspomnianego wcześniej narzędzia **Curve Fitting Tool** (wywołanie `cftool`). Dla ilustracji wygenerowano zaszumiany plik z danymi, a następnie wygładzono je interaktywnie jedną z metod narzędzia CFT. Ilustruje to przykład 5.13.

Przykład 5.13. Wygładzanie danych narzędziem CFT

 **p5_13.m**

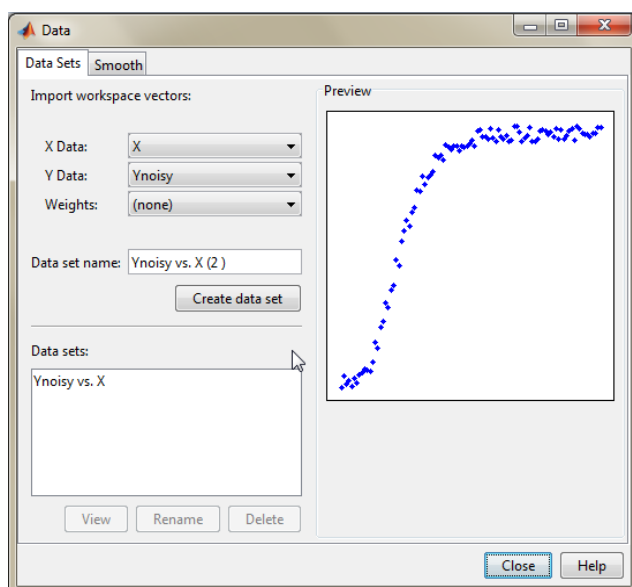
```
%Przykład 5.13 Wygładzanie danych za pomocą
%Curve Fitting Tool
```

```

clear all; clc;
%generacja zaszumionych danych
s=[0.087 0.839 2.085*10^-3 1.616];
T=20+273.15;
w1=@(X) -s(3)*T.^s(4).*X;
w2=@(X) -s(1)*T.^s(2).*exp(w1(X));
fi=@(X) exp(w2(X).*exp(w1(X)));
X=linspace(0,0.3,100);
Y=fi(X);
Ynoisy = Y + 0.07*(rand(size(X))-0.5);
%wykres zaszumionych danych
plot(X,Ynoisy,'x')
%wywołanie CFT
cftool
%tu opracujemy dane ręcznie i zapiszemy w zmiennej smoothed
%w pamięci roboczej, można teraz odczytać wektor wygładzonych
%wartości Y
Ys=smoothed.y;

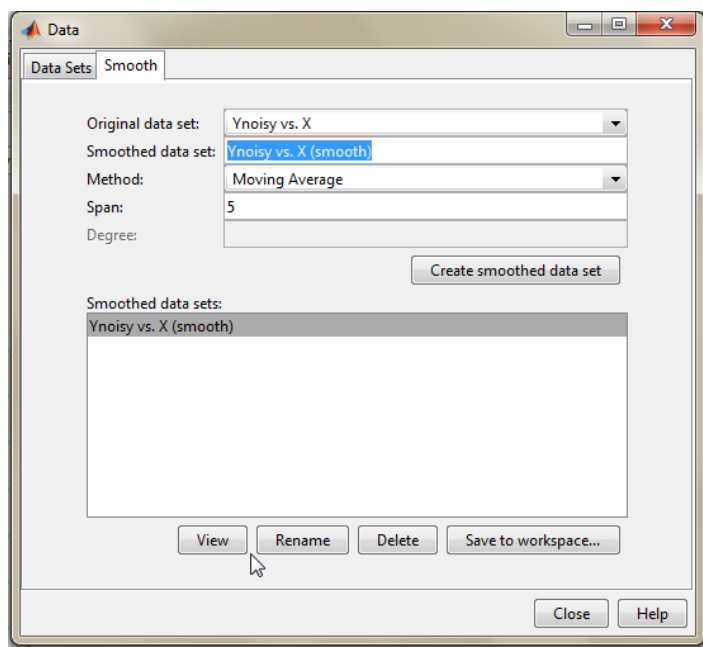
```

Rezultat 5.13



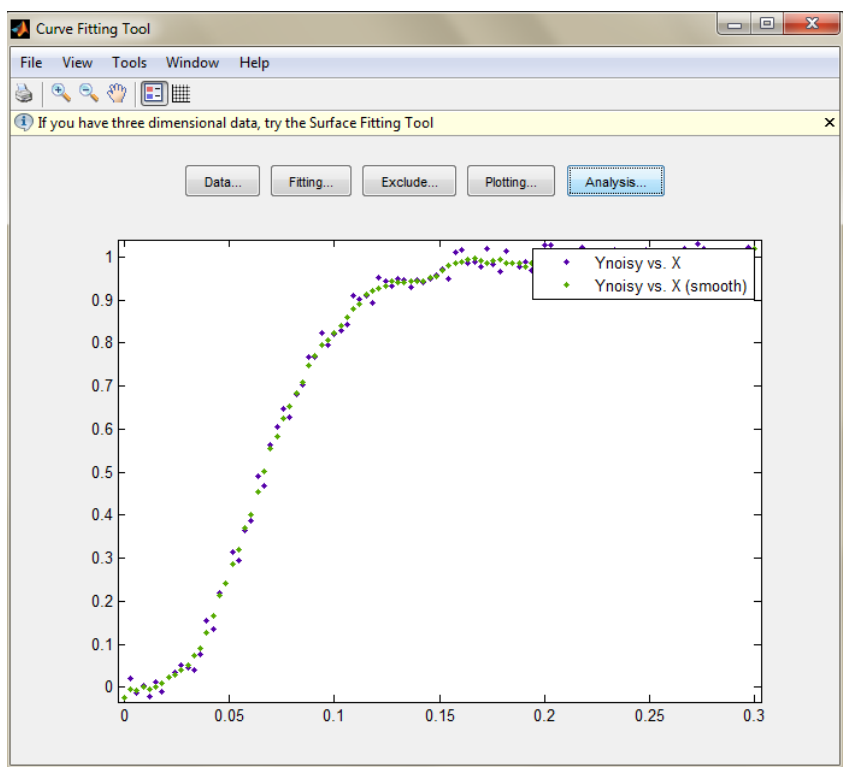
Rys. 5.30. Okno Data zakładka Data Sets – wczytanie danych

Po uruchomieniu programu w pamięci roboczej umieszcza on trzy wektory danych X, Y i Ynoisy, wykonuje wykres danych zaszumionych, a następnie wywołuje narzędzie **CFT**. Pierwszym oknem, które się otwiera jest okno główne **CFT** – Rys. 5.32, wtedy jeszcze puste. Wciskamy przycisk **Data**, pierwszy z lewej na liście kluczowych opcji **CFT**. Pojawi się wtedy okno **Data** (Rys. 5.30) zawierające dwie zakładki **Data Sets** i **Smooth**. W zakładce **Data Sets** wybieramy spośród zmiennych znajdujących się w pamięci roboczej X jako X Data i Ynoisy jako Y Data, a następnie wciskamy przycisk **Create Data Set**. W polu **Data sets**: pojawi się wtedy nazwa zestawu danych Ynoisy vs. X.



Rys. 5.31. Okno Data zakładka Smooth – wybór metody wygładzania

Otwieramy teraz zakładkę **Smooth** (Rys. 5.31) i wybieramy jako *Original data set*: zestaw Ynoisy vs. X. Jako metodę wygładzania wybieramy jedną z pozycji listy opcji – w tym przypadku *Moving Average* oraz *Span*: 5 (liczba punktów w schemacie wygładzania – im więcej tym gładzsza funkcja). Po wciśnięciu przycisku **Create smoothed data set** program utworzy wygładzony zestaw danych, i pokaże go na wykresie w oknie głównym **CFT** (Rys. 5.32).



Rys. 5.32. Wygładzony zestaw danych w głównym oknie CFT

Wygładzony zestaw danych można zapisać w pamięci roboczej jako macierz strukturalną przez wciśnięcie przycisku **Save to workspace** w oknie **Data/Smooth**. Można go stamtąd pobrać do dalszego wykorzystania. W kodzie przykładu 5.13 wygładzone wartości Y zawiera ostatnia linijka.

Wyspecjalizowane funkcje do wygładzania zaszumianych danych z poziomu programu są dostępne w skrzynce narzędziowej **Spline Toolbox**, i nie będą tu omawiane.

LITERATURA

- Balcerzak J., 2005, „Zmienność sorpcji wody przez folie chitozanowe wraz z temperaturą”, praca magisterska, Wydział Inżynierii Procesowej i Ochrony Środowiska PŁ.*
- Cady M.P., Trapp C.A., „A Mathcad Primer for Physical Chemistry”, Oxford Univ. Press, 1999, ss. 172-173.*
- Pakowski Z., DryPAK v.3.6, Program for drying and psychrometric calculations, Wyd. OMNIKON, Łódź 2000.*
- Pakowski, Z., B. Krupińska and R. Adamski, 2005, Water activity of energetic variety of wood in a wide temperature range applicable to both superheated steam and air drying, 7th World Congress of Chemical Engineering, Glasgow.*

Oznaczając macierz współczynników przez A , wektor niewiadomych przez x , a wektor wyrazów wolnych przez b , otrzymamy równanie:

$$A \cdot x = b. \quad (6.4)$$

Macierzą rozszerzoną U tego układu nazywamy macierz utworzoną przez dodanie wektora b jako ostatniej kolumny do macierzy współczynników A .

Warunek rozwiązywalności tego układu określa twierdzenie Kroneckera-Capellego:

- Jeśli rząd r macierzy A i rząd s macierzy U są równe i równają się liczbie niewiadomych N , to układ jest **oznaczony** i ma jedno rozwiązanie.
- Jeśli $r = s < N$ to układ jest **nieoznaczony** i część równań jest liniowo zależna od pozostałych. Należy je z układu wyeliminować, by doprowadzić do układu oznaczonego bądź wyznaczyć rozwiązanie jako funkcję wybranych $(N-r)$ parametrów. Ten drugi sposób pokazano w Projekcie 6.4.
- Jeśli występuje sytuacja, że $r = s > N$, to układ jest **nadokreślony**. Ten szczególny przypadek zilustrowano przykładem 6.6.
- Jeśli $r < s$ to układ jest **sprzeczny** i nie ma rozwiązań.

Przypominamy, że rząd macierzy A obliczamy poleceniem $\text{rank}(A)$.

W przypadku układu oznaczonego, jeśli macierz współczynników A jest nieosobliwa ($\det(A) \neq 0$), istnieje jedno rozwiązanie tego układu. Można je wyznaczyć przez proste przekształcenie układu do postaci:

$$x = \frac{b}{A} \quad \text{lub} \quad x = A^{-1} \cdot b, \quad (6.5)$$

gdzie A^{-1} oznacza macierz odwrotną.

Układy równań liniowych rozwiązuje się metodami ścisłymi (wzory Cramera, przez odwracanie macierzy, metodą Gaussa lub Gaussa-Jordana, metodą dekompozycji LU, metodą dekompozycji Cholesky'ego) lub iteracyjnymi (np. metoda Seidela, metoda relaksacji).

6.1.1. Układ równań dobrze i źle uwarunkowany

Układ równań, w którym minimalna zmiana współczynników wywołuje wielkie zmiany w wektorze rozwiązania, nazywa się źle uwarunkowanym.

W zależności od wskaźnika uwarunkowania układu równań dobiera się odpowiednią metodę rozwiązywania. Jeśli układ równań jest źle uwarunkowany i zostanie rozwiązany nieodpowiednią metodą, wtedy dokładność rozwiązania może być znacznie obniżona. Do oszacowania wskaźnika uwarunkowania

dla układu równań liniowych w MATLABie używa się funkcji `rcond(A)`. Dla dobrze uwarunkowanego układu wartość `rcond(A)` jest bliska 1. Wartości tego współczynnika bliskie zero świadczą o złym uwarunkowaniu, np. dla `rcond=1.00000e-003` rozwiązanie uzyskane w wyniku obliczeń macierzowych może nie być wiarygodne.

Dla ilustracji rozwiążmy układ równań postaci:

$$\begin{aligned} 1x_1 + 2x_2 + 3x_3 &= 2 \\ 4x_1 + 5x_2 + 6x_3 &= 2 \\ 7x_1 + 8x_2 + 9x_3 &= 2 \end{aligned} \quad (6.6)$$

Definiujemy macierz A współczynników:

```
A=[ 1 2 3
    4 5 6
    7 8 9 ];
```

Definiujemy wektor b wyrazów wolnych:

```
b=[2; 2; 2];
```

Obliczamy wektor niewiadomych za pomocą dzielenia macierzowego:

```
x1=A\b
```

W wyniku otrzymamy:

```
Warning: Matrix is close to singular or badly scaled. Results
may be inaccurate. RCOND = 1.541976e-018.
```

```
x1 =
    -5
     8
    -3
```

Obliczamy wektor niewiadomych za pomocą odwracania macierzy.

```
x2=inv(A)*b
```

W wyniku otrzymamy:

```
Warning: Matrix is close to singular or badly scaled. Results
may be inaccurate. RCOND = 1.541976e-018.
```

```
x2 =
    -5
    10
    -3
```

Jak widzimy, rezultat działań obu instrukcji jest różny, co wynika z tego, że układ równań jest źle uwarunkowany. W takim przypadku trzeba zbadać

własności układu równań. W tym celu sprawdzamy wyznacznik macierzy i wskaźnik uwarunkowania odwrotności macierzy⁴ $\mathbf{A}$.

```
wyznacznik=det(A)
wskaźnik=rcond(A)
```

W rezultacie otrzymamy:

```
wyznacznik = 0
wskaźnik = 1.541976e-018
```

Wyznacznik jest równy zero, wskaźnik jest bardzo mały, więc rozwiązania takiego układu równań ww. metodami nie możemy być pewni. Dokonajmy sprawdzenia. Sprawdzamy residua w obu powyższych rozwiązaniach:

```
r1=b-A*x1
r2=b-A*x2
```

Otrzymujemy:

```
r1 =
    0
    0
    0
r2 =
   -4
  -10
  -16
```

Widzimy, że residuum wynikające z rozwiązania r_1 jest równe zero, możemy więc przyjąć, że to rozwiązanie jest dokładne. Dla pewności jednak lepiej rozwiązać taki układ równań inną metodą. Dla równań źle uwarunkowanych zaleca się stosowanie metod opartych o faktoryzację np.:

- macierz ortogonalną i macierz trójkątną QR,
- macierze trójkątne LU,
- macierz trójkątną górną R – dekompozycję Cholesky’ego.

Powyższe metody będą przedstawione w Projekcie 6.1, choć układ tam występujący jest dobrze uwarunkowany.

⁴ Można też sprawdzać wskaźnik uwarunkowania $\text{cond}(\mathbf{A})$, który jest bardzo duży dla macierzy źle uwarunkowanej.

PROJEKT 6.1

Temat: Przeliczanie stężeń

Toolbox: -

W 1 kg wody należy rozpuścić 5 substancji podanych poniżej tak, aby otrzymać roztwór o zadanych ułamkach molowych. Ile gramów każdej substancji należy odważyć? (*Pakowski, Głębowski, 2001*)

Substancje	LiCl	NaCl	KCl	MgCl ₂	CaCl ₂
Ułamki molowe	0.052	0.098	0.042	0.100	0.070
składników [-]					
Masy molowe	43.394	58.442	74.551	95.21	111.0
substratów					
[kg/kmol]					

Ułamek molowy zdefiniowany jest wzorem:

$$x_i = \frac{n_i}{\sum_{j=1}^N n_j} = \frac{\frac{m_i}{M_i}}{\sum_{j=1}^{N-1} \frac{m_j}{M_j} + \frac{m_w}{M_w}} \quad (6.7)$$

Wprowadźmy oznaczenie

$$\mu_i = \frac{M_w}{M_i} \quad (6.8)$$

Otrzymamy

$$x_i = \frac{m_i \mu_i}{\sum_{j=1}^{N-1} m_j \mu_j + m_w} \quad (6.9)$$

co daje następujący układ równań

$$\begin{aligned}
x_1\mu_1m_1 + x_1\mu_2m_2 + x_1\mu_3m_3 + x_1\mu_4m_4 + x_1\mu_5m_5 + x_1m_w &= \mu_1m_1 \\
x_2\mu_1m_1 + x_2\mu_2m_2 + x_2\mu_3m_3 + x_2\mu_4m_4 + x_2\mu_5m_5 + x_2m_w &= \mu_2m_2 \\
x_3\mu_1m_1 + x_3\mu_2m_2 + x_3\mu_3m_3 + x_3\mu_4m_4 + x_3\mu_5m_5 + x_3m_w &= \mu_3m_3 \\
x_4\mu_1m_1 + x_4\mu_2m_2 + x_4\mu_3m_3 + x_4\mu_4m_4 + x_4\mu_5m_5 + x_4m_w &= \mu_4m_4 \\
x_5\mu_1m_1 + x_5\mu_2m_2 + x_5\mu_3m_3 + x_5\mu_4m_4 + x_5\mu_5m_5 + x_5m_w &= \mu_5m_5
\end{aligned} \tag{6.10}$$

Przekształcimy ten układ równań tak, aby wszystkie wyrazy zawierające niewiadomą znalazły się po lewej stronie, a wszystkie wyrazy wolne po prawej.

$$\begin{aligned}
-\mu_1(1-x_1)m_1 + x_1\mu_2m_2 + x_1\mu_3m_3 + x_1\mu_4m_4 + x_1\mu_5m_5 &= -x_1m_w \\
x_2\mu_1m_1 - \mu_2(1-x_2)m_2 + x_2\mu_3m_3 + x_2\mu_4m_4 + x_2\mu_5m_5 &= -x_2m_w \\
x_3\mu_1m_1 + x_3\mu_2m_2 - \mu_3(1-x_3)m_3 + x_3\mu_4m_4 + x_3\mu_5m_5 &= -x_3m_w \\
x_4\mu_1m_1 + x_4\mu_2m_2 + x_4\mu_3m_3 - \mu_4(1-x_4)m_4 + x_4\mu_5m_5 &= -x_4m_w \\
x_5\mu_1m_1 + x_5\mu_2m_2 + x_5\mu_3m_3 + x_5\mu_4m_4 - \mu_5(1-x_5)m_5 &= -x_5m_w
\end{aligned} \tag{6.11}$$

Pomnożymy jeszcze ten układ równań obustronnie przez -1. Pokażemy teraz różne sposoby jego rozwiązania. Niewiadomą jest wektor mas $[m_1; m_2; m_3; m_4; m_5]$.

Rozwiązania mieszczą się w czterech plikach **C.m**, **C_QR.m**, **C_LU.m**, **C_ChB.m**.

- Rozwiązanie układu równań przez odwracanie macierzy

C.m

```

%Projekt 6.1
%Rozwiązanie liniowego układu równań
clear all;
disp(' Zadanie')
disp(' W 1 kg wody należy rozpuścić substancje podane poniżej
tak,')
disp(' aby otrzymać roztwór o zadanych ułamkach molowych. ')
disp(' Ile gramów każdej substancji należy odważyć ?')

%      LiCl      NaCl      KCl  MgCl2  CaCl2
M=[43.394;  58.442;  74.551;  95.21;  111.0];      % masy molowe
składników [kg/kmol]
x=[ 0.052;   0.098;   0.042;  0.100;  0.070];      % ułamki molowe
składników

```

```

Mw=18.01; % masa molowa H2O [kg/kmol]
mw=1; % masa uzytej wody [kg]
ni=Mw./M; % liczba moli danego skladnika
M=-x*ni';
D=(1-x).*ni;
for i=1:5
    M(i,i)=D(i,1);
End

%-----rozwiazywanie ukkladu-----
m=inv(M)*x*mw;
%-----
disp('Odpowiedz w [kg]')
disp('LiCl='), disp(m(1,1))
disp('NaCl='), disp(m(2,1))
disp('KCl='), disp(m(3,1))
disp('MgCl2='),disp(m(4,1))
disp('CaCl2='),disp(m(5,1))

```

Wyniki 6.1a

```

Odpowiedz w [kg]
LiCl  = 0.1964
NaCl  = 0.4984
KCl   = 0.2725
MgCl2 = 0.8286
CaCl2 = 0.6762

```

- Rozwiązanie układu równań metodą rozkładu na iloczyn dwóch macierzy: ortogonalnej Q i górnej trójkątnej R .

Macierz ortogonalna to macierz kwadratowa spełniająca równość:

$$Q^T \cdot Q = I, \quad (6.12)$$

gdzie A^T – jest macierzą transponowaną względem A , I – jest macierzą jednostkową. Macierz współczynników przekształca się do postaci

$$A = Q \cdot R. \quad (6.13)$$

Rozwiązanie układu równań ma więc postać

$$x = R \setminus (Q \setminus b). \quad (6.14)$$

📁 **C_QR.m** – pokazana tylko część dot. rozwiązywania układu równań.

```
% sprawdzenie czy układ jest źle uwarunkowany
```

```
wyznacznik=det(M)
```

```
warunek=cond(M)
```

```
% dekompozycja macierzy
```

```
[Q,R]=qr(M);
```

```
% wyliczamy wektor b;
```

```
b=x*mw;
```

```
% wyliczenie elementów wektora rozwiązań
```

```
m=R\ (Q\b);
```

Wyniki 6.1b

```
wyznacznik = 6.0503e-004
```

```
warunek = 3.1599
```

```
Odpowiedź w [kg]
```

```
LiCl = 0.1964
```

```
NaCl = 0.4984
```

```
KCl = 0.2725
```

```
MgCl2 = 0.8286
```

```
CaCl2 = 0.6762
```

- Rozwiązanie układu równań z zastosowaniem rozkładu LU

Metoda rozkładu LU polega na zapisaniu macierzy współczynników A jako iloczynu dwóch macierzy trójkątnych⁵: dolnej L i górnej U .

$$A = L \cdot U \quad (6.15)$$

Rozwiązanie układu równań ma więc postać

$$x = U \setminus (L \setminus b) \quad (6.16)$$

📁 **C_LU.m** – pokazana tylko część dot. rozwiązywania układu równań.

```
% rozkład macierzy M na macierze trójkątne L oraz U
```

```
% oraz obliczenie elementów wektora m 'masy'
```

```
[L,U]=lu(M)
```

```
y=L\ (x*mw);
```

```
m=U\y;
```

⁵ W macierzy trójkątnej nad (lub pod) przekątną są tylko elementy zerowe.

Wyniki są następujące:

L =	1.0000	0	0	0	0
	-0.1034	1.0000	0	0	0
	-0.0443	-0.0494	1.0000	0	0
	-0.1055	-0.1176	-0.1238	1.0000	0
	-0.0738	-0.0824	-0.0866	-0.0989	1.0000
U =	0.3935	-0.0160	-0.0126	-0.0098	-0.0084
	0	0.2763	-0.0250	-0.0196	-0.0168
	0	0	0.2296	-0.0093	-0.0080
	0	0	0	0.1657	-0.0201
	0	0	0	0	0.1462

Wyniki 6.1c

Odpowiedź w [kg]

LiCl = 0.1964

NaCl = 0.4984

KCl = 0.2725

MgCl₂ = 0.8286

CaCl₂ = 0.6762

- Rozwiązanie układu równań z zastosowaniem rozkładu Cholesky'ego

Metoda Cholesky'ego polega na zapisaniu macierzy współczynników za pomocą takiej macierzy trójkątnej górnej R, że zachodzi równość:

$$A = R' R \quad (6.17)$$

Wtedy rozwiązanie układu równań można przedstawić jako

$$x = R \backslash (R' b) \quad (6.18)$$

Rozkład Cholesky'ego działa tylko dla macierzy dodatnio określonych, których elementy diagonalne muszą być dodatnie, a pozostałe niezbyt duże.

 **C_ChB.m**

% macierz górna trójkątna wg schematu Choleskiego

R=chol(M)

% transpozycja macierzy r

G=R';

% rozwiązanie pierwszego układu


```
y=G\'(x*mw);
% rozwiązanie ostatecznie układu
m=R\y;
```

Wyniki 6.1d

R =	0.6273	-0.0255	-0.0200	-0.0157	-0.0135
	0	0.5266	-0.0459	-0.0360	-0.0308
	0	0	0.4785	-0.0207	-0.0178
	0	0	0	0.4102	-0.0437
	0	0	0	0	0.3841

Odpowiedź w [kg]

LiCl = 0.1964

NaCl = 0.4984

KCl = 0.2725

MgCl2 = 0.8286

CaCl2 = 0.6762

6.2. Równania i układy równań nieliniowych

Znacznie częściej niż liniowe występują w praktyce równania i układy równań nieliniowych. Ścisłe metody macierzowe są wtedy nieodpowiednie, a stosowane są metody numeryczne, z natury przybliżone. Istnieje wiele metod numerycznych rozwiązywania równań nieliniowych jak np. metoda Newtona, oraz układów równań jak np. metoda Newtona-Raphsona. MATLAB ma zaimplementowane bardziej efektywne i sprawne metody numeryczne wbudowane w tzw. solvery (narzędzia rozwiązujące) numeryczne.

6.2.1. Rozwiązywanie równań i układów równań algebraicznych z pomocą solverów MATLABa

Solvery to narzędzia numeryczne w postaci funkcji do rozwiązywania równań algebraicznych i ich układów. W MATLABie znajdziemy tylko dwie funkcje do rozwiązywania równań algebraicznych liniowych i nieliniowych. Poniżej pokazano ich wywołania w wersji uproszczonej

`x=fzero(@fun,x0)` polecenie oblicza miejsca zerowe x zadanej funkcji fun , rozpoczynając obliczenia od miejsca określonego przez $x0$, $@fun$ to nazwa zadanej funkcji jednej zmiennej. Uwaga: $x0$ może mieć również postać przedziału (wektor dwuelementowy), w którym funkcja zmienia znak.

`x=root(p)` funkcja wyznacza wszystkie pierwiastki wielomianu `p` podanego jako wektor współczynników.

Niestety narzędzie do rozwiązywania układów równań znajduje się w skrzynce narzędziowej **Optimization Toolbox** i ma postać podaną poniżej:

`x=fsolve(@fun,x0)` polecenie rozwiązuje układy równań nieliniowych, `x0` – punkt startowy, `@fun` – nazwa funkcji, w której zdefiniowany jest układ równań.

6.2.1.1. Rozwiązanie pojedynczego równania algebraicznego

- Miejsce zerowe funkcji liniowej:

$$y = 3x - 5. \quad (6.19)$$

Przykład 6.1

Definiujemy funkcję, której miejsca zerowego poszukujemy, i zapisujemy w pliku **f1.m**.

```
function y = f1(x)
y = 3.*x-5;
```

Definiujemy skrypt, w którym wywołamy solver obliczający miejsca zerowe funkcji.

p6_1.m

```
%Przykład 6.1 Rozwiązanie równania liniowego
%definiujemy punkt startowy.
x0=[2];
%wywołujemy solver
r1 = fzero(@f1,x0)
```

Rezultat 6.1

```
r1 = 1.6667e+000
```

- Miejsca zerowe wielomianu:

$$y = 2x^4 + 2x^2 - 5. \quad (6.20)$$

Przykład 6.2

Definiujemy skrypt, w którym wywołamy solver `fzero` i `roots`.

p6_2.m

```
%Przykład 6.2 Pierwiastki wielomianu
%definiujemy funkcję
f2=@(x) 2.*x.^4+2.*x.^2-5;
%określamy punkt startowy.
x0=[2];
%wywołujemy solver
r1=fzero(f2,x0)
%to samo przez roots
r2=roots([2 0 2 0 -5])
```

Rezultat 6.2

```
r1 =
    1.0762
r2 =
    0.0000 + 1.4691i
    0.0000 - 1.4691i
   -1.0762
    1.0762
```

Jak widać wywołanie solvera `fzero` znajduje tylko jeden pierwiastek położony w pobliżu punktu startowego, natomiast funkcja `roots` znajduje wszystkie pierwiastki wielomianu.

- Rozwiążmy równanie przestępne (transcendentalne⁶):

$$y = e^x + x. \quad (6.21)$$

Przykład 6.3

Definiujemy funkcję, której miejsca zerowego poszukujemy, i zapisujemy w pliku **f3.m**.

f3.m

```
function y = f3(x)
y = exp(x)+x;
```

⁶ Równanie przestępne – równanie zawierające funkcje niealgebraiczne takie jak np. $\exp$, $\log$, $\sin$ itd. Często równań takich nie można rozwiązać analitycznie.

Definiujemy skrypt, w którym wywołamy solver obliczający miejsca zerowe funkcji. Dodatkowo wykonamy wykres funkcji wraz z rozwiązaniem.

p6_3.m

%Przykład 6.3 Rozwiązanie równania przestępnego

```
clear all;
```

```
%definiujemy punkt startowy.
```

```
x0=[20];
```

```
%wywołujemy solver.
```

```
r3 = fzero(@f3,x0)
```

```
%prezentacja graficzna rozwiązania.
```

```
x=-2:.1:2;
```

```
y=f3(x);
```

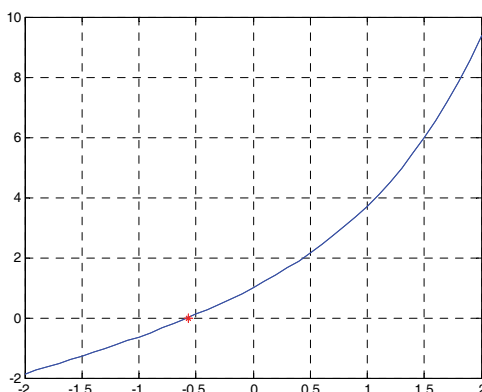
```
plot(x,y)
```

```
grid on
```

```
hold on
```

```
plot(r3,f3(r3),'r*')
```

Rezultat 6.3



Rys. 6.1. Wykres równania 6.24 z naniesionym rozwiązaniem

- Miejsca zerowe funkcji okresowej

$$y = x \sin(x^2). \quad (6.22)$$

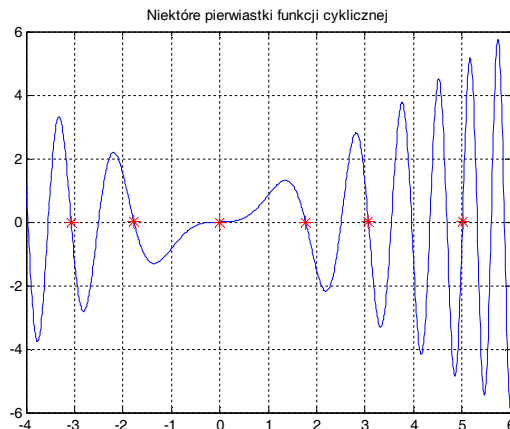
Przykład 6.4

Definiujemy skrypt, w którym wywołamy solver obliczający miejsca zerowe funkcji, i wykonamy ilustrację graficzną. Użyjemy różnych punktów startowych.

p6_4.m

```
%Przykład 6.4 Miejsca zerowe funkcji okresowej  
%wywołanie solverów z różnymi punktami startowymi.  
clear all; clc, close all;  
f4=@(x) x.*sin(x.^2);  
r1 = fzero(f4,-3); r2 = fzero(f4,-1); r3 = fzero(f4,0);  
r4 = fzero(f4,2); r5 = fzero(f4,3); r6 = fzero(f4,5);  
r=[r1 r2 r3 r4 r5 r6];  
%prezentacja graficzna wyników  
x=-4:.01:6;  
y=f4(x);  
plot(x,y)  
grid on  
hold on  
plot(r,f4(r),'r*','MarkerSize',9)  
title('Niektóre pierwiastki funkcji cyklicznej')
```

Rezultat 6.4



Rys. 6.2. Wykres równania 6.22 ze znalezionymi pierwiastkami dla różnych punktów startowych

Widzimy, że ta funkcja posiada wiele miejsc zerowych. Aby znaleźć właściwy pierwiastek, trzeba odpowiednio dobrać punkt startowy. Najlepiej znaleziony wynik ocenić graficznie, wykonując wykres.

6.2.1.2. Rozwiązywanie układów równań

Przykład 6.5

Rozwiążmy układ równań postaci:

$$y = \begin{bmatrix} x_1^3 - 2x_2 + 1 \\ x_3 \exp(x_1) + x_2 \\ -2x_3 + 1 \\ x_2^2 \sqrt{x_1} \sin(x_4) \end{bmatrix} \quad (6.23)$$

Do rozwiązania zastosujemy polecenie **fsolve**, służy ono do rozwiązywania układów równań nieliniowych. Pełne wywołania solverów **fzero** i **fsolve** mają postać

```
[x,fval,exitflag,output] =fzero(@fun,x0,options)
[x,fval,exitflag,output,jacobian]=fsolve(@fun,x0,options)
```

gdzie	
fval	wartość funkcji w punkcie zerowym
exitflag	wyświetla warunek ukończenia obliczeń
output	wyświetla dane o przebiegu optymalizacji
jacobian	wyświetla wartość Jakobianu w punkcie rozwiązania
options	opcje rozwiązania takie jak wybór algorytmu, tolerancji wartości funkcji w miejscu zerowym, tolerancji wartości pierwiastka etc. Opcje definiuje się za pomocą funkcji optimiset (zob. Pomoc).

W tym przykładzie wykorzystamy opcje wywołania, aby móc śledzić, jak solver dąży do rozwiązania i czy jest ono poprawne.

Definiujemy układ równań, którego rozwiązania poszukujemy.

 **f5.m**

```
function y=f5(x)
y=[ x(1).^3-2.*x(2)+1
    x(3).*exp(x(1))+x(2)
    -2.*x(3)+1
    x(2).^2.*sqrt(x(1)).*sin(x(4))];
```

Definiujemy skrypt, w którym wywołamy solver rozwiązujący układ równań nieliniowych

p6_5.m

```
%Przykład 6.5 Rozwiązanie układu równań nieliniowych
clear all; close all; clc
%definiujemy punkty startowe możliwie blisko
%oczekiwanego rozwiązania.
x0=[1 1 1 1];
%wywołanie solvera wraz z opcjami.
options=optimset('Display','iter');
x = [fsolve(@f5,x0,options)]'
```

Rezultat 6.5

Iteration	Func-count	f(x)	Norm of step	First-order optimality	Trust-region radius
0	5	15.5337		12.1	1
1	10	1.61	1	1.67	1
2	15	0.760671	2.5	0.345	2.5
3	20	0.368551	2.5	0.697	2.5
4	21	0.368551	0.788016	0.697	6.25
5	26	0.202142	0.197004	0.42	0.197
6	27	0.202142	0.49251	0.42	0.493
7	32	0.148632	0.123127	0.483	0.123
8	37	0.0133113	0.307819	0.174	0.308
9	42	0.000197996	0.524849	0.0317	0.77
10	47	4.39515e-007	0.415711	5.78e-005	1.31
11	52	1.49505e-014	0.0228356	3.55e-009	1.31

Equation solved.

fsolve completed because the vector of function values is near zero as measured by the default value of the function tolerance, and the problem appears regular as measured by the gradient.

<stopping criteria details>

```
x =
-1.1005
-0.1664
0.5000
6.2832
```

Przykład wykorzystania solvera `fsolve` do układu równań nieliniowych przedstawia Projekt 6.2.

PROJEKT 6.2

Temat: Rozwiązanie macierzowe równania dyfuzji wieloskładnikowej GSM. Strumienie masy znajdowane przez rozwiązanie równania uwikłanego

Toolbox: **Optimization**

Wykorzystano tu Projekt 3.1, który uzupełniono o funkcję będącą różnicą założonych i obliczonych strumieni molowych składników i jej solver **fsolve**. Znalezienie miejsca zerowego tej funkcji daje w wyniku poszukiwany wektor strumieni molowych. Dane są identyczne jak w Projekcie 3.1. Funkcje, które się powtarzają w projekcie 3.1, zamknięto w Edytorze (wyświetlają się tylko nagłówki). W tym celu wybrano opcję *Code Folding* (opcje *Fold/Expand*) w oknie, które pojawia się po kliknięciu prawym klawiszem myszy na słowie **function**.

P6_2.m

%PROJEKT 6.2

%Rozwiązanie macierzowe równania dyfuzji wieloskładnikowej GSM
%Strumienie masy znajdowane przez rozwiązanie równania uwikłanego

function SMSolve3

clear all; close all; clc;

global C L xd x0 n bet0 k0 D0 D N3 Dane N

%składnik 1 - aceton

%składnik 2 - metanol

%składnik 3 - powietrze

%Dane

P0=99.4; %ciśnienie ogólne, kPa

T=328.5; %temperatura, K

L=0.238; %długość drogi dyfuzji,m

R=8.314; %uniwersalna stała gazowa, kJ/(kmol K)

n=3; %liczba składników;

C=P0/(R*T); %stężenie ogólne, kmol/m3

%Eksperymentalne profile stężeń

x01=0.319;

x02=0.528;

x03=1-x01-x02;

l=0;


```

Dane=[1 x01 x02 x03;
      0.047 0.265 0.463 0.221;
      0.072 0.250 0.456 0.279;
      0.094 0.235 0.426 0.338;
      0.120 0.221 0.382 0.412;
      0.146 0.184 0.316 0.515;
      0.173 0.135 0.250 0.603;
      0.190 0.125 0.162 0.721;
      0.238 0 0 1;];

%Macierz dyfuzyjności binarnych SM
D12=8.48e-6;    D21=D12;
D13=13.72e-6;  D31=D13;
D23=19.91e-6;  D32=D23;

D=[1 D12 D13;
   D21 1 D23;
   D31 D32 1;];

x0=[x01;x02];      %wektor ułamków molowych na lewym krańcu
xd=[0;0];           %wektor ułamków molowych na prawym krańcu
lamb=[0;0;1];       %współczynniki macierzy bootstrapowej

D0=(B(D,x0))^-1;
k0=D0/L;
bet0=beta(lamb,x0);

N0=[1e-6; 1e-6];   %założony wektor strumieni składników N1 i N2.
N3=0;              % N3=0 - inert

N=fsolve(@fun,N0);

disp('strumień molowy składnika 1'); disp(N(1));
disp('strumień molowy składnika 2'); disp(N(2));

wykres;

function y=fun(N) %obliczony wektor różnic strumieni zał. i obl.
global bet0 k0 x0 xd C D n N3
NN=N;
NN(n)=N3;
Fi=Fif(D,NN);

```

```

I=eye(n-1);
Sig0=Fi*(expm(Fi)-I)^-1;
y=C*bet0*k0*Sig0*(x0-xd)-N;

function wykres

function w=B(D,x) %oblicza odwrotność macierzy dyfuzyjności SM

function w=Fif(D,N) %oblicza macierz strumieni N to wektor[1xn]

function w=beta(lam,x) % oblicza macierz bootstrapową

function y=xmol(x,Fi) %oblicza ułamki molowe

```

Wyniki 6.2

```

strumień molowy składnika 1
1.7830e-006

strumień molowy składnika 2
3.1270e-006

```

Zauważ, że założone wartości wynosiły odpowiednio 1e-6 i 1e-6 kmol/(m² s). Program rysuje również wykres identyczny jak w Projekcie 3.1.

PROJEKT 6.3

Temat: Przepływ przez złożo o zmiennej grubości
Toolbox: **Optimization**

Złożo materiału ziarnistego o maksymalnej grubości H [m] spoczywa na półkolistej przegrodzie sitowej o średnicy D [m] i długości L [m], jak pokazano na Rys. 6.3. Pod złożo dopływa V [m³/s] powietrza. Obliczyć rozkład prędkości powietrza w złożu oraz spadek ciśnienia na złożu. Porowatość złoża jest zależna od prędkości.

Spadek ciśnienia w złożu opisuje równanie Erguna postaci

$$\frac{\Delta p}{h} = 150u \mu_g \left(\frac{1-\varepsilon}{d_p} \right)^2 \frac{1}{\varepsilon^3} + 1.75u^2 \rho_g \left(\frac{1-\varepsilon}{d_p} \right) \frac{1}{\varepsilon^3}, \quad (6.24)$$

gdzie Δp – spadek ciśnienia [Pa] u – prędkość [m/s], μ_g – lepkość powietrza [Pa s], ρ_g – gęstość powietrza [kg/m³], d_p – średnica cząstki [m], h – wysokość warstwy [m].

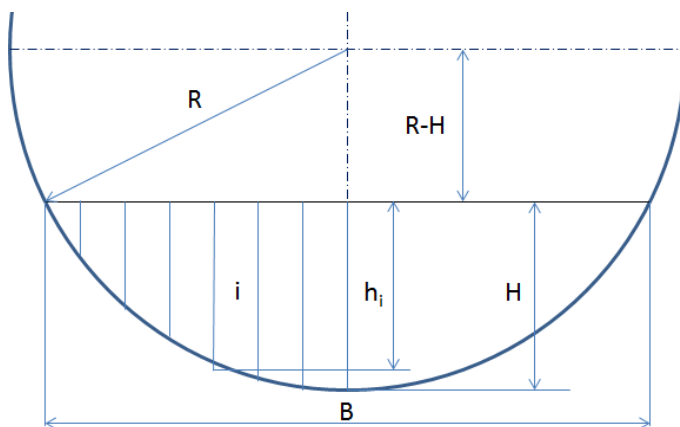
Porowatość jest stała, gdy złożo jest nieruchome bądź zależy od prędkości jeśli złożo fluidyzuje. Przyjmujemy, że graniczną wartością porowatości złoża nieruchomego jest 0.45. Zależność porowatości od prędkości dla złoża fluidalnego opisuje równanie Todesa (*Pakowski, 2011*)

$$Re_p = \frac{Ar \varepsilon^{4.75}}{18 + 0.61 \sqrt{Ar \varepsilon^{4.75}}}, \quad (6.25)$$

gdzie

$$Re_p = \frac{u d_p \rho_g}{\mu_g} \quad (6.26)$$

$$Ar = \frac{d_p^3 \rho_g (\rho_p - \rho_g) g}{\mu_g^2}. \quad (6.27)$$



Rys. 6.3. Schemat warstwy o przekroju odcinka koła

Geometrię i oznaczenia pokazano na Rys. 6.3. W celu eliminacji konieczności traktowania złoża jako obiektu o parametrach rozłożonych (lokalna wysokość warstwy jest ciągłą funkcją jej szerokości) zdyskretyzowano złożo za pomocą N jednostkowych warstw o skokowo zmiennej wysokości i szerokości ΔB .

W efekcie równania na spadek ciśnienia dla każdej warstwy mają postać

$$P_1 - P_0 = f(u_i, h_i), \quad (6.28)$$

gdzie i jest numerem warstwy (1..N), P_1 oznacza ciśnienie pod złożem – jednakowe dla wszystkich warstw, a P_0 to ciśnienie atmosferyczne. Ponieważ układ N równań zawiera tylko N niewiadomych, których jest N+1, należy dodać jeszcze jedno równanie. Jest nim równanie bilansu strumieni

$$V = L\Delta B \sum_{i=1}^N u_i. \quad (6.29)$$

P6_3.m

```
%PROJEKT 6.3 Rozkład prędkości w warstwie materiału porowatego
%o przekroju odcinka koła
clear all
clc
global dp rop V P0 g N L dB
%dane
D=1.56;           %średnica bębna, m
L=1.0;            %długość sekcji, m
H=0.4;            %głębokość złoża w centrum, m
dp=2e-3;          %średnica cząstki, m
rop=2500;         %gęstość cząstki, kg/m3
V=1;              %natężenie objętościowe przepływu, m3/s na 1 m
długości
tg=200;           %temperatura powietrza, C
Y=0.001;          %wilgotność powietrza, kg/kg
P0=101.325e3;     %ciśnienie atmosferyczne, Pa
g=9.81;           %przyspieszenie ziemskie, m/s2

%obliczenia
N=10;
R=D/2;
W=R-H;
B=2*sqrt(R.^2-W.^2);
dB=B/N;
%zależność h od B
bi=linspace(0,B,N+1);
hi=-W+sqrt(R.^2-(B/2-bi).^2);
for i=1:10
    h(i)=(hi(i)+hi(i+1))/2;
```

```

end;
figure(1);
plot(bi,hi);
xlabel('B, m'); ylabel('h, m');
%punkty startowe
u=linspace(0,N,N);
u(:)=1;
P1=P0+5000; %Pa
z0=[u';P1];
%rozwiązanie układu równań
fs=@(z0) uk_row(z0(1:N),z0(N+1),h,V,tg,Y);
opt=optimset('Display','off');
z=fsolve(fs,z0,opt);
%wizualizacja i wydruk wyników
u=z(1:N);
DP=z(N+1)-P0 % nadciśnienie w komorze dolnej
DV=sum(u.*dB)*L-V % sprawdzenie natężenia przepływu
figure(2);
b=linspace(dB/2,B-dB/2,10); % środki przedziałów
bar(b,u,'g');
ylabel('prędkość, m/s'); xlabel('odległość, m');

```

Układ równań zawarty jest w pliku **uk_row.m**, pokazanym poniżej

uk_row.m

```

%Układ równań
function y=uk_row(u,P1,h,V,tg,Y)
global dp L P0 N dB
mig=gvisc(tg,Y);
rog=gdens(tg,Y,P1);
deltapL=P1-P0;
ep=void(u,tg,Y);
for i=1:1:N
    ep0=((1-ep(i))./dp);
    ep3=ep(i).^3;
    ep1=ep0./ep3;
    ep2=ep0.^2./ep3;
    deltapR(i)=(150*u(i)*mig*ep2+1.75*u(i).^2*rog*ep1)*h(i);
    deltap(i)=deltapR(i)-deltapL;
end;

%różnice lewych i prawych stron odpowiednich równań
deltap=deltap; %na spadek ciśnienia

```

```
dArea=dB*L;
deltaV=sum(u*dArea)-V; %bilansu strumieni
y=[deltap;deltaV];
```

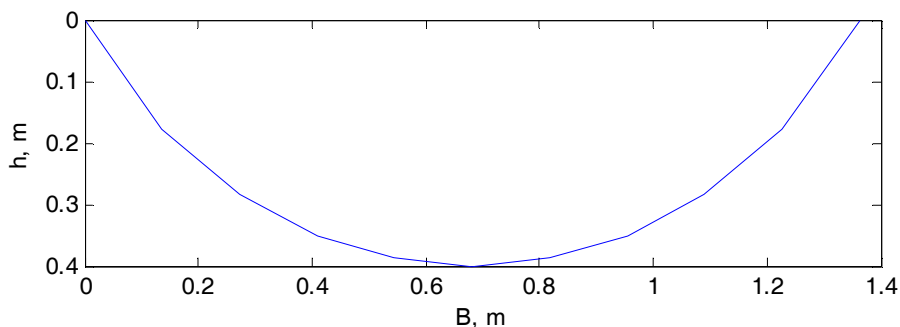
Dodatkowo napisano funkcję `void(u,tg,Y)`, która oblicza lokalną porowatość na podstawie wzoru 6.28 oraz funkcje do obliczania gęstości (`rog`) i lepkości gazu (`mig`). Program pokazuje wyniki obliczeń lokalnej prędkości graficznie, a spadku ciśnienia liczbowo. Dodatkowo podaje różnicę między danym i obliczonym, jako suma udziałów, natężeniem przepływu powietrza.

Warto tu skomentować sposób przekazywania dodatkowych parametrów do funkcji `uk_row`. Ponieważ funkcja ta wymaga więcej parametrów niż niewiadome rozwiązywanego układu równań dokonano tego przez funkcję anonimową `fs`.

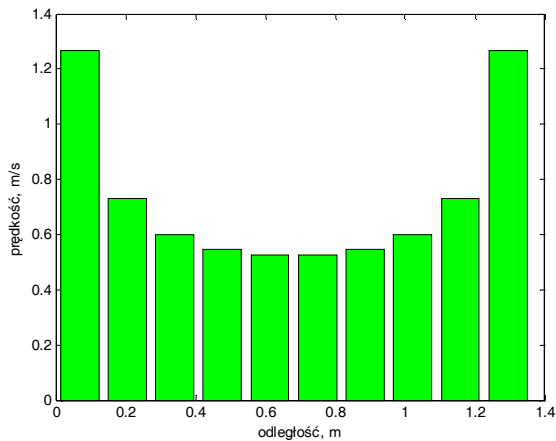
Wyniki 6.3

```
DP =
    1.1562e+003
```

```
DV =
    0
```



Rys. 6.4. Widok przekroju warstwy
(oś wysokości odwrócono ręcznie w menu Figure)



Rys. 6.5. Obliczony rozkład prędkości lokalnej

6.2.2. Rozwiązywanie nadokreślonych układów równań

Do tej pory rozważaliśmy układ równań o kwadratowej macierzy współczynników, tzn. liczba równań równała się liczbie niewiadomych. A co w przypadku, gdy mamy do czynienia z układem, w którym jest więcej równań niż niewiadomych? Układ taki nazywamy układem nadokreślonym. Rozwiązywanie nadokreślonego układu równań jest zagadnieniem optymalizacyjnym (*Szustalewicz, 2002/2003*). Weźmy układ równań

$$\begin{aligned} 2x_1 + x_2 &= -1 \\ -3x_1 + x_2 &= -2 \\ -x_1 + x_2 &= 1 \end{aligned} \tag{6.30}$$

Układ ten przedstawia trzy proste na płaszczyźnie x_1, x_2 . Jego rozwiązaniem w sensie geometrycznym jest punkt, który leży najbliżej wszystkich trzech prostych. Ilustruje to rysunek 6.6. Ścisłe rozwiązanie tego układu wg zasad rachunku macierzowego ma postać

$$x = (A' \cdot A)^{-1} \cdot A' \cdot B. \tag{6.31}$$

MATLAB oblicza to rozwiązanie przez wyrażenie $A \setminus B$ (wyrażenie $A^{-1}B$ nie działa z uwagi na to, że macierz A nie jest kwadratowa), posługując się metodą najmniejszych kwadratów.

Przykład 6.6

p6_6.m

%Przykład 6.6 Nadokreślony układ równań

function nadokr

clear all; clc

global A B

%trzy równania dwie niewiadome

A=[2 1; -3 1; -1 1];

B=[-1 -2 1]';

x1=A\B %rozwiązanie MATLABa

x2=(A'*A)^-1*A'*B %rozwiązanie ścisłe

x=linspace(-2,2,20);

hold on

plot(x,f1(x)), plot(x,f2(x))

plot(x,f3(x)), plot(x1(1),x1(2),'o')

xlabel('x'), ylabel('y')

function y=f1(x)

global A B

y=B(1)-A(1,1)*x;

function y=f2(x)

global A B

y=B(2)-A(2,1)*x;

function y=f3(x)

global A B

y=B(3)-A(3,1)*x;

Rezultat 6.6

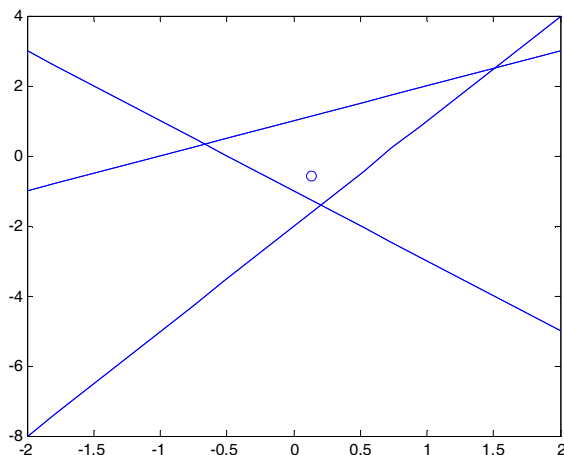
Poniżej x_1 to rozwiązanie MATLABa, x_2 to rozwiązanie ścisłe.

$x_1 =$

0.1316

-0.5789

x2 =
0.1316
-0.5789



Rys. 6.6. Ilustracja rozwiązania nadokreślonego układu równań 6.33

6.2.3. Rozwiązywanie nieoznaczonych układów równań

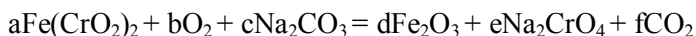
W przypadku nieoznaczonego układu równań mamy do czynienia z mniejszą liczbą równań niż liczba niewiadomych. Rozwiązanie jest wtedy możliwe jedynie jako funkcja wybranych niewiadomych.

Poniżej przedstawiono projekt, w którym spotykamy się z takim układem równań.

PROJEKT 6.4

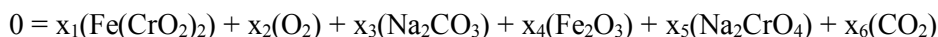
Temat: Stechiometria reakcji chemicznej
Toolbox: -

Zbilansować równanie reakcji utleniania chromitu FeCr_2O_4 przy użyciu węglanu sodowego. Należy ustalić wartości współczynników a, b, c, d, e oraz f.



Zwykle w przypadku prostych równań dokonujemy doboru współczynników stechiometrycznych ręcznie, jednakże w przypadku równań bardziej złożonych przydatna jest automatyzacja tego procesu.

Przedstawmy równanie z nieznanymi współczynnikami stechiometrycznymi, zapisanymi jako wektor x .



Uprościmy zapis, wprowadzając odpowiednie indeksy i stwórzmy układ równań dla każdego pierwiastka.

Pierwiastek	Nr	Równanie
Fe	1	$x_1 + 2x_4 = 0$
Cr	2	$2x_1 + x_5 = 0$
O	3	$4x_1 + 2x_2 + 3x_3 + 3x_4 + 4x_5 + 2x_6 = 0$
Na	4	$2x_3 + 2x_5 = 0$
C	5	$x_3 + x_6 = 0$

Zatem mamy pięć równań i sześć niewiadomych, czyli układ jest nieoznaczony. Macierz współczynników układu jest macierzą prostokątną [5x6]. Należy ją zredukować do macierzy [5x5]. Wtedy ostatni wiersz zawiera wektor, który pomnożony przez ostatni współczynnik, x_6 określi pozostałe współczynniki. Ponieważ wektor ten może zawierać współczynniki ułamkowe należy tak dobrać x_6 , aby stały się one całkowite.

P6_4.m

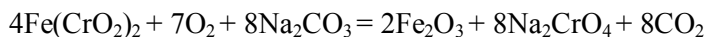
```
% PROJEKT 6.6
% Dobór współczynników stechiometrycznych reakcji chemicznej.
clear all; clc
% Definiujemy macierz współczynników
A=[ 1 0 0 2 0 0
    2 0 0 0 1 0
    4 2 3 3 4 2
    0 0 2 0 2 0
    0 0 1 0 0 1 ];
% dokonujemy dekompozycji macierzy metodą Gaussa-Jordana
[R] = rref(A);
% ostatnia kolumna macierzy R to wyniki R
```

% należy je pomnożyć przez najmniejszą liczbę całkowitą tak aby były
 % całkowite. Dobrana liczba wynosi 8. Jest to współczynnik x6.
 $x=R(:,6)*8$

```
R =
    1.0000         0         0         0         0    0.5000
         0    1.0000         0         0         0    0.8750
         0         0    1.0000         0         0    1.0000
         0         0         0    1.0000         0   -0.2500
         0         0         0         0    1.0000   -1.0000

x =
     4
     7
     8
    -2
    -8
```

Tak więc równanie ma ostatecznie postać



LITERATURA

Myers A.L., Seider W.D., „Obliczenia komputerowe w inżynierii chemicznej”,
 Seria: Inżynieria chemiczna, WNT 1979, s. 225.

Pakowski Z., Głębowski M., „Symulacja procesów inżynierii chemicznej”, Wyd.
 PŁ, 2001.

Pakowski Z., „Projektowanie suszarek do suszenia parą przegrzaną”, Wyd. PŁ,
 Łódź 2011.

Szustalewicz A., „Matematyka obliczeniowa”, Mat. wykład., Uniwersytet
 Wrocławski 2002/2003, w. 11.

7. ROZWIĄZYWANIE RÓWNAŃ RÓŻNICZKOWYCH ZWYCZAJNYCH

7.1. Typy równań i warunków brzegowych

Równanie różniczkowe postaci

$$\frac{dy}{dx} = f(x, y) \quad (7.1)$$

jest równaniem różniczkowym zwyczajnym (ang. *ODE – ordinary differential equation*) pierwszego rzędu, a równanie różniczkowe postaci

$$\frac{d^2y}{dx^2} = f\left(x, y, \frac{dy}{dx}\right) \quad (7.2)$$

jest równaniem różniczkowym zwyczajnym drugiego rzędu. Równania wyższych rzędów nie będą rozważane jako mało przydatne w praktyce inżynierskiej. Gdy $f = 0$ równanie nazywa się *jednorodnym*.

Przez proste podstawienie $dy/dx = z$ równanie II rzędu 7.2 można przekształcić w układ równań I rzędu

$$\begin{aligned} \frac{dy}{dx} &= z \\ \frac{dz}{dx} &= f(x, y, z) \end{aligned} \quad (7.3)$$

Do rozwiązania pojedynczego równania I rzędu niezbędny jest jeden warunek określający wartość funkcji y w danym punkcie x należącym do przedziału $\langle x_p, x_k \rangle$. Jeśli jest on dany na lewym krańcu przedziału zmiennej niezależnej x , problem rozwiązania równania określa się jako zagadnienie początkowe (zagadnienie Darcy), jeśli wewnątrz przedziału lub na jego drugim krańcu wtedy jest to zagadnienie brzegowe.

W przypadku układu dwu równań I rzędu, lub pojedynczego równania II rzędu, jeśli jeden warunek jest określony na lewym a drugi na prawym krańcu przedziału mamy do czynienia z tzw. dwupunktowym zagadnieniem brzegowym. Sytuacja ta występuje np. przy symulacji przeciwprądowego kontaktu dwu faz oraz podczas rozwiązywania równań różniczkowych II rzędu np. konwekcji z nałożoną dyfuzją bądź dyspersją wzdłużną.

7.1.1. Sztywność równań różniczkowych

7.1.1.1. Pojedyncze sztywne ODE

Funkcja, która w otoczeniu pewnego punktu gwałtownie rośnie lub maleje stwarza ogromne problemy w rozwiązaniu numerycznym (zob. Rozdz. 7.2). Równanie takie nazywamy sztywnym (ang. *stiff*). Przykład prostego równania ODE, którego rozwiązaniem jest taka funkcja, podaje Moler, 2004. Ma ono postać

$$\frac{dr}{d\tau} = r^2 - r^3 \quad (7.4)$$

Należy je rozwiązać dla $r(0) = \delta$ w przedziale $0 < \tau < 2/\delta$. Rozwiązanie dąży do funkcji skokowej, gdy $\delta \rightarrow 0$ i w otoczeniu punktu $1/\delta$ gwałtownie rośnie. Nb. jest to model wzrostu kuli ognia, której promień zależy od dostępnej ilości O_2 . Szybkość dopływu O_2 zależy od powierzchni (proporcjonalnie do r^2) a szybkość jego zużywania od objętości (proporcjonalnie do r^3). Równanie to rozwiązano w Przykładzie 7.3.

7.1.1.2. Sztywność układów równań

W przypadku układów równań I rzędu czasem występuje sytuacja, gdy jedna funkcja zmienia się bardzo wolno lub bardzo szybko w stosunku do pozostałych. Układ taki nazywamy sztywnym układem równań różniczkowych (Finlayson, 1980). Współczynnik sztywności jest ściśle określony jedynie dla układu równań liniowych. Jest to wtedy stosunek największej do najmniejszej wartości modułu części rzeczywistej wartości własnych (ang. *eigenvalues*) macierzy współczynników tego układu

$$s = \frac{\max |\operatorname{Re}(\lambda)|}{\min |\operatorname{Re}(\lambda)|} \quad (7.5)$$

Za sztywny uważa się układ, którego współczynnik sztywności jest większy od 1000. Występowanie sztywności znacznie utrudnia numeryczne rozwiązywanie równań – przy użyciu integratora o stałym kroku musiałby on być bardzo mały, co wydłużyłoby znacznie czas całkowania. Dla układów sztywnych stosuje się specjalne algorytmy zastosowane w integratorach opisanych w następnej sekcji. W inżynierii procesowej sztywność układu równań występuje często w modelach kinetyki reakcji chemicznych a także jednoczesnej wymiany ciepła i masy, mającej miejsce w wielu procesach.

7.1.2. Numeryczne rozwiązywanie ODE

Korzysta się tu z przybliżenia pochodnej ilorazem różnicowym, który zbliża się do wartości pochodnej tym bardziej im mniejszy jest przyrost zmiennej niezależnej. Najprostszą metodą numeryczną rozwiązywania ODE z warunkiem początkowym jest tzw. jawna metoda Eulera, polegająca na zastąpieniu równania 7.1 równaniem

$$\frac{\Delta y}{\Delta x} \cong f(x, y). \quad (7.6)$$

Mając warunek początkowy y_0 , kolejne wartości funkcji obliczymy z zależności

$$y_{n+1} \cong y_n + f(x_n, y_n) \Delta x. \quad (7.7)$$

Metoda ta jest niezwykle prosta, lecz wymaga małego kroku całkowania Δx , aby uzyskać zbieżność rozwiązania numerycznego i ścisłego (analitycznego). Szybciej zbieżna jest metoda niejawna Eulera, posługująca się wartością pochodnej obliczoną w punkcie $n+1$.

$$y_{n+1} \cong y_n + f(x_{n+1}, y_{n+1}) \Delta x. \quad (7.8)$$

Wymaga ona jednak podczas każdego kroku całkowania rozwiązywania równania 7.8 celem znalezienia wartości y_{n+1} .

Od czasu pierwszych algorytmów tego rodzaju stworzono cały szereg wysoko-wydajnych integratorów równań różniczkowych zwyczajnych. Należą do nich również integratory ODE MATLABa.

7.2. Integratory numeryczne równań różniczkowych MATLABa

Narzędzia numeryczne służące do rozwiązywania zagadnień początkowych dla ODE oraz ich układów nazywamy solverami lub integratorami. Mają one następujące najprostsze wywołanie (pełne wywołanie opisano w sekcji 7.3.3):

```
[t,x]=<nazwa>(@fun,[tp tk],x0)
```

gdzie

<nazwa> oznaczenie integratora wybrane z zestawienia w Tablicy 7.1 (poza ode15i),

@fun uchwyt funkcji, w której zdefiniowane jest równanie lub układ równań (jeśli rozwiązujemy układ równań zapisujemy go w formie wektora prawych stron równań 7.1),

t_p, t_k przedział zmiennej niezależnej (dziedzina funkcji) – zamiast niego można wstawić wektor kolejnych wartości t , dla których chcemy uzyskać rozwiązanie
 x_0 wektor warunków początkowych
 W rozwiązaniu:
 x wektor wartości zmiennej zależnej,
 t wektor kolejnych wartości zmiennej niezależnej odpowiadających kolejnym wartościom x .

Dostępne w MATLABie solvery ODE zawarte są w Tablicy 7.1.

Tablica 7.1. Solvery numeryczne ODE w MATLABie

Integrator	Typ układu	Dokładność
ode23	niesztywny	mała
ode45	niesztywny	średnia
ode113	niesztywny	mała do wysokiej
ode15s	sztywny	mała do średniej
ode23s	sztywny	mała
ode23t	mało sztywny	mała
ode23tb	sztywny	mała
ode15i	uwikłany	-

Sposób użycia integratorów z tabl. 7.1 ilustrują poniższe przykłady.

Przykład 7.1. Pojedyncze ODE

Rozwiążmy równanie postaci:

$$\frac{dx}{d\tau} = x^{5 \cdot \sin(x)} \quad (7.9)$$

z warunkiem początkowym

$$\tau = 0 \quad x = 1 \quad (7.10)$$

w przedziale $\langle 0, 2 \rangle$.

 p7_1.m

%Przykład 7.1 Pojedyncze ODE

```
function ODEjeden
```

```
clear all; clc
```

```
% warunek początkowy
```

```
x0=[1];
```

```
[t,x]=ode45(@row,[0 2],x0);
```

```
% wizualizacja wyników
```

```
plot(t,x)
```

```
xlabel('\tau'); ylabel('x')
```

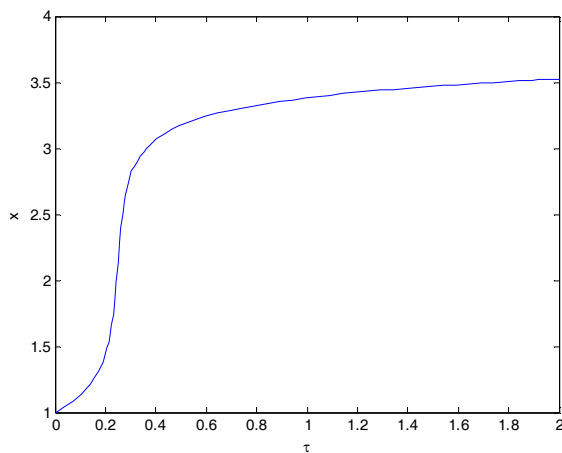
```
% funkcja w której zdefiniowana jest prawa strona równania róż-
```

```
niczkowego
```

```
function y = row(t,x)
```

```
y = [ x(1).^(5.*sin(x(1))) ];
```

Rezultat 7.1



Rys. 7.1. Rozwiązanie równania różniczkowego 7.9

Uwaga: wywołanie solvera bez argumentów wyjściowych

```
ode45(@row,[0 2],x0);
```

spowoduje automatyczne generowanie wykresu podczas rozwiązywania w standardowym formacie. Warto to sprawdzić w powyższym przykładzie. Wtedy należy skomentować te linie

```
plot(t,x)
```

```
xlabel('t'); ylabel('x')
```


Przykład 7.2. Układ ODE

Rozwiążmy układ równań różniczkowych postaci:

$$\frac{dx}{d\tau} = \begin{bmatrix} 3x_1 + x_2^3 - \cos(x_1) \\ \sin(x_2)x_1 - 4x_1 \end{bmatrix} \quad (7.11)$$

W przedziale $\langle 1, 2 \rangle$ z warunkiem początkowym

$$\tau = 1 \quad x_1 = 0 \quad x_2 = 2 \quad (7.12)$$

p7_2.m

%Przykład 7.2 Dwa równania ODE

```
function ODEdwa
```

```
clear all; clc
```

```
% warunek początkowy
```

```
x0=[0; 2];
```

```
[t,x]=ode113(@row,[1 2],x0);
```

```
% wizualizacja wyników
```

```
plot(t,x)
```

```
xlabel('\tau'); ylabel('x')
```

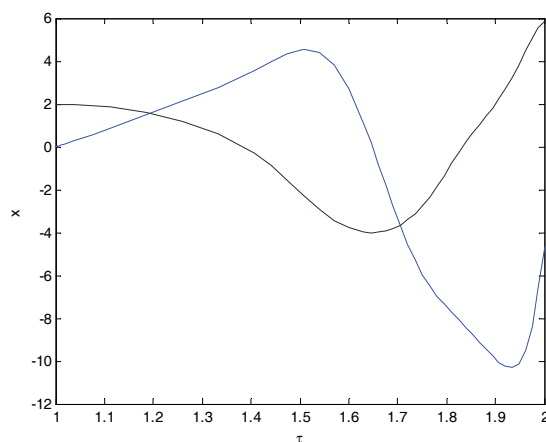
```
% funkcja w której zdefiniowane jest równanie różniczkowe
```

```
function y = row(t,x)
```

```
y = [3.*x(1)+x(2).^3-cos(x(1))
```

```
sin(x(2)).*x(1)-4.*x(1)];
```

Rezultat 7.2



Rys. 7.2. Rozwiązanie układu równań różniczkowych 7.11

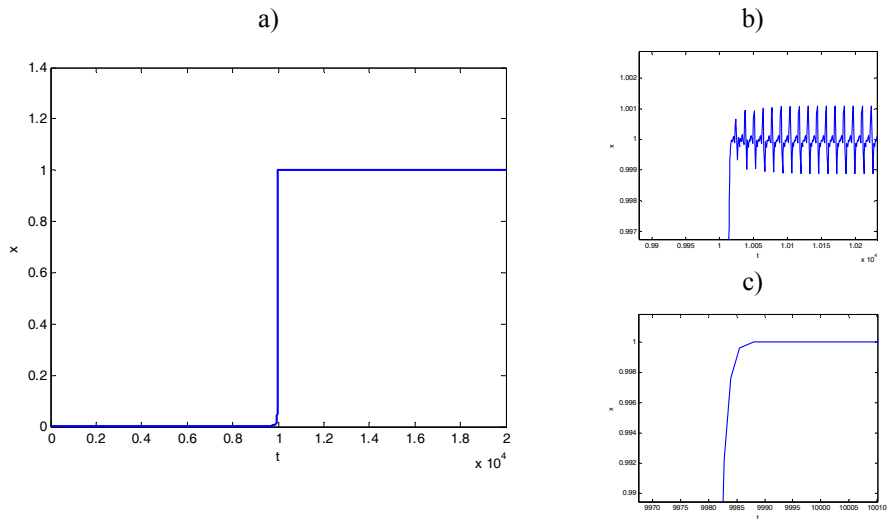
Przykład 7.3. Pojedyncze sztywne ODE

W przykładzie tym porównano dwa typy integratorów, użyte do rozwiązywania równania sztywnego 7.4.

p7_3.m

```
%Przykład 7.3 Sztywne ODE
function ODEstiff
clear all; clc
delta=0.0001;
% warunek początkowy
x0=[delta];
t1=cputime;
[t,x]=ode23s(@row,[0 2/delta],x0);
t2=cputime;
% wizualizacja wyników
plot(t,x,'b','LineWidth',2)
xlabel('t'); ylabel('x')
disp('czas obliczeń [s]'); disp(t2-t1);
% prawa strona równania różniczkowego
function y = row(t,x)
y = [ x(1).^2-x(1).^3 ];
```

Rezultat 7.3



Rys. 7.3. Rozwiązanie równania różniczkowego 7.4, a) rozwiązanie. Powiększenie okolic punktu (1,1) – b) ode45, c) ode23s

Przy użyciu integratora `ode45` odnotowano czas wykonania 1.669 s. Po wymianie na integrator do równań sztywnych `ode23s` czas wykonania zmniejszył się do 0.328 s. Dodatkowo, po powiększeniu na rysunku opcją **Zoom in** fragmentu odcinka krzywej tuż za punktem skoku widać, że rozwiązanie uzyskane integratorem `ode45` oscyluje, czego nie zauważa się na rozwiązaniu integratorem `ode23s`.

Przykład 7.4. Sztywny układ ODE

Rozwiążmy sztywny układ równań van der Pola (Walas, 1991, s. 103) postaci

$$\frac{dy}{d\tau} = \begin{bmatrix} y_2 \\ 1000(1 - y_1^2)y_2 - y_1 \end{bmatrix} \quad (7.13)$$

z warunkiem początkowym

$$\begin{aligned} y_1(0) &= 2 \\ y_2(0) &= 0 \end{aligned} \quad (7.14)$$

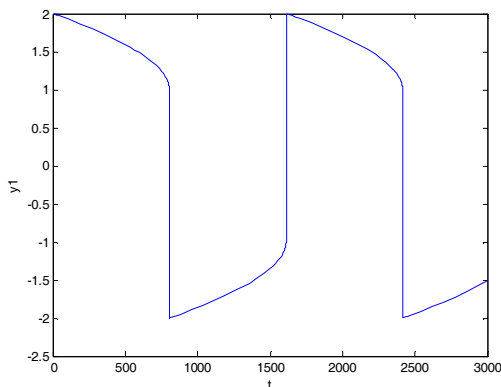
P7_4.m

```
%Przykład 7.4 Dwa sztywne ODE
function ODEstiff_two
%przykład 7.4
clear all; clc
% warunek początkowy
x0=[2 0];
t1=cputime;
[t,x]=ode15s(@row,[0 3000],x0);
t2=cputime;

% wizualizacja wyników
plot(t,x(:,1))
xlabel('t'); ylabel('y1')
disp('czas obliczeń [s]'); disp(t2-t1);
function dy = row(t,y)
% równania van der Pola
dy=zeros(2,1);
dy(1) = y(2);
dy(2)=1000*(1-y(1)^2)*y(2)-y(1);
```

Rezultat 7.4

Ponieważ układ równań van der Pola pochodzi z przekształcenia równania drugiego rzędu na układ równań pierwszego rzędu pokażemy tylko wykres funkcji y_1 . Funkcja y_2 to pierwsza pochodna y_1 .



Rys. 7.4. Rozwiązanie układu równań van der Pola wykazujące jego silną sztywność

7.3. Rozwiązywanie zagadnień początkowych

7.3.1. Procesy nieustalone

Procesy nieustalone to te, w których równaniu bilansowym występuje akumulacja. Ma ona postać pochodnej po czasie. Procesy nieustalone opisuje dynamika procesów. Model dynamiki obiektu o parametrach skupionych jest równaniem bądź układem równań różniczkowych z pochodną (I rzędu) poszukiwanej funkcji po czasie.

Przykładem takiego równania jest np. równanie bilansu ciepła cieczy w mieszalniku z idealnym wymieszaniem, ogrzewanej kondensującą parą przez ściankę zbiornika.

$$mc_{pl} \frac{dT_l}{d\tau} = KA(T_p - T_l), \quad (7.15)$$

gdzie: m – masa cieczy w mieszalniku [kg], c_{pl} – ciepło właściwe cieczy [kJ/(kg K)], A – powierzchnia ogrzewana [m²], K – współczynnik przenikania ciepła [kW/(m²K)], T_p – temperatura pary [K], T_l – temperatura cieczy [K], τ – czas [s].

Równanie należy przekształcić do postaci:

$$\frac{dT_l}{d\tau} = \frac{KA}{mc_{pl}}(T_p - T_l) \quad (7.16)$$

Mając do dyspozycji warunek początkowy

$$\text{dla } \tau = 0 \quad T_l = T_{l0}$$

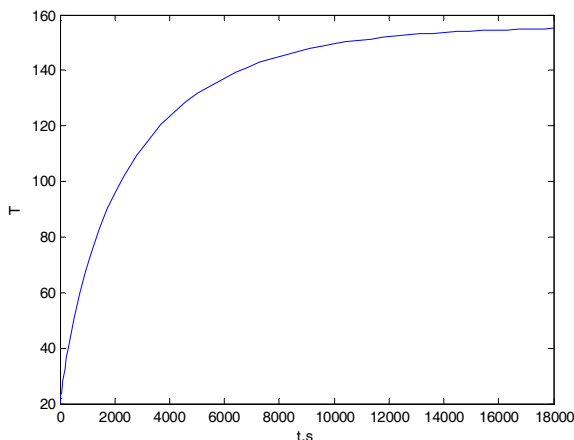
równanie można rozwiązać stosując jeden z integratorów z Tablicy 7.1. Ilustruje to przykład 7.5. Uwaga: przyjęto, że ciepło właściwe c_{pl} jest liniową funkcją temperatury.

Przykład 7.5. Nieustalone nagrzewanie cieczy

 p7_5.m

```
%Przykład 7.5 Nieustalone nagrzewanie
function nieust_nagrz
% nagrzewanie cieczy w zbiorniku pod ciśnieniem
clear all; clc
global cp0 K A m Tp cp0
% dane
cp0 = 3.89; % ciepło właściwe w 0 C, kJ/(kg K)
K = 2.7; % współczynnik przenikania ciepła, kW/(m2 K)
A = 1.55; % powierzchnia ogrzewana, m2
m = 1500; % masa cieczy, kg
Tp = 156; % temperatura pary, C
T0 = 20; % początkowa temperatura cieczy, C
% warunek początkowy
x0=[T0];
[t,x]=ode45(@rown,[0 18000],x0);
% wizualizacja wyników
plot(t,x)
xlabel('t'); ylabel('T')
% funkcja w której zdefiniowana jest prawa strona równania róż-
niczkowego
function y = rown(t,T)
global K A m Tp
y = (K*A/m/cp(T))*(Tp-T);
function y = cp(T)
global cp0
y = cp0*(1+0.012*T);
```

Rezultat 7.5



Rys. 7.5. Przebieg procesu nagrzewania cieczy w mieszalniku

W tym układzie dynamicznym obserwuje się monotoniczne dochodzenie układu do stanu stacjonarnego. Bardziej złożone zachowania dynamiczne obserwuje się dla obiektów opisanych układem równań ODE lub równaniem ODE rzędu drugiego, które, jak wiadomo, daje się sprowadzić do układu dwu równań ODE rzędu pierwszego. Opisano to poniżej.

7.3.1.1. Teoretyczne podstawy dynamiki liniowej

Weźmy układ, którego zachowanie dynamiczne opisują dwa równania liniowe.

$$\begin{aligned}\frac{dx}{d\tau} &= a_1x + b_1y \\ \frac{dy}{d\tau} &= a_2x + b_2y\end{aligned}\tag{7.17}$$

Rozwiązanie tego układu, otrzymane dla danego wektora warunków początkowych, można przedstawić na wykresach czasowych $x(\tau)$ i $y(\tau)$ bądź na płaszczyźnie fazowej jako wykres $y(x)$. Wykres zależności $y(x)$ rozpoczynający się w punkcie początkowym (x_0, y_0) i kończący w punkcie końcowym (x_k, y_k) nosi nazwę *trajektorii procesu*. Do dalszej analizy zachowania się rozwiązania układu 7.17 na płaszczyźnie fazowej posłużą wartości własne macierzy współczynników

prawej strony tego układu. Wartości te λ_1 i λ_2 wyznacza równanie charakterystyczne

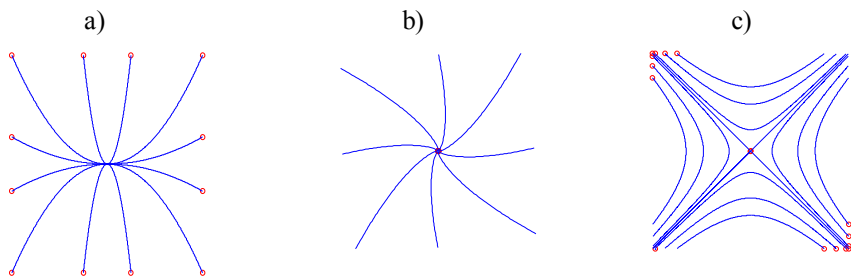
$$\lambda^2 - (a_1 + b_2)\lambda + (a_1b_2 - a_2b_1) = 0. \quad (7.18)$$

W MATLABie można je obliczyć funkcją **eig(A)**, gdzie $A = [a_1 \ b_1; a_2 \ b_2]$. W zależności od wartości własnych λ_1 i λ_2 układ równań może mieć różne trajektorie.

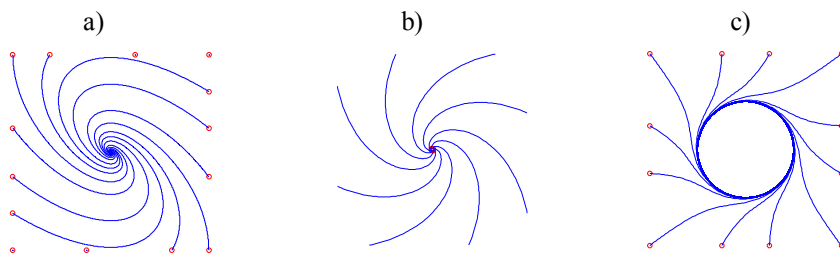
Tablica 7.2. Możliwe trajektorie układu równań 7.17

Wartości własne	Zachowanie	Stabilność
1. Pierwiastki są rzeczywiste i różne od zera		
a) $\lambda_1 < 0$; $\lambda_2 < 0$	węzeł	stabilny asymptotycznie
b) $\lambda_1 > 0$; $\lambda_2 > 0$	węzeł	niestabilny
c) $\lambda_1 > 0$; $\lambda_2 < 0$	siodło	niestabilny
2. Pierwiastki są zespolone postaci $a+bi$ lub $a-bi$		
a) $a < 0$; $b \neq 0$	ognisko	stabilne
b) $a > 0$; $b \neq 0$	ognisko	niestabilne
c) $a = 0$; $b \neq 0$	centrum	stabilne
3. Pierwiastki są równe		
a) $\lambda < 0$	węzeł	stabilny asymptotycznie
b) $\lambda > 0$	węzeł	niestabilny

Zachowania te pokazano na Rys. 7.6 i 7.7.



Rys. 7.6. Trajektorie wygenerowane dla przypadków a) węzeł stabilny b) węzeł niestabilny c) siodło niestabilne. Kółeczka oznaczają punkty początkowe



Rys. 7.7. Trajektorie wygenerowane dla przypadków, a) ognisko stabilne b) ognisko niestabilne, c) centrum stabilne. Kółeczka oznaczają punkty początkowe.

Przedstawione poniżej przykłady ilustrują niektóre z tych przypadków.

Przykład 7.6. Centrum stabilne

Centrum stabilne jest opisane układem następujących równań różniczkowych

$$\begin{aligned}\frac{dx_1}{d\tau} &= 5x_1 - 5x_2 \\ \frac{dx_2}{d\tau} &= 10x_1 - 5x_2\end{aligned}\tag{7.19}$$

Rozwiążemy ten układ w przedziale $\langle 0, 20 \rangle$.

p7_6.m

```
%Przykład 7.6 Centrum stabilne
clear all; close all
global A
% zakres zmiennej niezależnej
t0=0; tk=20;
% macierz prawych stron
A=[5 -5;10 -5];
eigenvalues=eig(A)
% warunek początkowy
x0=[1; 1];
% wywołanie solvera
[t,x]=ode23s('centr_stab_eq',[t0 tk],x0);
% prezentacja graficzna
subplot(2,1,1);
plot(t,x(:,1),'r');
```



```

title('Rozwiazanie');
xlabel('czas');
hold on;
plot(t,x(:,2),'b');
subplot(2,1,2);
plot(x(:,1),x(:,2));
title('wykres na płaszczyźnie fazowej');

```

centr_stab_eq.m jest plikiem w którym zapisany jest układ równań

```

function y=centr_stab_eq(t,x)
global A
y=A*x;

```

Rezultat 7.6

Po uruchomieniu programu w oknie **Command Window** pojawiają się obliczone wartości własne macierzy prawych stron tego układu równań.

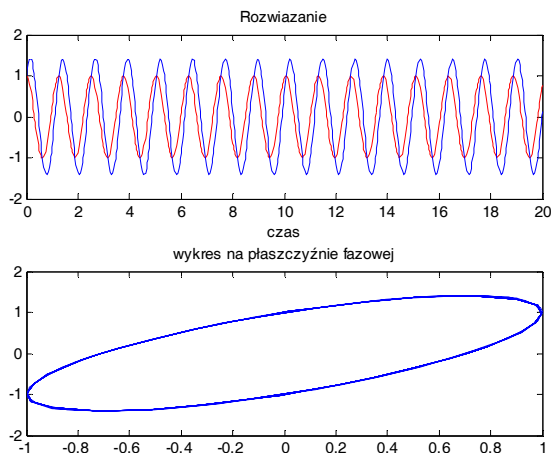
eigenvalues =

```

0.0000 + 5.0000i
0.0000 - 5.0000i

```

Jak widać obie ich części rzeczywiste są zerami, natomiast części urojone są różne od zera. Zachodzi więc przypadek 2c z Tabeli 7.2, a więc układ powinien wygenerować centrum stabilne. W istocie wykresy rozwiązania to potwierdzają.



Rys. 7.8. Rozwiązanie równania tworzącego centrum stabilne: a) przebieg oscylacji w czasie, b) przedstawienie wyników na płaszczyźnie fazowej

Przykład 7.7. Ognisko niestabilne

Ognisko niestabilne opisane jest następującym układem równań różniczkowych

$$\begin{aligned}\frac{dx_1}{d\tau} &= x_1 - 20x_2 \\ \frac{dx_2}{d\tau} &= x_1\end{aligned}\tag{7.20}$$

p7_7.m

```
%Przykład 7.7 Ognisko niestabilne
clear all; close all
% zakres zmiennej niezależnej
t0=0; tk=5;
% warunek początkowy
x0=[1; 1];
% wywołanie solvera
[t,x]=ode23tb('ogn_niestab_eq',[t0 tk],x0);
% prezentacja graficzna
figure(1)
plot(t,x(:,1),'r');
title('Rozwiązanie');
xlabel('czas');
hold on;
plot(t,x(:,2),'b');
figure(2)
plot(x(:,1),x(:,2));
title('wykres na płaszczyźnie fazowej');
```

ogn_niestab_eq.m jest plikiem w którym zapisany jest układ równań

```
function y=ogn_niestab_eq(t,x)
y=[x(1)-20*x(2)
    x(1)];
```

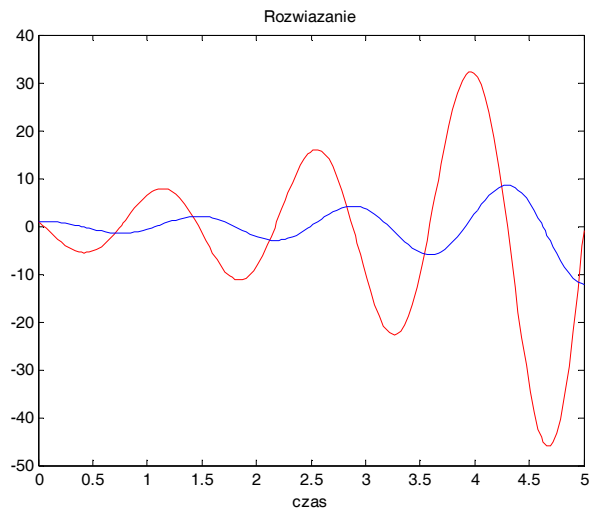
Rezultat 7.7

Otrzymano następujące wartości własne

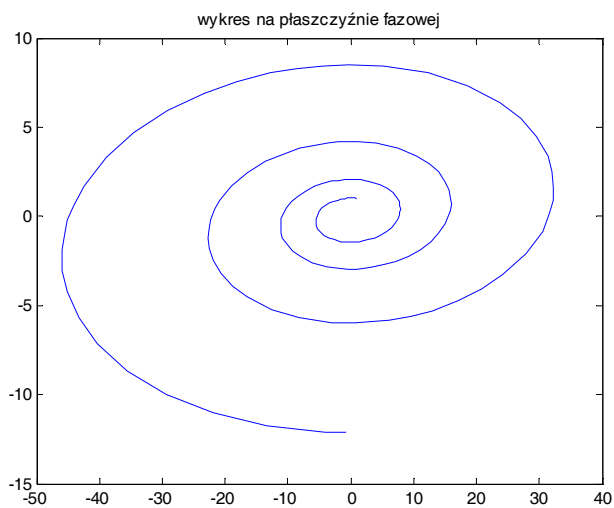
eigenvals =

```
0.5000 + 4.4441i
0.5000 - 4.4441i
```

Wynika z nich typ 2b zachowania wg Tablicy 7.2, czyli ognisko niestabilne.



Rys. 7.9. Rozwiązanie równania tworzącego ognisko niestabilne
– oscylacje rosną w czasie



Rys. 7.10. Rozwiązanie równania tworzącego ognisko niestabilne
– wykres na płaszczyźnie fazowej

7.3.1.2. Dynamika nieliniowa

Układy równań, których prawą stronę stanowią zależności nieliniowe występują w praktyce równie często, a nawet częściej, jak te z zależnościami liniowymi. Nieliniowe równania różniczkowe I rzędu z jedną niewiadomą mają tylko punkty stacjonarne jako rozwiązania asymptotyczne. Na płaszczyźnie fazowej mają one postać węzła stabilnego. Dla układu dwu równań I rzędu oprócz rozwiązań stacjonarnych mogą pojawić się rozwiązania periodyczne oscylujące w czasie. Na płaszczyźnie fazowej mają one postać cyklu granicznego (krzywej zamkniętej). Takie przyciągające trajektorie obiekty nazywamy atraktorami. Atraktory mają tzw. wymiar przestrzenny, węzły wymiar 0 a cykle graniczne 1. W przypadku układu trzech równań nieliniowych I rzędu mogą wystąpić jeszcze atraktory o wymiarze fraktalnym (ułamkowym). Nazywa się je dziwnymi atraktorami (Ott, 1997). Pojawiają się wtedy rozwiązania oscylujące, ale nie periodycznie, mające dodatkowo tę własność że silnie zależą od warunków początkowych. Oznacza to, że minimalna nawet zmiana warunku początkowego powoduje, że po upływie odpowiednio długiego czasu oddalają się znacznie od siebie („efekt motyla”). Rozwiązania takie są bardzo skomplikowane i mają charakter chaotyczny. Jednak ten rodzaj chaosu w odróżnieniu od chaosu stochastycznego nazywany jest deterministycznym.

Warto prześledzić techniki rozwiązywania układów równań ODE opisujących takie zjawiska na kilku wybranych przykładach.

Przykładem niekonwencjonalnego zachowania się układu nieliniowego dwu równań jest zjawisko bifurkacji rozwiązania równania Hopfa postaci

$$\begin{aligned}\frac{dx}{d\tau} &= -y + x[a - (x^2 - y^2)] \\ \frac{dy}{d\tau} &= x + y[a - (x^2 - y^2)]\end{aligned}\tag{7.21}$$

Dla ujemnych wartości parametru a uzyskuje się po pewnym czasie jedno rozwiązanie stacjonarne niezależnie od punktu startu na płaszczyźnie fazowej, dla dodatnich wartości a uzyskuje się powtarzające się oscylacje. Na płaszczyźnie fazowej powstaje cykl graniczny. Zjawisko zmiany stanu stacjonarnego obiektu z węzła stabilnego na cykl graniczny nazywa się bifurkacją Hopfa.

Przykład 7.8. Bifurkacja Hopfa

 **p7_8.m**

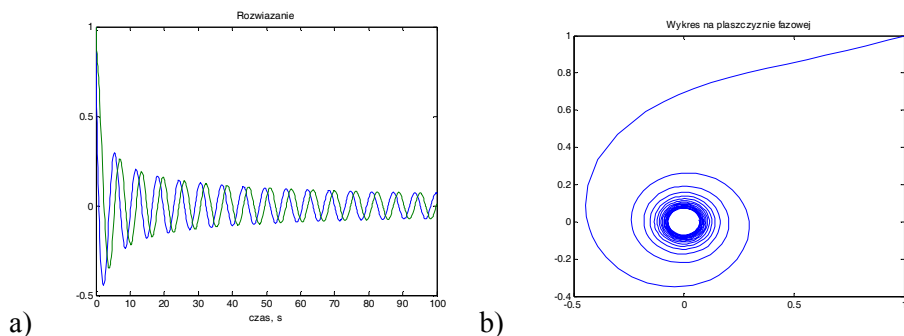
```
%Przykład 7.8 Bifurkacja Hopfa
clear all; close all; clc
x0=[1 1];
[t,x]=ode23tb(@Hopf_eq,[0 500],x0);
figure(1)
plot(t,x)
title('Rozwiazanie');
xlabel('czas, s');
figure(2)
plot(x(:,1),x(:,2))
title('Wykres na płaszczyźnie fazowej');
```

Hopf_eq.m jest plikiem w którym zdefiniowano układ równań

 **Hopf_eq.m**

```
function y=Hopf_eq(t,x)
%współczynnik zmienny
a=0;
%układ równań Hopfa
y=[ -x(2)+x(1)*(a-(x(1).^2+x(2).^2))
    x(1)+x(2)*(a-(x(1).^2+x(2).^2))];
```

Rezultat 7.8



Rys. 7.11. Rozwiązanie równania Hopfa: a) oscylacje w czasie, b) trajektoria na płaszczyźnie fazowej

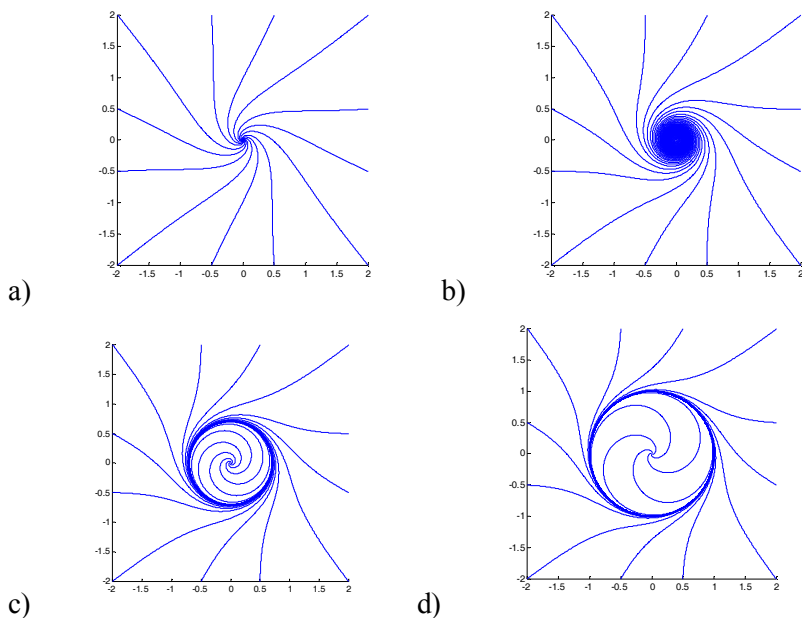
Można też wykorzystać funkcję **streamline** (sekcja 4.2.4.2), wygodne narzędzie obrazowania pól wektorowych 2D i 3D. Rysuje ona trajektorie rozwiązań na płaszczyźnie bądź w przestrzeni fazowej. Sposób wykorzystania

tej funkcji pokazano poniżej. Uruchamiamy plik **Hopfstrm.m** zawarty w tym samym katalogu.

Hopfstrm.m

```
clear all; close all;
v=linspace(-2,2,50);
[X,Y]=meshgrid(v);
%definicja pola wektorowego
a=1;
U= -Y+X.*(a-(X.^2+Y.^2));
V=  X+Y.*(a-(X.^2+Y.^2));
%punkty startowe trajektorii
X0=[-2 -2 -2 -2 -.5 -.5 .5 .5 2 2 2 2 -.01 -.01 .01 .01];
Y0=[-2 -.5 .5 2 -2 2 -2 2 -2 -.5 .5 2 -.01 .01 -.01 .01];
%wykres trajektorii
streamline(X,Y,U,V,X0,Y0)
axis square
```

Wyniki dla wybranych wartości parametru a pokazane są na Rys. 7.12.



Rys. 7.12. Trajektorie rozwiązań dla a) $a = -1$, b) $a = 0$, c) $a = 0.5$, d) $a = 1$

Uwaga: trajektoria na Rys. 7.13b dąży do ogniska stabilnego. Nie widać tego na Rys. 7.11b, gdyż tam przedział czasowy jest krótszy.

Dziwny atraktor o kształcie końskiego kopyta (ang. *horse hoof attractor*) – zob. *Tabiś, 2000* to rozwiązanie układu równań różniczkowych postaci

$$\begin{aligned}\frac{dx_1}{d\tau} &= 1 - x_1 - 18000x_1x_3^2 \\ \frac{dx_2}{d\tau} &= 1 - x_2 - 400x_2x_3^2 \\ \frac{dx_3}{d\tau} &= 1 + 1.5 \cdot 18000x_1x_3^2 + 4.2 \cdot 400x_2x_3^2 - 81x_3\end{aligned}\tag{7.22}$$

Przykład 7.9. Dziwny atraktor

 **p7_9.m**

%Przykład 7.9 Dziwny atraktor

clear all; close all; clc

% zakres czasu

t0=0;

tk=20;

% warunek początkowy

x=[0.1; 0.1; 0.1];

% wywołanie solvera

[t,x]=ode15s('dziwny_eq',[t0 tk],x);

% prezentacja graficzna

figure(1)

plot(t,x)

title('Wykres oscylacji')

xlabel('\tau')

ylabel('x1, x2, x3')

figure(2)

plot(x(:,1),x(:,2))

title('Wykres trajektorii na płaszczyźnie fazowej')

xlabel('x1'); ylabel('x2')

figure(3)

plot(x(:,1),x(:,3))

title('Wykres trajektorii na płaszczyźnie fazowej')

xlabel('x1'); ylabel('x3')

figure(4)

plot(x(:,2),x(:,3))

```

title('Wykres trajektorii na płaszczyźnie fazowej')
xlabel('x2'); ylabel('x3')
figure(5)
plot3(x(:,1),x(:,2),x(:,3),'r')
title('Wykres w przestrzeni fazowej')
xlabel('x1'); ylabel('x2'); zlabel('x3')
grid on

```

W pliku **dziwny_eq.m** zdefiniowano układ równań

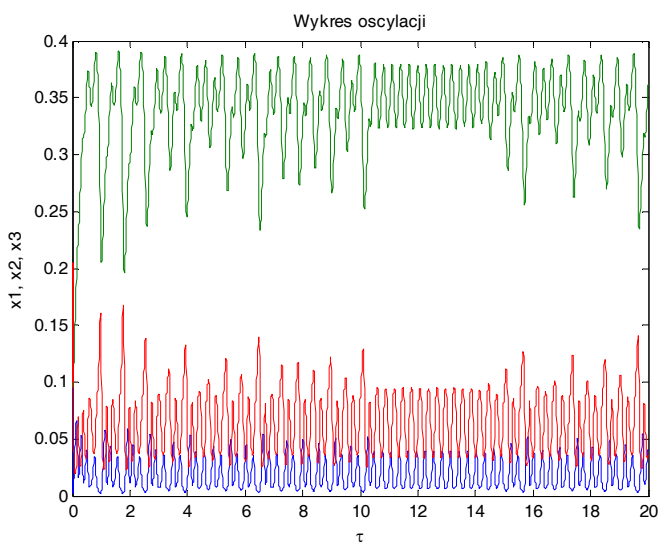
 **dziwny_eq.m**

```

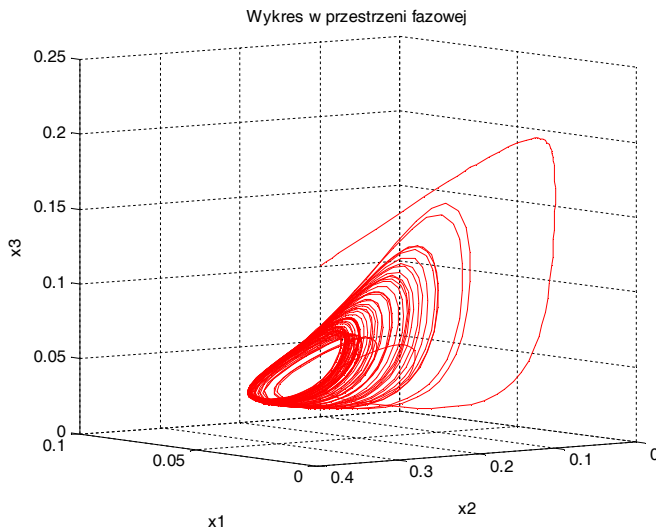
function y=dziwny(t,x)
ax=1.5;
ay=4.2;
k1=18000;
k2=400;
k3=80+1;
y=[ 1-x(1)-k1*x(1)*(x(3)^2)
    1-x(2)-k2*x(2)*(x(3)^2)
    1+ax*k1*x(1)*(x(3)^2)+ay*k2*x(2)*(x(3)^2)-k3*x(3) ];

```

Rezultat 7.9



Rys. 7.13. Przebiegi czasowe rozwiązania



Rys. 7.14. Wykres rozwiązania równań dziwnego atraktora w przestrzeni fazowej

Atraktor Lorenza (Orlik, 1996) występuje przy opisie zjawiska ruchu konwekcyjnego w warstwie cieczy ogrzewanej od dołu, które jest opisane układem równań (Lorenz, 1963)

$$\begin{aligned}\frac{dx_1}{d\tau} &= Pr \cdot (-x_1 + x_2) \\ \frac{dx_2}{d\tau} &= 1 - x_2 - 400x_2x_3^2 \\ \frac{dx_3}{d\tau} &= -bx_3 + x_1x_2\end{aligned}\tag{7.23}$$

Dla wartości liczby Prandtla równej 10 i parametru $b = 8/3$ zachowanie układu zależy wyłącznie od liczby Rayleigha Ra . Układ zachowuje się stabilnie do $Ra > \sim 20$ kiedy to pojawiają się chaotyczne oscylacje i układ nie osiąga już stanu stacjonarnego.

Przykład 7.10. Atraktor Lorenza

W przykładzie zastosowano animację wykresu atraktora (zob. Rozdz. 4).

p7_5.m

%Przykład 7.10 Atraktor Lorenza

```
clear all; close all; clc
```

```
% przedział czasu
```

```
t0=0; tk=40;
```

```
% warunek początkowy
```

```
x0=[5; 5; 5];
```

```
% wywołanie solvera
```

```
[t,x]=ode15s('Lorenz_eq',[t0 tk],x0);
```

```
% wizualizacja wyników
```

```
figure(1)
```

```
plot(t,x(:,1),'r',t,x(:,2),'b',t,x(:,3),'g');
```

```
xlabel('\tau'); ylabel('x1, x2, x3');
```

```
figure(2)
```

```
plot(x(:,1),x(:,2))
```

```
xlabel('x1'); ylabel('x2');
```

```
figure(3)
```

```
plot(x(:,1),x(:,3))
```

```
xlabel('x1'); ylabel('x3');
```

```
figure(4)
```

```
plot(x(:,2),x(:,3))
```

```
xlabel('x2'); ylabel('x3');
```

```
figure(5)
```

```
plot3(x(:,1),x(:,2),x(:,3));
```

```
xlabel('x1')
```

```
ylabel('x2')
```

```
zlabel('x3')
```

```
grid on
```

```
axis([min(x(:,1)) max(x(:,1)) min(x(:,2)) max(x(:,2))
```

```
min(x(:,3)) max(x(:,3))]);
```

```
%przygotowanie animacji
```

```
figure(6)
```

```
xf=x(:,1); y=x(:,2); z=x(:,3);
```

```
krzywa1 = plot3(xf,y,z,'r-
```

```
','EraseMode','none','LineWidth',2,'MarkerSize',2,'MarkerFace','r');
```

```
axis([min(xf) max(xf) min(y) max(y) min(z) max(z)]);
```

```
view(64,33);
```

```
xlabel('x1')
```

```
ylabel('x2')
```

```
zlabel('x3')
```

```

grid on
% animacja atraktora
for i=2:length(t)
    set(krzywa1,'XData',x(i-1:i),'YData',y(i-1:i),'ZData',z(i-1:i))
    drawnow
    pause(0.01);
end

```

Lorenz_eq.m jest plikiem w którym zdefiniowano układ równań

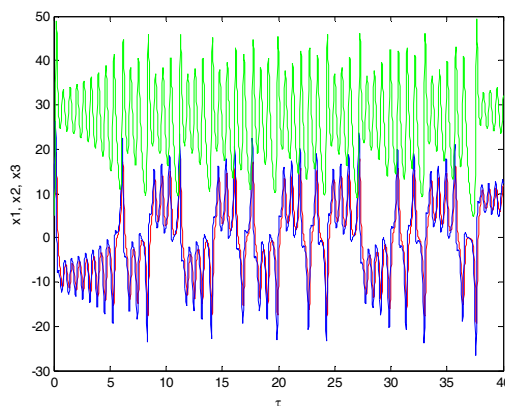
```

function dxdt=Lorenz_eq(t,x)
Pr=10;
b=8/3;
Ra=30;
dxdt=[ Pr*(-x(1)+x(2))
       Ra*x(1)-x(2)-x(1)*x(3)
       -b*x(3)+x(1)*x(2) ];

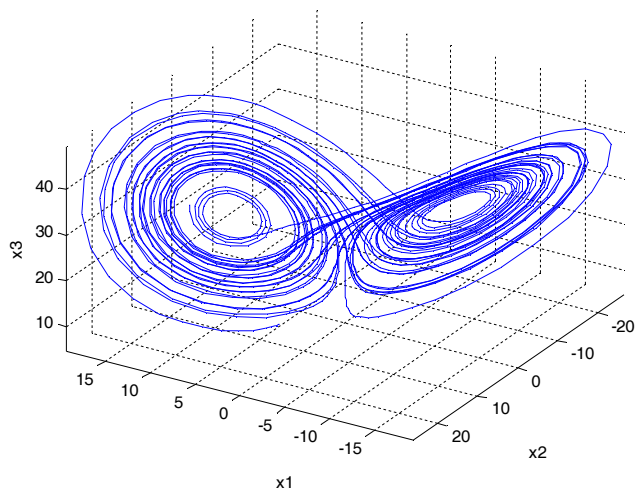
```

Rezultat 7.10

Program produkuje sześć wykresów: wykres czasowy, trajektoria na płaszczyźnie fazowej x_1, x_2 , trajektoria na płaszczyźnie fazowej x_1, x_3 , trajektoria na płaszczyźnie fazowej x_2, x_3 , trajektorii w przestrzeni fazowej i animowana trajektoria w przestrzeni fazowej. Pokazano tylko pierwszy i piąty z otrzymanych wykresów.



Rys. 7.15. Przebiegi czasowe rozwiązania



Rys. 7.16. Wykres rozwiązania równań Lorenza w przestrzeni fazowej

7.3.2. Modele procesów o postaci ODE w inżynierii procesowej

Dla obiektów o parametrach rozłożonych równanie bilansowe stosowane w inżynierii procesowej dla bilansowania pędu, ciepła i masy w różniczkowym elemencie przestrzeni ma postać:

$$\frac{\partial \Gamma}{\partial \tau} + \text{div}[\Gamma \cdot u] - \text{div}[D \cdot \text{grad} \Gamma] \pm \beta a \Delta \Gamma \pm G = 0, \quad (7.24)$$

gdzie Γ – wielkość bilansowana, u – wektor prędkości, D – współczynnik dyfuzji (dyspersji), β – współczynnik wymiany (o wymiarze m/s), a – powierzchnia właściwa w jednostce objętości, G – tempo produkcji (destrukcji) wielkości Γ . Człony tego równania noszą następujące nazwy (od lewej): akumulacja, człon konwekcyjny, człon dyfuzyjny, człon międzyfazowy, człon źródłowy.

Przykładowo, dla bilansu ciepła wielkością bilansowaną Γ jest entalpią liczoną na jednostkę objętości $\rho c_p T$.

$$\frac{\partial (\rho c_p T)}{\partial \tau} + \text{div}[\rho c_p T \cdot u] - \text{div}[E \cdot \text{grad}(\rho c_p T)] \pm \alpha a \Delta(T) \pm G = 0 \quad (7.25)$$

gdzie α – współczynnik wnikania ciepła. Dla jednowymiarowego przepływu w rurze z dyspersją wzdłużną, gdy brak członu źródłowego i akumulacji, przy założeniu niezależności c_p i ρ od temperatury, po wykonaniu działań mamy:

$$\rho c_p \frac{\partial T}{\partial \tau} + \rho c_p u \frac{\partial T}{\partial l} - E \rho c_p \frac{\partial^2 T}{\partial l^2} \pm \alpha \Delta T = 0. \quad (7.26)$$

Wyprowadzenie powyższych równań bilansowych dla pędu, ciepła i masy w geometrii kartezjańskiej, cylindrycznej i sferycznej jest zawarte w podręcznikach inżynierii chemicznej i procesowej.

Gdy proces przebiega w stanie nieustalonym lub gdy stan jest ustalony, lecz rozważamy przypadek dwu lub trójwymiarowy, z powyższego równania otrzymujemy równanie różniczkowe cząstkowe PDE. Przypadkiem takim zajmujemy się w Rozdziale 8. Gdy jednak proces jest ustalony a obiekt jednowymiarowy, uzyskujemy równanie różniczkowe zwyczajne z pochodną po wymiarze najwyższej drugiego rzędu, jeśli uwzględnimy człon dyfuzyjny. Jednowymiarowe przypadki procesów nieustalonych i ustalonych opisanych równaniami ODE przedstawiono poniżej.

7.3.2.1. Procesy nieustalone

PROJEKT 7.1

Temat:	Dynamika zmian wysokości cieczy w układzie trzech zbiorników
Toolbox:	-

Do pierwszego z trzech zbiorników połączonych kaskadowo dopływa woda z natężeniem Q_0 . Jest ono funkcją czasu podaną poniżej.

$$\begin{aligned} 1 \text{ przypadek} \quad Q &= Q_0 = \text{const} \\ 2 \text{ przypadek} \quad Q &= Q_0 \exp(-0.01\tau). \end{aligned} \quad (7.27)$$

Zbiorniki mają kształt walców i odpowiednie pola przekroju S_1 , S_2 , S_3 . Współczynniki wypływu wynoszą odpowiednio k_1 , k_2 i k_3 . Oblicz przebieg zmian wysokości cieczy w zbiornikach, jeśli początkowo były one puste.

Przyjmujemy, że natężenie wypływu cieczy przez otwór w dnie zależy od wysokości cieczy w zbiorniku w następujący sposób

$$Q = k\sqrt{h}. \quad (7.28)$$

Układ równań różniczkowych opisujących poziomy cieczy w zbiornikach ma wtedy następującą postać (Pakowski, Głębowski, 2001)

$$\begin{aligned} \frac{dh_1}{d\tau} &= \frac{Q_0 - k_1\sqrt{h_1}}{S_1} \\ \frac{dh_2}{d\tau} &= \frac{k_1\sqrt{h_1} - k_2\sqrt{h_2}}{S_2} \\ \frac{dh_3}{d\tau} &= \frac{k_2\sqrt{h_2} - k_3\sqrt{h_3}}{S_3} \end{aligned} \quad (7.29)$$

P7_1.m

```
%PROJEKT 7.1
%Dynamika poziomu cieczy w kaskadzie zbiorników
clear all; close all; clc
global S1 S2 S3 k1 k2 k3 ip
%przypadek ip=1 - stały przepływ, ip=2 - zmienny przepływ
ip=2;
%poła przekroju zbiorników [m2]
S1=0.8; S2=1; S3=1.3;
%stałe wypływu
k1=0.05; k2=0.04; k3=0.03;
% zakres czasu
t0=0; tk=450; t=[t0;tk];
% warunki początkowe
xp=[0; 0; 0];
% wywołanie integratora
[t,x]=ode113('kaskada_eq',t,xp);
% wizualizacja wyników
figure(1)
hold on
plot(t,x(:,1),'r+');
plot(t,x(:,2),'b+');
plot(t,x(:,3),'g+');
grid on;
title('Przebieg zmian czasie wysokości cieczy w zbiornikach')
```

```

xlabel('czas [s]')
ylabel('poziom cieczy [m]')
% interpolacja wyników
t_int=t'; %wektor czasu
xi=t0:0.01:tk;
yi1=interp1(t_int,x(:,1)',xi,'spline');
yi2=interp1(t_int,x(:,2)',xi,'spline');
yi3=interp1(t_int,x(:,3)',xi,'spline');
plot(xi,yi1,'k')
plot(xi,yi2,'k')
plot(xi,yi3,'k')
xf=x(:,1); y=x(:,2); z=x(:,3);
figure(2)
krzywa1 = plot3(xf,y,z,'r-','EraseMode','none','LineWidth',2,'MarkerSize',2,'MarkerFace','r');
axis([min(xf) max(xf) min(y) max(y) min(z) max(z)]);
view(64,33);
xlabel('h1')
ylabel('h2')
zlabel('h3')
grid on

```

W pliku **kaskada_eq.m** jest zapisany układ równań opisujący taki układ zbiorników. W takim układzie natężenie dopływu wody może być stałe lub zmienne. W tym celu zdefiniowano dwie funkcje opisujące natężenia dopływu wody Q_0 . Jeśli chcemy, aby dopływ był zmienny, musimy skasować % przed Q_0 w linii 4, a wstawić w linii 3.

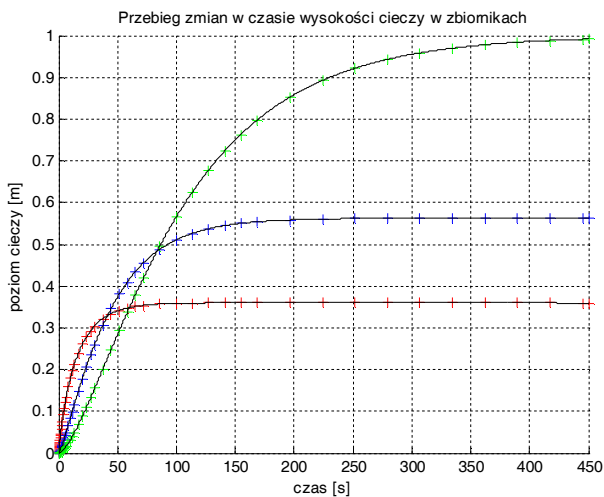
kaskada_eq.m

```

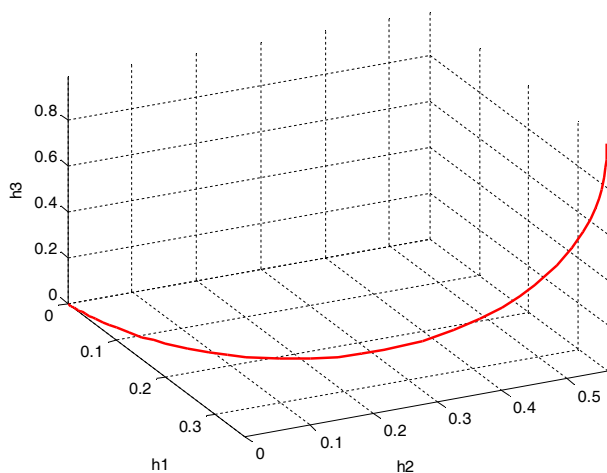
function y=kaskada_eq(t,x)
global S1 S2 S3 k1 k2 k3 ip
if ip==1
    Q0=30*(10^(-3)); %m^3/s - stały dopływ
end
if ip==2
    Q0=30*(10^(-3))*exp(-0.01*t); %m^3/s - zmienny dopływ
end
y=[ (Q0-k1*sqrt(x(1)))/S1
    (k1*sqrt(x(1))-k2*sqrt(x(2)))/S2
    (k2*sqrt(x(2))-k3*sqrt(x(3)))/S3 ];

```

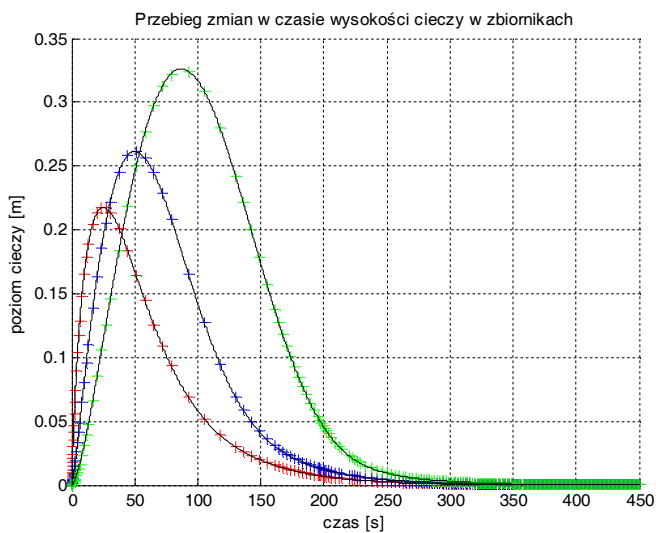
Wyniki 7.1



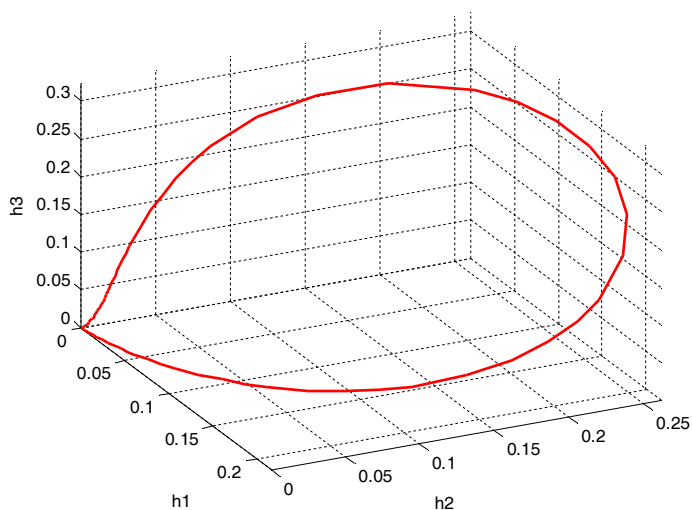
Rys. 7.17. Przebieg zmian wysokości cieczy w zbiornikach w czasie przy stałym natężeniu dopływu cieczy



Rys. 7.18. Wykres rozwiązania w przestrzeni fazowej



Rys. 7.19. Przebieg zmian wysokości cieczy w zbiornikach w czasie przy zmiennym natężeniu dopływu cieczy

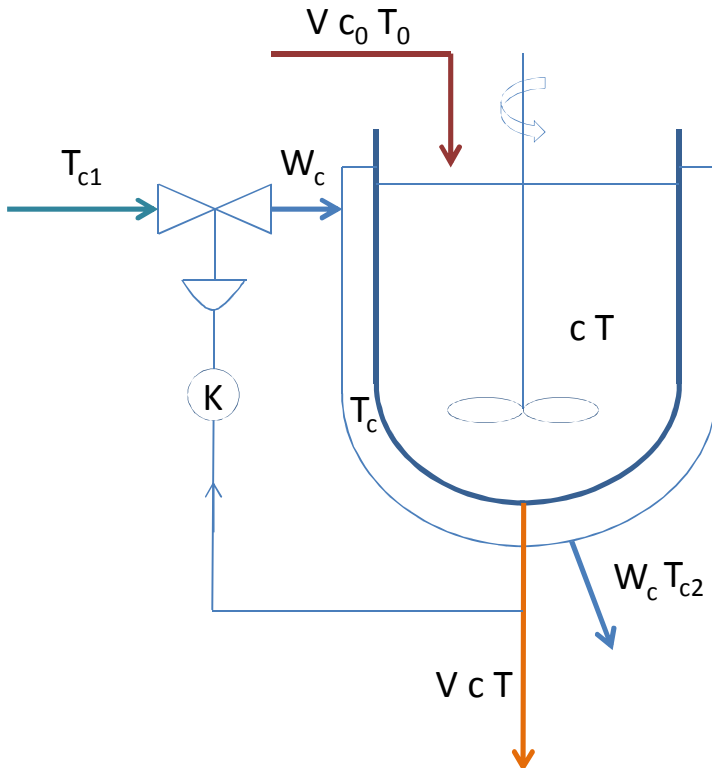


Rys. 7.20. Wykres rozwiązania w przestrzeni fazowej

PROJEKT 7.2

Temat: Stabilność reaktora z reakcją egzotermiczną
 Toolbox: -

Projekt ten oparto na przykładzie 4.3 z *Ramireza, 1997*. Dotyczy on reaktora przepływowego z idealnym wymieszaniem. W reaktorze jest prowadzona reakcja egzotermiczna katalitycznego rozkładu roztworu nadtlenu wodoru w wodzie. Roztwór dopływa ze stałym natężeniem i przy stałej temperaturze do reaktora, a po przereagowaniu odpływa z niego z tym samym natężeniem. Wydzielające się ciepło reakcji jest odbierane przez czynnik chłodzący przepływający przez płaszcz. Regulator steruje natężeniem cieczy chłodzącej na podstawie temperatury strumienia wylotowego. Ciecz chłodząca ma stałą temperaturę wlotową. Schemat instalacji reaktora pokazano na Rys. 7.21.



Rys. 7.21. Schemat reaktora zbiornikowego z chłodzeniem

Bilanse masy i ciepła dla zawartości reaktora mają postać

$$v \frac{\partial c}{\partial \tau} = -vr - V(c - c_0) \quad (7.30)$$

oraz

$$v\rho c_p \frac{\partial T}{\partial \tau} = -v\Delta h_r r - UA(T - T_c) - V\rho c_p(T - T_0). \quad (7.31)$$

Szybkość reakcji chemicznej oblicza się wg równania reakcji pierwszorzędowej jako

$$r = k_0 \exp\left(-\frac{E}{T}\right)c, \quad (7.32)$$

gdzie E [K] to energia aktywacji podzielona przez uniwersalną stałą gazową R . Drugi człon po prawej stronie równania 7.31 to strumień ciepła [kW] wymieniany pomiędzy zawartością reaktora o temperaturze T a cieczą chłodzącą o temperaturze T_c .

Współczynnik przenikania ciepła U [kW/(m²K)] w tym równaniu obliczany jest ze wzoru

$$U = C_1(C_4 W_c)^{1/3}, \quad (7.33)$$

gdzie W_c to natężenie przepływu cieczy chłodzącej [kg/s].

Różnicę temperatur pomiędzy temperaturami reagenta i czynnika chłodzącego, niezbędną do obliczenia strumienia ciepła oddawanego do płaszcza wyznacza się z bilansu ciepła tego czynnika. W tym celu zakłada się, że jego temperatura jest średnią arytmetyczną temperatury na wlocie T_{c1} i wylocie T_{c2} . Wtedy otrzymuje się

$$T - T_c = \frac{T - T_{c1}}{1 + \frac{UA}{2W_c c_{pc}}}. \quad (7.34)$$

Sterowanie odbywa się w następujący sposób:

- Na podstawie temperatury reagenta T oblicza się sygnał sterujący

$$G = K(T - T_s)C_2 + C_3 \quad (7.35)$$

gdzie K – współczynnik wzmocnienia [1/K], T_s – temperatura zadana, C_2 i C_3 współczynniki empiryczne.

Na podstawie tego sygnału oblicza się względne otwarcie zaworu

$$q = \frac{G - G_{\min}}{G_{\max} - G_{\min}} \cdot 100\% \quad (7.36)$$

Charakterystyka zaworu jest następująca:

$$\begin{aligned} \text{dla } q < 20\% \quad W_c &= 0.166 \\ \text{dla } q > 85.56\% \quad W_c &= 8 \cdot 10^{-5} \end{aligned} \quad (7.37)$$

$$\text{w pozostałym zakresie} \quad W_c = -3.871 \cdot 10^{-5} (q - 20)^2 + 0.166$$

Aby zbadać charakterystyki reaktora dla różnych wartości współczynnika wzmocnienia K wykonano wykres rozwiązania układu równań bilansowych 7.30 i 7.31 na płaszczyźnie fazowej stosując różne zestawy warunków początkowych stężenia i temperatury w reaktorze. Dane liczbowe i oznaczenia zawarte są w programie.

P7_2.m

%PROJEKT 7.2

%Wykres stabilności reaktora z reakcją egzotermiczną
%chłodzonego wodą (na podst. przykładu 4.3 z Ramirez, 1997)

function reaktor1

clear all; close all; clc

global K Ts Tc1 T0 A v Cpc Cp dhr k0 E To rho V c0

%najważniejsze dane regulatora - pozostałe w funkcji QA

K = 1; % współczynnik wzmocnienia

Ts = 30; % temperatura zadana C

%dane reakcji chemicznej

k0 = 0.084*10^10; % współczynnik kinetyczny, 1/s

E = 9310; % stała, K

dhr = 9.462e4; % ciepło reakcji, kJ/kmol

%dane reaktora

v = 16.2e-3; % objętość, m3

A = 0.259; % powierzchnia chłodzenia, m2

%dane cieczy chłodzącej

Tc1 = 10; % temperatura na wlocie, C

Cpc = 4.187; % ciepło właściwe, kJ/(kg K)

%dane strumienia reagenta

V = 8.333e-6; % natężenie dopływu, m3/s

c0 = 11.5; % wlotowe stężenie reagenta, kmol/m3

```

T0 = 26.8; % temperatura wlotowa, C
rho = 1108.1; % gęstość, kg/m3
Cp = 3.622; % ciepło właściwe, kJ/(kg K)
To = 273.15; % temperatura zera C, K
%zestaw warunków początkowych
pnts=[0 0 0 0 0 2 6 10 14 16 16 16 16 10 10 %stężenie
      0 20 40 60 70 0 0 0 0 20 30 40 60 60 65];%temperatura
%wykres na płaszczyźnie fazowej
figure(1)
hold on
plotall(pnts);
axis([0 80 0 16]);
grid on
h=xlabel('Temperatura [\circ C]')
set(h)
ylabel('Stężenie [mol/l]');
%punkt stacjonarny
pnts=[0;0];
pk=plotall(pnts); %punkt końcowy tej trajektorii
plot(pk(2),pk(1),'ro');
text(8,15,['K = ' num2str(K)]);

function y=plotall(pnts)
[~,n]=size(pnts);
czasmax=1e5; %maksymalny czas w sekundach
for i=1:n
    z0=pnts(:,i);
    [~,z]=ode15s(@uk_row,[0 czasmax],z0);
    conc=z(:,1); temp=z(:,2);
    plot(temp,conc)
end
m=length(z);
y=z(m,:); %ostatni punkt trajektorii

function y=uk_row(~,z)
global dhr k0 E To rho V c0 T0 v Cp
c=z(1,:);
T=z(2,:);
dC = (-v*k0*exp(-E./(T+To)).*c - V.*(c-c0))./v;
dT = (dhr*v*k0*exp(-E./(T+To)).*c...
      - QA(T)-rho*V*Cp*(T-T0))./(v*rho*Cp);
y=[dC;dT];

```

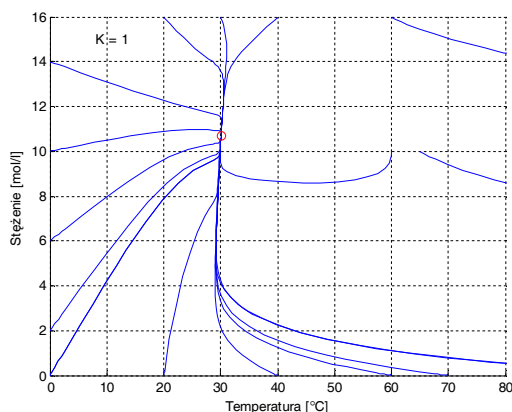
```

function y=QA(T)
global K Ts Tc1 A Cpc
C1 = 0.3549;           % stała
C2 = 0.1068;           % stała
C3 = 17.55;            % stała
C4 = 132.159;          % stała
Gmin = 4;              % -
Gmax = 20;             % -
G = -K*(T-Ts)*C2+C3;    % wartość sygnału sterującego
q = 100*(G-Gmin)./(Gmax-Gmin); % stopień otwarcia zaworu
%charakterystyka zaworu
if q<20%
    Wc=0.166; %[kg/s]
elseif q>85.56%
    Wc=8e-5; %[kg/s]
else
    Wc=-3.871e-5*(q-20).^2 + 0.166; %[kg/s]
end
%Współczynnik przenikania ciepła
U=C1*(C4*Wc).^(1./3);    %U - kW/(m2 K)
DT=(T-Tc1)./(1+U*A./(2*Wc*Cpc)); %DT - degC
y=U*A*DT;

```

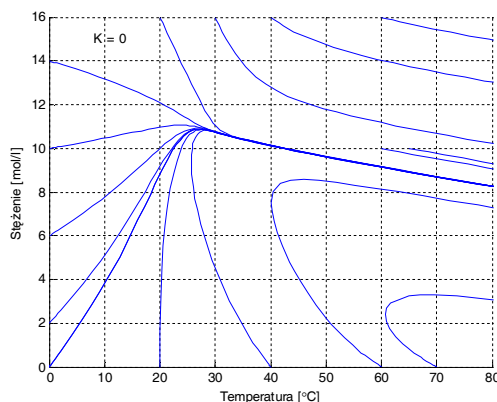
Wyniki 7.2

Dla współczynnika wzmocnienia $K = 1$ otrzymuje się następujący wykres:



Rys. 7.22. Wykres na płaszczyźnie fazowej dla $K = 1$. Kółko oznacza węzeł stabilny

Jak widać, gdy sterowanie jest włączone dla części stanów początkowych uzyskujemy stan stacjonarny, w którym temperatura utrzymuje się na zadanej wartości 30°C , a stężenie w reaktorze osiąga wartość ok. 10.5 mola/l. Gdy jednak przy wyższych stężeniach temperatura przekracza ok. 60°C reaktor zaczyna się przegrzewać. W przypadku wyłączonego sterowania (brak chłodzenia) zawartość reaktora przegrzewa się zawsze, gdyż wszystkie trajektorie dążą ostatecznie w kierunku rosnącej temperatury, nie osiągając stanu stacjonarnego.



Rys. 7.23. Wykres na płaszczyźnie fazowej dla $K = 0$

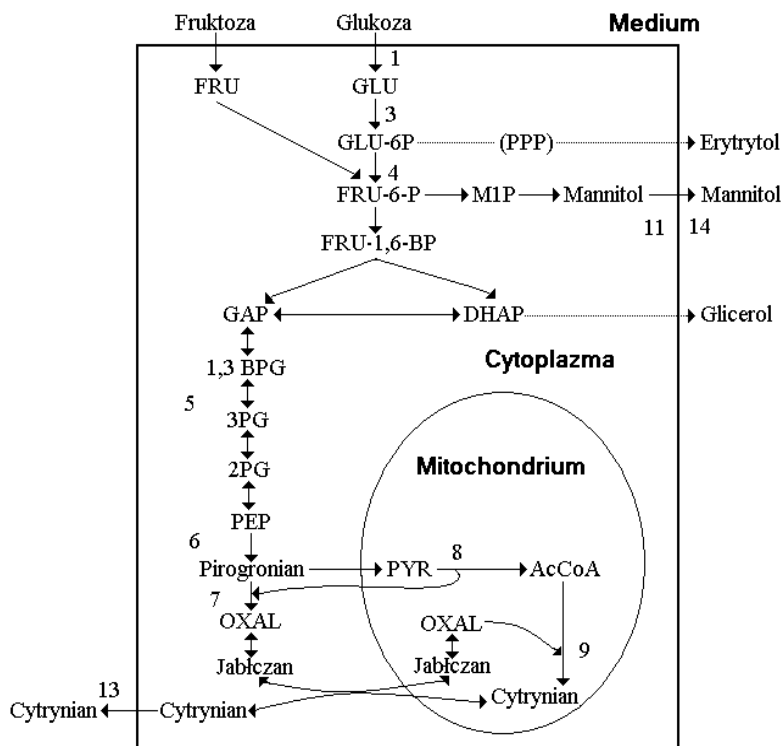
PROJEKT 7.3

Temat: Model kinetyczny procesu wytwarzania kwasu cytrynowego przez *Aspergillus Niger*

Toolbox: -

Projekt ilustruje rozwiązywanie typowych modeli kinetycznych złożonych reakcji chemicznych. Należy rozwiązać model kinetyczny wytwarzania kwasu cytrynowego przez *A. Niger* sformułowany w pracy Krzystek, 2002, pozwalającego przewidywać przebiegi czasowe stężeń poszczególnych reagentów w danych warunkach operacyjnych wg schematu reakcji pokazanego na Rys. 7.24. Model obejmuje procesy zachodzące podczas wzrostu *A. Niger*, procesy zachodzące podczas produkcji kwasu cytrynowego oraz uwzględnia tworzenie wyższych alkoholi.

Składa się on z 16 równań bilansowych typu ODE. Warunki początkowe dla tego układu równań zawiera Tabela 7.3.



Rys. 7.24. Uproszczony schemat metaboliczny biosyntezy kwasu cytrynowego przez *Aspergillus Niger* (wg Krzysiek, 2010)

Tablica 7.3. Warunki początkowe

Metabolit	Stężenie początkowe [mmol]
S – substrat	584
GLU – glukoza	0.2
GLU6P – glukoza-6-fosforan	0.2
FRU6P – fruktoza-6-fosforan	0.07
PEP – fosfoenolopirogronian	0.5
PYR – pirogronian	0.5
AcCoA – acetylo-koenzym A	0.005
CIT – cytrynian	0.00127
OXAL – szczawiooctan	0.00029
CO ₂ – dwutlenek węgla	0.016
AMON – wewnątrzkomórkowa pula jonów amonowych	0.0001

CoA – koenzym A	0.0001
X - biomasa	1
amon – zewnątrzkomórkowa pula jonów amonowych	32
POL – Alkohole wyższe	0.0001
Stężenia przyjęte za stałe	
NAD	0.12
NADH2/NADPH2	0.035
ATP	0.5
ADP	0.075

Równania bilansowe modelu:

– zmiana ilości substratu:

$$\frac{d(S)}{d\tau} = -r_{GLU}X \quad (7.38)$$

– zmiana ilości jonów amonowych:

$$\frac{d(amon)}{d\tau} = -r_{amon}X \quad (7.39)$$

– zmiana ilości glukozy:

$$\frac{d(GLU)}{d\tau} = r_{GLU} - r_{GLU6P} - 0.163r_{TOT} - 0.017r_{POL} - \mu GLU \quad (7.40)$$

– zmiana ilości glukozo-6-fosforanu

$$\frac{d(GLU6P)}{d\tau} = 0.82(r_{GLU6P} - r_{FRU6P}) - \mu GLU6P \quad (7.41)$$

– zmiana ilości fruktozo-6-fosforanu:

$$\frac{d(FRU6P)}{d\tau} = 0.82(r_{FRU6P} - r_{PEP}) - \mu FRU6P \quad (7.42)$$

– zmiana ilości fosfoenolopirogronianu:

$$\frac{d(PEP)}{d\tau} = 1.64(r_{PEP} - r_{PYR}) - \mu PEP \quad (7.43)$$

– zmiana ilości pirogronianu:

$$\frac{d(PYR)}{d\tau} = 1.64r_{PYR} - 0.82r_{AcCoA} - 0.82r_{OXAL} - \mu PYR \quad (7.44)$$

– zmiana ilości koenzymu A:

$$\frac{d(CoA)}{d\tau} = 0.82(r_{CIT} - r_{AcCoA}) - \mu CoA \quad (7.45)$$

– zmiana ilości acetylo-koenzymu A:

$$\frac{d(AcCoA)}{d\tau} = 0.82(r_{AcCoA} - r_{CIT}) - \mu AcCoA \quad (7.46)$$

– zmiana ilości szczawiooctanu:

$$\begin{aligned} \frac{d(OXAL)}{dt} &= 0.82(r_{OXAL} - r_{CIT}) - \mu OXAL \\ \frac{d(OXAL)}{d\tau} &= 0.82(r_{OXAL} - r_{CIT}) - \mu OXAL \end{aligned} \quad (7.47)$$

– zmiana ilości wewnątrzkomórkowej puli jonów amonowych:

$$\frac{d(AMON)}{dt} = r_{AMON} - \mu AMON \quad (7.48)$$

– zmiana ilości biomasy:

$$\frac{d(X)}{d\tau} = -\mu X \quad (7.49)$$

– zmiana ilości dwutlenku węgla:

$$\frac{d(CO_2)}{d\tau} = (0.82r_{AcCoA} - 0.82r_{OXAL} + 0.163r_{TOT} + 0.043r_{POL})X \quad (7.50)$$

– zmiana ilości cytrynianu:

$$\frac{d(CIT)}{d\tau} = 0.82r_{CIT}X \quad (7.51)$$

– zmiana ilości alkoholi wyższych:

$$\frac{d(POL)}{d\tau} = 0.016r_{POL}X \quad (7.52)$$

– suma szybkości tworzenia wszystkich składników biomasy:

$$\mu = r_{GLU} + r_{AMON} + 0.82r_{PEP} - 0.82r_{AcCoA} - 0.163r_{TOT} - 0.017r_{POL} \quad (7.53)$$

Szybkości r poszczególnych reakcji są obliczane wg wzorów, których tu nie przytaczamy. Są one zawarte w pliku **bilans.m**.

P7_3.m

```
%Kinetyka układu reakcji biochemicznych
clear all; close all; clc
t0=0;          %początek czasu
tk=80*3600; %koniec czasu
%warunki początkowe
XP=[584;32;0.2;0.2;0.07;0.5;0.5;0.0001;0.005;0.00029;0.0001;1;0.
016;0.00127;0.0001];
%rozwiązanie modelu
[t,X]=ode23tb(@bilans,[t0 tk],XP);
figure(1)
plot(t/3600,X(:,2),'LineWidth',2)
title('Stężenie jonów amonowych (zewnatrzkomórkowych)')
xlabel('czas [h]')
ylabel('stężenie [mmol/l]')
grid on
figure(2)
plot(t/3600,X(:,14)*10,'LineWidth',2)
title('Stężenie kwasu cytrynowego')
xlabel('czas [h]')
ylabel('stężenie [mmol/l]')
grid on
figure(3)
plot(t/3600,X(:,15)*10,'LineWidth',2)
title('Stężenie wyższych alkoholi')
xlabel('czas [h]')
ylabel('Stężenie [mmol/l]')
grid on
figure(4)
plot(t/3600,X(:,10),'LineWidth',2)
title('Stężenie szczawiooctanu')
xlabel('czas [h]')
```

```
ylabel('Stężenie [mmol/l]')
grid on
```

bilans.m to funkcja, w której zdefiniowano w/w układ 15 równań różniczkowych.

bilans.m

```
function y=bilans(~,X)
kaGLU11=0.0000437; %stała szybkości transportu glukozy względem
substratu [mmol/min*mg]
KS11=0.26; %stała nasycenia względem substratu [mmol]
ICIT11=3.5; %stała inhibicji względem cytrynianu [mmol]
kaGLU21=0.00012102; %stała szybkości transportu glukozy względem
substratu [mmol/min*mg]
KS21=3.67; %stała nasycenia względem substratu [mmol]
ICIT21=5.5; %stała inhibicji względem cytrynianu [mmol]
kaNH42=0.000315; %stała szybkości transportu jonów amonowych
[mmol/min*mg]
KNH42=160; %stała inhibicji [mmol]
kaGLU6P3=0.000016; %stała szybkości heksokinazy [mmol/min*mg]
KGLU3=0.27; %stała nasycenia heksokinazy względem
glukozy [mmol]
KATP3=0.15; %stała nasycenia heksokinazy względem ATP
[mmol]
kaFRU6P4=0.0153; %stała szybkości izomerazy fosfoglukozy
[mmol/min*mg]
KGLU6P4=420; %stała nasycenia izomerazy fosfoglukozy
względem glukozo-6-fosforanu [mmol]
KFRU6P4=103; %stała nasycenia izomerazy fosfoglukozy
względem fruktozo-6-fosforanu [mmol]
kaPEP5=0.000075; %stała szybkości fosfofruktokinazy 1
[mmol/min*mg]
KFRU6P5=0.25; %stała nasycenia fosfofruktokinazy 1
względem fruktozo-6-fosforanu [mmol]
KATP5=0.03; %stała nasycenia fosfofruktokinazy 1
względem ATP
IPEP5=0.2; %stała inhibicji fosfofruktokinazy 1
względem fosfoenolopirogronianu [mmol]
kaPYR6=0.00015; %stała szybkości kinazy pirogronianowej
[mmol/min*mg]
KPEP6=0.1; %stała nasycenia kinazy pirogronianowej
względem fosfoenolopirogronianu [mmol]
KADP6=0.07; %stała nasycenia kinazy pirogronianowej
względem ADP [mmol]
```

```

KAAMON6=80;           %stała KA - wewnątrzkomórkowej puli jonów
amonowych
KBAMON6=0.01;         %stała KB - wewnątrzkomórkowej puli jonów
amonowych
kaOXAL7=0.00015;      %stała szybkości karboksylazy
pirogronianowej [mmol/min*mg]
KPYR7=0.28;           %stała nasycenia karboksylazy
pirogronianowej względem pirogronianu [mmol]
KATP7=0.21;           %stała nasycenia karboksylazy
pirogronianowej względem ATP [mmol]
KC027=1.33;           %stała nasycenia karboksylazy
pirogronianowej względem CO2 [mmol]
kaCOA8=0.00037;       %stała szybkości dehydrogenazy
pirogronianowej [mmol/min*mg]
KPYR8=0.16;           %stała nasycenia dehydrogenazy
pirogronianowej względem pirogronianu [mmol]
KCOA8=0.0011;         %stała nasycenia dehydrogenazy
pirogronianowej względem koenzymu A [mmol]
KNAD8=0.077;          %stała nasycenia dehydrogenazy
pirogronianowej względem NAD [mmol]
INADH28=0.009;        %stała inhibicji dehydrogenazy
pirogronianowej względem NADH2 [mmol]
kaCIT9=0.0042;        %stała szybkości syntazy cytrynianowej
[mmol/min*mg]
KOXAL9=0.005;         %stała nasycenia syntazy cytrynianowej
względem szczawiooctanu [mmol]
KCOA9=0.01;           %stała nasycenia syntazy cytrynianowej
względem acetylokoenzymu A [mmol]
ICOA9=0.015;          %stała inhibicji syntazy cytrynianowej
względem koenzymu A [mmol]
kaTOT10=3.44e-6;      %stała szybkości całkowitego spalania
glukozy [mmol/min*mg]
KGLU10=12;            %stała nasycenia całkowitego spalania
glukozy względem glukozy [mmol]
kaPOL11=5.15e-6;      %stała szybkości syntezy alkoholi wyższych
[mmol/min*mg]
KGLU6P11=0.037;       %stała nasycenia syntezy alkoholi wyższych
względem glukozo-6-fosforanu [mmol]
KATP11=0.15;          %stała nasycenia syntezy alkoholi wyższych
względem ATP [mmol]

%----szybkość transportu glukozy
rGLU1A=kaGLU11*(X(1)/(X(1)+KS11));

```

```

%----szybkość transportu glukozy
rGLU1=(kaGLU11*(X(1)/(X(1)+KS11))*(ICIT11/(ICIT11+X(14))))+(kaGLU21*(X(1)/(X(1)+KS21))*(ICIT21/(ICIT21+X(14))));
%----szybkość transportu jonów amonowych
rNH42=kaNH42*((X(2)/(X(2)+KNH42))^2);
%----szybkość syntezy glukozy-6-fosforanu
rGLU6P3=kaGLU6P3*((X(3)/(X(3)+KGLU3))*(0.5/(0.5+KATP3)));
%----szybkość izomeryzacji glukozy-6-fosforanu
rFRU6P4=kaFRU6P4*((X(4)-(X(5)/0.4))/(KGLU6P4*(1+(X(5)/KFRU6P4))+X(4)));
%----szybkość syntezy fosfoenolopirogronianu
rPEP5=kaPEP5*(((X(5))^4)/(((X(5))^4)+KFRU6P5))*(0.5/(0.5+KATP5))*(IPEP5/(X(6)+IPEP5));
%----szybkość syntezy pirogronianu
rPYR6=kaPYR6*(X(6)/(X(6)+KPEP6))*(0.075/(0.075+KADP6))*((X(11)+KAAMON6)/(X(11)+KBAMON6));
%----szybkość anaplerotycznego wiązania CO2 i synteza
szczawiooctanu
rOXAL7=kaOXAL7*(X(7)/(X(7)+KPYR7))*(X(13)/(X(13)+KCO27))*(0.5/(0.5+KATP7));
%----szybkość syntezy acetylokoenzymu A
rCOA8=kaCOA8*(X(7)/(X(7)+KPYR8))*(X(8)/(X(8)+KCOA8))*(0.075/(0.075+(KNAD8*(1+(0.035/INADH28)))));
%----szybkość syntezy cytrynianu
rCIT9=(kaCIT9*(X(10)/(X(10)+KOXAL9))*(X(9)/(X(9)+KCOA9))*(ICOA9/(X(8)+ICOA9)));
%----szybkość całkowitego spalania glukozy
rTOT10=kaTOT10*(X(3)/(X(3)+KGLU10));
%----szybkość syntezy alkoholi wyższych
rPOL11=kaPOL11*(X(4)/(X(4)+KGLU6P11))*(0.5/(0.5+KATP11));
%----szybkość syntezy alkoholi wyższych
ni=rGLU1A+rPYR6+0.82*rPEP5-0.82*rCOA8-0.82*rCIT9-0.163*rTOT10-0.017*rPOL11;

%Poniższy układ równań przedstawia bilanse masy:
y=[-rGLU1A*X(12)
-rNH42*X(12)
rGLU1A-0.82*rGLU6P3-0.163*rTOT10-0.017*rPOL11-ni*X(3)
0.82*(rGLU6P3-rFRU6P4)-ni*X(4)
0.82*(rFRU6P4-rPEP5)-ni*X(5)
1.64*(rPEP5-rPYR6)-ni*X(6)
1.64*rPYR6-0.82*rCOA8-0.82*rOXAL7-ni*X(7)
0.82*(-rCOA8+rCIT9)-ni*X(8)

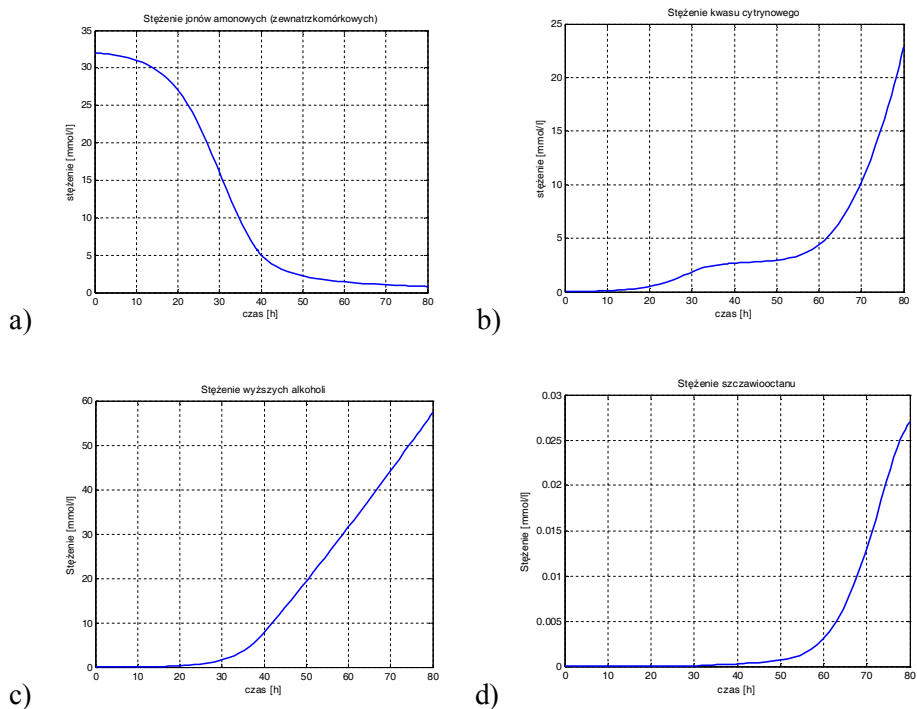
```

```

0.82*(rCOA8-rCIT9)-ni*X(9)
0.82*(rOXAL7-rCIT9)-ni*X(10)
rPYR6-ni*X(11)
ni*X(12)
(0.82*rCOA8-0.82*rOXAL7+0.163*rTOT10+0.043*rPOL11)*X(12)
0.82*rCIT9*X(12)
0.016*rPOL11*X(12)];

```

Wyniki 7.3



Rys. 7.25. Wykres: a) zmian stężenia jonów amonowych w czasie: b) zmian stężenia kwasu cytrynowego czasie, c) zmian stężenia wyższych alkoholi czasie, d) zmian stężenia szczawiooctanu w czasie

PROJEKT 7.4

Temat: Nieustalona dyfuzja wieloskładnikowa
Toolbox: -

Zadanie sformułowali *Wessenlingh i Krishna, 2006*. Projekt opisuje przypadek nieustalanej dyfuzji wieloskładnikowej w układzie dwu zamkniętych zbiorników połączonych kapilarą. Oba zbiorniki mają tę samą objętość V , kapilara ma długość L i pole przekroju S . Składniki układu to wodór (1), dwutlenek węgla (2) i azot (3). Początkowo w zbiorniku A znajdowały się tylko składniki 1 i 3, a w zbiorniku B składniki 2 i 3 w równych ilościach molowych. Należy wyznaczyć przebieg zmian stężenia w obu zbiornikach w czasie oraz przebieg zmian szybkości molowej dyfuzji składnika 3 w czasie.

Układając bilanse masy dla zbiornika A, mamy

$$\begin{aligned} V \frac{dx_{1A}}{d\tau} &= -\frac{S}{L} [D_{1,1}(x_{1A} - x_{1,B}) + D_{1,2}(x_{2A} - x_{2,B})] \\ V \frac{dx_{2A}}{d\tau} &= -\frac{S}{L} [D_{2,1}(x_{1A} - x_{1,B}) + D_{2,2}(x_{2A} - x_{2,B})], \end{aligned} \quad (7.54)$$

gdzie D to macierz dyfuzyjności wieloskładnikowych Ficka obliczana na podstawie wzoru 3.4. Aktualną wartość ułamka molowego x_{2A} obliczymy z zależności

$$x_{3A} = 1 - x_{1A} - x_{2A}. \quad (7.55)$$

Ponieważ układ jest zamknięty, to akumulacje w zbiorniku B są równe tym określonym równaniami 7.32, ale z przeciwnym znakiem.

P7_4.m

%PROJEKT 7.8

%Nieustalona dyfuzja trójskładnikowa między dwoma zbiornikami

%Ilustracja zjawiska odwróconej dyfuzji

clear all; close all; clc;

global DSM a C L

format short

%składniki 1=H2 2=CO2 3=N2

xA0=[0.5 0 0.5]; %początkowe ułamki molowe w zb. A

xB0=[0 0.5 0.5]; %początkowe ułamki molowe w zb. B

M= [2 44 28]; %masy molowe składników, kg/kmol

v= [4.62 28.12 9.08]; %objętości dyfuzyjne we wzorze Fullera


```

d=0.005;           %średnica kapilary, m
L=0.1;            %długość kapilary, m
V=1e-3;           %objętość każdego zbiornika, m3
T=25;             %temperatura procesu, C
T0=273.15;
R=8.314;
P0=101.325;        %ciśnienie procesu, kPa
a=pi*d.^2./4./V./L;

%obliczenie dyfuzyjności binarnych SM ze wzoru Fullera
D=eye(3);
for i=1:3
    for j=1:3
        if i==j
            D(i,j)=0;
        else
            Msr=sqrt(2*(1./M(i)+1./M(j)).^-1);
            v23=(v(i).^0.33333+v(j).^0.33333).^2;
            D(i,j)=(T+T0).^1.75./(P0*Msr*v23);
        end
    end
end

DSM=D*1.43e-5;     %dyfuzyjność SM, m2/s
% rozwiązanie równań bilansowych
x0=[xA0(1) xA0(2) xB0(1) xB0(2)];
[t,x]=ode23(@uk_row,[0 3e5],x0);
t=t/3600;
xA=x(:,1:2);
xA=[xA(:, :) 1-xA(:,1)-xA(:,2)];
xB=x(:,3:4);
xB=[xB(:, :) 1-xB(:,1)-xB(:,2)];
%Wykresy
figure(1)
plot(t,xA)
grid on
legend('x1','x2','x3',1)
xlabel('Czas [h]'); ylabel('Ułamek molowy [-]');
title('ZMIANY STĘŻENIA W ZBIORNIKU A')
figure(2)
plot(t,xB)
grid on
legend('x1','x2','x3',1)

```

```

xlabel('Czas [h]'); ylabel('Ułamek molowy [-]');
title('ZMIANY STĘŻENIA W ZBIORNIKU B')
%Obliczenie szybkości molowej dyfuzji
C=P0./(R*(T+T0));
N3=rate3(xA,xB);
figure(3)
plot(t,N3);
hold on
plot(t,N3*0,'r-') %linia zero
grid on
xlabel('Czas [h]');
ylabel('Szybkość dyfuzji składnika 3 [kmol/(m2 s)]');

```

Układ równań jest w pliku **uk_row.m**

 **uk_row.m**

```

function y=uk_row(t,z)
global a
xA=[z(1) z(2) 1-z(1)-z(2)];
xB=[z(3) z(4) 1-z(3)-z(4)];
%dyfuzyjność obliczymy dla średnich stężeń na drodze dyfuzji
x=0.5*(xA+xB);
D=DF(x);

A(1)=-(D(1,1)*(z(1)-z(3))+D(1,2)*(z(2)-z(4)))*a;
A(2)=-(D(2,1)*(z(1)-z(3))+D(2,2)*(z(2)-z(4)))*a;
A(3)=-A(1);
A(4)=-A(2);
y=A';

```

Dyfuzyjność wieloskładnikową Ficka obliczono w pliku **DF.m**

 **DF.m**

```

function y=DF(x)
%macierz wieloskładnikowych dyfuzyjności Ficka
global DSM
B=eye(2);
B(1,1)=x(1)./DSM(1,3)+x(2)./DSM(1,2)+x(3)./DSM(1,3);
B(1,2)=-x(1)*(1./DSM(1,2)-1./DSM(1,3));
B(2,1)=-x(2)*(1./DSM(2,1)-1./DSM(2,3));
B(2,2)=x(1)./DSM(2,1)+x(2)./DSM(2,3)+x(3)./DSM(2,3);
y=B^-1;

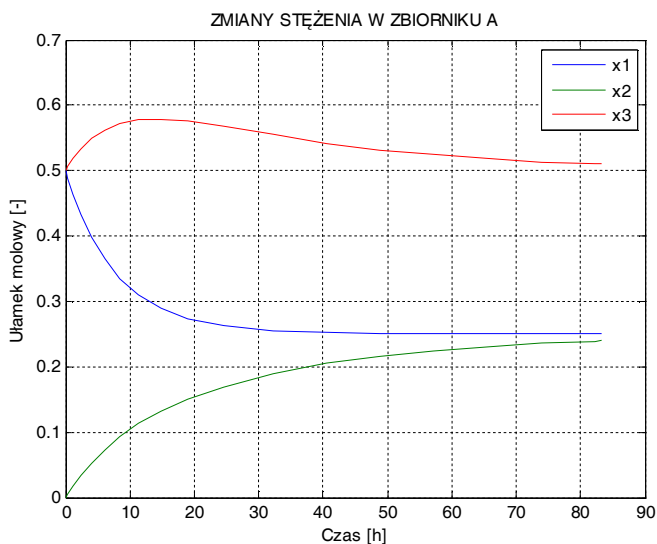
```

Szybkość molową dyfuzji obliczana jest w pliku **rate3.m**

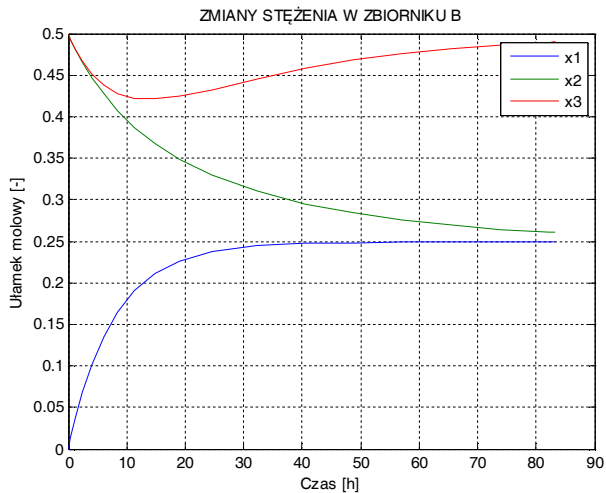
 **rate3.m**

```
function y=rate3(xA,xB)
%funkcja oblicza szybkości molowe dyfuzji
global C L
xm=(xA+xB)./2;
n=length(xm);
for i=1:n
D=DF(xm(i,:));
N1=(D(1,1)*(xA(i,1)-xB(i,1))+D(1,2)*(xA(i,2)-xB(i,2)))*C./L;
N2=(D(2,1)*(xA(i,1)-xB(i,1))+D(2,2)*(xA(i,2)-xB(i,2)))*C./L;
N3(i)=-N1-N2;
end
y=N3;
```

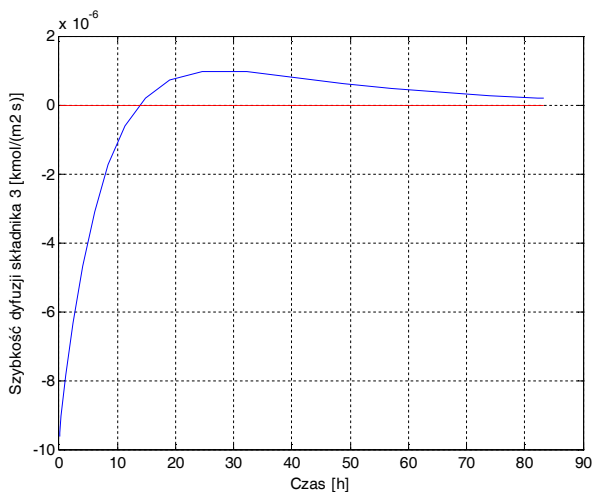
Wyniki 7.4



Rys. 7.26. Zmiany ułamków molowych składników w zbiorniku A



Rys. 7.27. Zmiany ułamków molowych składników w zbiorniku B



Rys. 7.28. Zmiany szybkości molowej dyfuzji składnika 3

Jak widać na Rys. 7.28, szybkość dyfuzji składnika 3 początkowo dyfunduje ze zbiornika B do A (ujemna szybkość dyfuzji) mimo tego, że jego ułamki molowe w obu zbiornikach są takie same. Do ok. 13 godziny składnik 3 dyfunduje przeciwko swojemu gradientowi stężenia. Później jednak zmienia kierunek dyfuzji i dyfunduje z A do B, zgodnie ze swoim gradientem stężenia.

7.3.2.2. Procesy ustalone

W równaniach opisujących procesy ustalone brak akumulacji, a pochodne powstają w wyniku bilansowania różniczkowego elementu objętości.

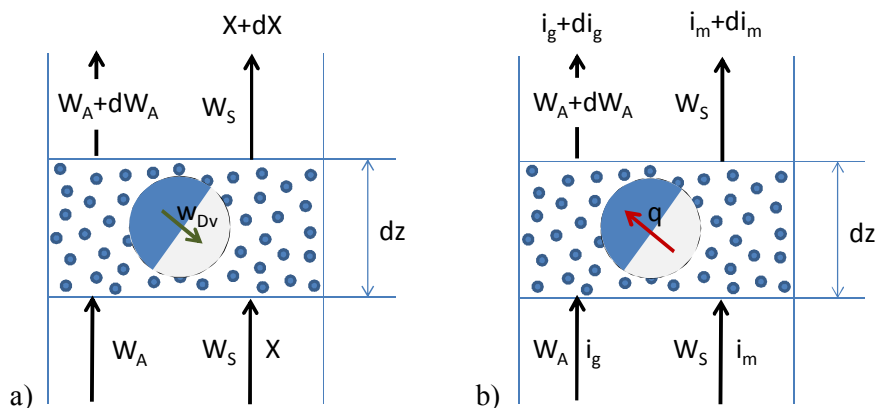
PROJEKT 7.5

Temat: Model suszarki pneumatycznej na parę przegrzaną

Toolbox: -

W suszarce pneumatycznej suszona jest biomasa w formie rozdrobnionego drewna wierzby energetycznej w strumieniu pary przegrzanej. Należy sformułować i rozwiązać model procesu umożliwiające obliczenie rozkładów prędkości obu faz, ich temperatur i zawartości wilgoci fazy stałej wzdłuż długości suszarki. Przykładowe dane są zawarte w programie.

Do ułożenia bilansów masy i ciepła należy wziąć element różniczkowy objętości suszarki w formie odcinka o wysokości dz i przekroju poprzecznym S . Oś wysokości z jest skierowana ku górze (Rys. 7.29).



Rys. 7.29. Schemat strumieni masy (a) i ciepła, (b) w suszarce pneumatycznej

Ułożone bilanse mają postaci pokazane poniżej (Pakowski i in., 2002, Pakowski, 2012).

Bilans masy dla fazy stałej:

$$\frac{dX}{dz} = -w_{Dv} \frac{S}{W_S} \quad (7.56)$$

Bilans masy dla fazy parowej:

$$\frac{dW_A}{dz} = w_{Dv} S. \quad (7.57)$$

Bilans ciepła dla fazy stałej:

$$\frac{dt_m}{dz} = \frac{S}{W_S} \frac{\alpha_v (T_g - T_m) - w_{Dv} \Delta h_v (T_m)}{c_s + c_{Al} X}. \quad (7.58)$$

Bilans ciepła dla fazy parowej:

$$\frac{dt_g}{dz} = - \frac{S}{W_S} \frac{\alpha_v (T_g - T_m) - w_{Dv} [\Delta h_v (T_m) + c_A (T_m - T_g)]}{c_A}. \quad (7.59)$$

Bilans pędu dla cząstki (indeks p oznacza cząstkę, m – fazę stałą):

$$\frac{du_p}{dz} = \frac{3}{4} \frac{1}{\rho_p u_p} \frac{c_D (u_g - u_p)^2}{d_p} \varepsilon^{-2.67} - \frac{g}{u_p} \frac{\rho_p - \rho_g}{\rho_p}. \quad (7.60)$$

Zasada ciągłości fazy parowej:

$$\frac{du_g}{dz} = \frac{1}{\rho_p \varepsilon} \left(\varepsilon u_g \frac{dT_g}{dz} \left(\frac{P_0 M_A}{RT_g^2} \right) + w_{Dv} - \frac{u_g \rho_g}{u_p \rho_p} (1 - \varepsilon) \left(u_p \frac{d\rho_p}{dX} \frac{dX}{dz} + \rho_p \frac{du_p}{dz} \right) \right) \quad (7.61)$$

Szybkość suszenia oblicza się z następujących równań konstytutywnych:

– szybkość suszenia w okresie nagrzewania/kondensacji i I okresie

$$w_{Dv,I} = \begin{cases} \alpha_v \frac{T_s - T_m}{\Delta h_v} & \text{gd}y \ T_m < T_s \\ \alpha_v \frac{T_g - T_s}{\Delta h_v} & \text{gd}y \ T_g > T_s \end{cases} \quad (7.62)$$

– szybkość suszenia w II okresie

$$w_{DvII} = w_{DvI} \left(\frac{X - X_{eq}}{X_{cr} - X_{eq}} \right)^n, \text{ gdy } X < X_{cr}. \quad (7.63)$$

Oznaczenia zmiennych w powyższych równaniach są zawarte w pliku głównym programu.

P7_5.m

%PROJEKT 7.5

%Model suszarki pneumatycznej na parę przgrzaną

clear all; close all; clc; memory;

global S Ws cs cAL Xcr Xeq n ts t0 mip R Mw g dp ff Pat lambp

% DANE RUCHOWE

Wm=2.0; % natężenie przepływu mokrej biomasy [T/h]

b1=0.6; % wilgotność początkowa [%]

b2=0.085; % wilgotność końcowa [%]

tm0=15; % początkowa temperatura materiału [C]

X1=btoX(b1); % wilgotność początkowa [kg/kg]

X2=btoX(b2); % wilgotność końcowa [kg/kg]

Ws=Wm*(1000/3600)*(1/(1+X1));
% przepływ suszonego materiału [kg/s]

WA1=2.5; % natężenie przepływu pary [kg/s]

tg0=500; % początkowa temperatura pary [C]

% WARTOŚCI ODCZYTANE Z KRZYWEJ RÓWNOWAGI

Xcr=0.4; % wilgotność krytyczna [kg/kg]

Xeq=0.081; % wilgotność równowagowa [kg/kg]

% GEOMETRIA SUSZARKI

D=0.6; % średnica suszarki [m]

L=40; % długość rury [m]

S=(pi*D^2)/4; % przekrój poprzeczny rury [m^2]

% WŁASNOŚCI FIZ-CHEM

cAL=4.190; % ciepło właściwe wody [kJ/kg*K]

cs=1.47; % ciepło właściwe ciała stałego [kJ/kg*K]

ts=100; % temperatura wrzenia wody [C]

g=9.81; % przyspieszenie ziemskie [m2/s]

t0=273.15; % temperatura 0 C w skali Kelvina

```

n=0.8; % współczynnik CDC
mip=18*(10^(-6)); % lepkość pary
R=8.314; % stała gazowa
Mw=18.01; % masa molowa wody
lambp=0.0325*10^(-3); % przewodność cieplna pary [kW/m*K]
Pat=101.325; % ciśnienie atmosferyczne [kPa]

% GEOMETRIA CZASTKI
x1=0.002; % wymiar x cząstki [m]
x2=0.004; % wymiar y cząstki [m]
x3=0.001; % wymiar z cząstki [m]
dp=((6*x1*x2*x3)/pi)^0.33333; % równoważna średnica kuli [m]
ff=(pi*dp^2)/(2*(x1*x2+x1*x3+x2*x3)); % sferyczność [-]

% WLOTOWA PREDKOSC PARY I CZASTKI
up1=WA1/(S*rop(tg0)); % prędkość pary [m/s]
um1=100*(Ws/(S*rocZ(X1))) ; % prędkość materiału [m/s]

% WARUNEK POZATKOWY
x=[tm0; tg0; WA1; X1; um1; up1;];

% ZAKRES
z0=0;
zk=L;

% WYWOLANIE SOLVERA
[zz,xx]=ode23s('uk_row',[z0 zk],x);

%WYKRESY
figure(1);
plot(zz,xx(:,1),'b',zz,xx(:,2),'r--','LineWidth',2)
title('Wykres zmian temperatury cząstki i pary');
xlabel('długość suszarki [m]');
ylabel('temperatura [C]');
legend('temp. cząstki',' temperatura pary',4)
grid on;

figure(2)
plot(zz,xx(:,4),'g','LineWidth',2)
title('Wykres zmian wilgotności materiału');
xlabel('długość suszarki [m]');
ylabel('wilgotność [kg/kg]');
grid on;

```



```

figure(3);
plot(zz,xx(:,3),'m','LineWidth',2)
title('Wykres zmian natężenia przepływu pary');
xlabel('długość suszarki [m]');
ylabel('natężenie przepływu pary [kg/s]');
grid on

figure(4);
plot(zz,xx(:,5),'b',zz,xx(:,6),'r--','LineWidth',2)
title('Wykres zmian prędkości cząstki i pary');
xlabel('długość suszarki [m]');
ylabel('prędkość [m/s]');
legend('prędkość materiału','prędkość pary',4)
grid on;

[W,K]=size(zz);
[M,N]=size(xx);
disp('-----')
disp('Temperatura materiału na końcu suszarki [C]'),
disp(xx(M,1))
disp('Temperatura pary na początku suszarki [C]'), disp(tg0)
disp('Temperatura pary na końcu suszarki [C]'), disp(xx(M,2))
disp('Przyrost masy pary [kg/s]'), disp(xx(M,3)-WA1)
disp('Prędkość przepływu pary na końcu suszarki [m/s]'),
disp(xx(M,6))
disp('Prędkość cząstki na końcu suszarki [m/s]'), disp(xx(M,5))

if (X2 < xx(M,4));
    disp('!!!! -- Suszarka o danych parametrach nie spełnia wy-
magań --!!!!')
    disp('Wilgotność jaką osiąga materiał suszony [kg/kg]'),
disp(xx(M,4))
    disp('Wilgotność wymagana materiału wysuszonego [kg/kg]'),
disp(X2)
else (X2 >= xx(M,4));
    disp('OK')
    disp('Wilgotność jaką osiąga materiał suszony [kg/kg]'),
disp(xx(M,4))
    disp('Wilgotność wymagana materiału wysuszonego [kg/kg]'),
disp(X2)
end
disp('Długość suszarki [m]'), disp(L)
disp('-----')

```

uk_row.m jest plikiem, w którym zdefiniowano układ równań różniczkowych

 **uk_row.m**

```
function y=uk_row(z,x)

global S Ws cs cAL Xcr Xeq n ts dp g Pat Mw R t0

tm=x(1); tg=x(2); WA=x(3); X=x(4); um=x(5); ug=x(6);

alfv=alfaV(x);
epsi=epsilon(X,um);
qq=alfv*(tg-tm);
wDr=wD(x); cAg=cAp(tg); cAm=cAp(tm);
ropa=ro_p(tg); rom=rocz(X);

M1=S/Ws/(cs+cAL*X)*(qq-wDr*((cAm-cAL)*tm+HV(0)));
M2=-S/(WA* cAg)*(qq-wDr*(cAm*tm-cAg*tg));
M3=wDr*S;
M4=-wDr*S/Ws;
M5=(3/(4*rom*um))*cDD(tg,um,ug)*ropa*(((ug-um)^2)/dp)*(epsi^(-
2.67))-(g/um)*(rom-ropa)/rom;
MM=ug*ropa/um/rom*(1-epsi)*(um*drodX(x)*M4+rom*M5);

M6=1/(ropa*epsi)*(epsi*ug*M2*(Pat*Mw/R/(tg+t0)^2)+wDr-MM);

y=[M1; M2; M3; M4; M5; M6];
```

wD.m jest plikiem, w którym zdefiniowano szybkość suszenia materiału

 **wD.m**

```
function y3=wD(x)
global ts
%wD
if (x(1) < ts);
    y3 = -wD2(x);
else y3 = wD2(x);
end
```

Jeśli aktualna zawartość wilgoci jest większa niż X_{cr} , oblicza się ją dla pierwszego okresu suszenia jako (plik **wD1.m**)

wD1.m

```
function y1=wD1(x)
global ts
%wD1
tg=x(2); tm=x(1);
if (tg > ts);
    if (tm < ts) ;
        y1 = alfaV(x)*((ts-tm)/HV(tm)) ;
    else
        y1 = alfaV(x)*((tg-ts)/HV(tm)) ;
    end
else y1=0;
end
```

lub szybkość suszenia dla drugiego okresu suszenia (plik **wD2.m**)

wD2.m

```
function y2=wD2(x)
global Xcr Xeq n
X=x(4);
if (x(4) > Xcr);
    y2 = wD1(x);
else
    y2=wD1(x)*(((X-Xeq)/(Xcr-Xeq))^n);
end
```

Pozostałe funkcje z własnościami materiału i pary, oraz z obliczeniem współczynników wnikania ciepła, znajdują się w tym samym katalogu.

Wyniki 7.5

```
Temperatura materiału na końcu suszarki [C]
276.0034
Temperatura pary na początku suszarki [C]
500
Temperatura pary na końcu suszarki [C]
290.4390
Przyrost masy pary [kg/s]
0.3150
Prędkość przepływu pary na końcu suszarki [m/s]
25.0080
```

Prędkość cząstki na końcu suszarki [m/s]

16.9089

OK

Wilgotność jaką osiąga materiał suszony [kg/kg]

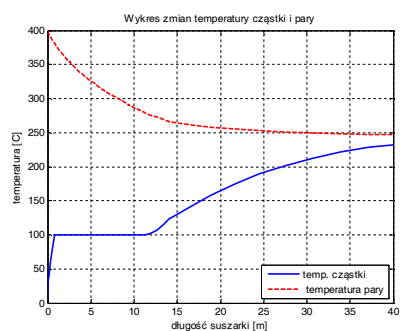
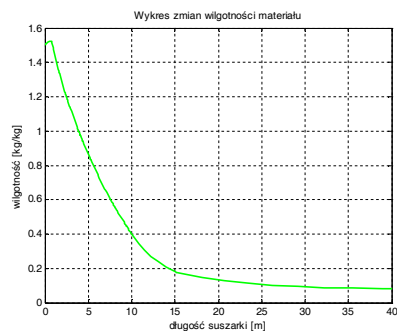
0.0824

Wilgotność wymagana materiału wysuszonego [kg/kg]

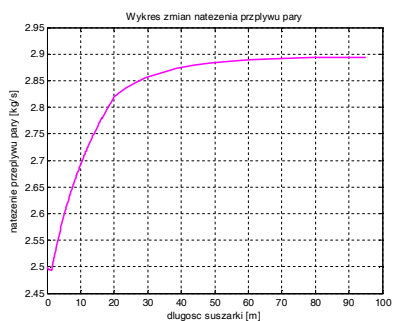
0.0929

Długość suszarki [m]

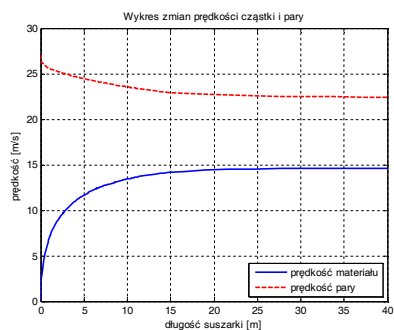
40



b)



c)



d)

Rys. 7.30. Wyniki symulacji suszarki pneumatycznej – przebiegi podstawowych parametrów w funkcji długości aparatu [m]: a) wilgotność cząstek [kg/kg], b) temperatura [°C] zrzędków (linia ciągła) i gazów (linia przerywana), c) natężenie przepływu pary (kg/s), d) prędkość liniowa (m/s) fazy stałej (linia ciągła) i parowej (linia przerywana)

PROJEKT 7.6

Temat:	Reakcja drugiego rzędu w kulistym ziarnie katalizatora
Toolbox:	-

Reagent dyfunduje od powierzchni kuli do jej wnętrza, po drodze ulegając reakcji chemicznej drugiego rzędu (Walas, 1991, s. 310). W stanie ustalonym proces ten jest opisany równaniem

$$D_{eff} \left(\frac{d^2 c_A}{dr^2} + \frac{2}{r} \frac{dc_A}{dr} \right) - k c_A^2 = 0, \quad (7.64)$$

gdzie D_{eff} dyfuzyjność efektywna [m^2/s], k – stała szybkości reakcji chemicznej [$m^3/(kmol \cdot s)$]. Warunki brzegowe są następujące:

$$\begin{aligned} \text{dla } r=0 \quad c_A &= c_{A0} \quad i \quad \frac{dc_A}{dr} = 0 \\ \text{dla } r=R \quad c_A &= c_{As} \end{aligned} \quad (7.65)$$

Wartość c_{A0} nie jest znana, lecz będzie niezbędna do rozpoczęcia całkowania. Równanie w postaci bezwymiarowej przyjmuje formę

$$\frac{d^2 y}{d\rho^2} + \frac{2}{\rho} \frac{dy}{d\rho} = \Phi^2 y^2, \quad (7.66)$$

gdzie

$$y = \frac{c_A}{c_{As}}, \quad \rho = \frac{r}{R} \quad \text{oraz} \quad \Phi = R \sqrt{k \frac{c_{As}}{D_{eff}}}. \quad (7.67)$$

A warunki brzegowe przyjmują postać:

$$\begin{aligned} \text{dla } \rho=0 \quad y &= y_0 \quad i \quad \frac{dy}{d\rho} = 0 \\ \text{dla } \rho=1 \quad y &= 1 \end{aligned} \quad (7.68)$$

W centrum kuli, gdy $\rho = 0$, pojawia się problem dzielenia przez zero w drugim członie równania 7.47. Stosując regułę de l'Hospitala (granicą tego wyrażenia jest granica ilorazu pochodnej licznika i mianownika) wyraz ten można zamienić na

$$\frac{2}{\rho} \frac{dy}{d\rho} = 2 \frac{d^2 y}{d\rho^2}. \quad (7.69)$$

Równanie 7.47 w centrum kuli przyjmie więc postać

$$3 \frac{d^2 y}{d\rho^2} = \Phi^2 y^2. \quad (7.70)$$

Równania 7.65 i 7.69 sprowadzimy do układu równań I rzędu przez podstawienie nowej zmiennej za pierwszą pochodną. I tak np. równanie 7.69 przyjmuje postać

$$\begin{aligned} \frac{dy}{d\rho} &= z \\ \frac{dz}{d\rho} &= \frac{1}{3} \Phi^2 y^2. \end{aligned} \quad (7.71)$$

Rozwiązywanie równań rzędu drugiego często prowadzi do dwupunktowego zagadnienia brzegowego (zob. rozdz. 7.4). Tak jest i tutaj. Określone ściśle mamy tylko warunek na pochodną w centrum kuli i na stężenie na powierzchni kuli. Ponieważ do rozwiązania są niezbędne dwa warunki na jednym z końców przedziału dlatego chcąc rozpocząć całkowanie od $\rho = 0$, trzeba znać y_{A0} . Znajdziemy je przez rozwiązanie równania algebraicznego

$$y(y_0) - 1 = 0, \quad (7.72)$$

gdzie funkcja $y(y_0)$ będzie stanowił rozwiązanie równania 7.47 na powierzchni kuli, w którym y_0 będzie brakującym warunkiem brzegowym w centrum. Podejście takie jest analogiem klasycznej metody strzałów (ang. *shooting method*).

Po znalezieniu tej wartości będzie można scałkować równania 7.70, aby uzyskać ostatecznie profil stężenia w kuli.

P7_6.m

```
function reakcja
% reakcja katalityczna w kuli
clear all; clc
global fi
fi=2;
% znalezienie y w centrum kuli
yc=fzero(@csurf,0.4);
disp('Brakujące y0:'); disp(yc);
```

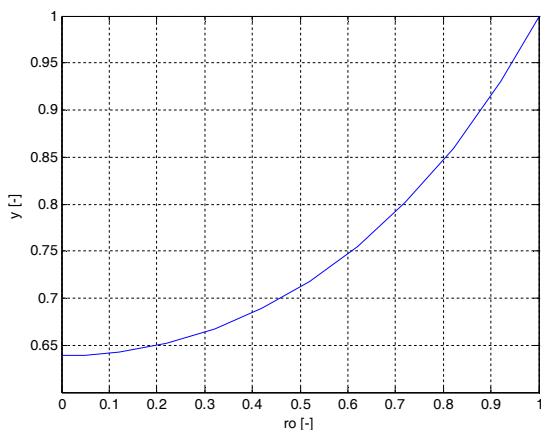
```

% profil stężenia
y0=[yc; 0];
[ro,y]=ode23(@row,[0 1], y0);
% wizualizacja wyników
figure(1)
plot(ro,y(:,1))
xlabel('ro [-]'); ylabel('y [-]')
grid on
function y = csurf(yi0) %-----
% funkcja określa warunek y=1 na powierzchni
x0=[yi0; 0];
[t,x]=ode23(@row,[0 1],x0);
[n,~]=size(x); y=1-x(n,1);
function v = row(t,x) %-----
% funkcja w której zdefiniowany jest układ równań
global fi
if t<0.01
    v = [x(2); 2/3*fi^2*x(1).^2];
else
    v = [x(2); fi^2*x(1).^2-2/t*x(2)];
end

```

Wyniki 7.6

Brakujące y0:
0.6388



Rys. 7.31. Ustalony profil bezwymiarowego stężenia w kuli podczas drugorzędowej reakcji katalitycznej

7.3.3. Wywoływanie integratorów z opcjami

Dotychczas pokazano jedynie najprostsze wywołanie integratora. W istocie pełne wywołanie wszystkich integratorów (z wyjątkiem ode15i) ma postać:

```
[t, y, te, ye, ie]=nazwa_solvera(@fun,zakres_t,x0,opcje)
```

Występujące tu parametry mają następujące znaczenie:

t	zmienna niezależna
y	wektor zmiennych zależnych
te,ye,ie	dodatkowe zmienne wyjściowe do obsługi wydarzeń (zob. 7.3.3)
@fun	funkcja wektorowa prawych stron układu,
zakres_t	zakres zmiennej niezależnej [tpocz tkonc],
x0	wektor wartości początkowych funkcji,
opcje	zmienna zawierająca opcje tworzona poleceniem odeset
opcje=odeset('opcja1',wartość,'opcja2',wartość2,...)	
do opcji należą	
reltolbłąd względny	
abstolbłąd bezwzględny	
outputfcn funkcja wyjściowa	
events obsługa wydarzeń	
stats statystyka procesu	
i inne.	

Przykład wywołania integratora z opcjami ilustruje przykład 7.11.

Przykład 7.11. Całkowanie układu równań sztywnych z opcjami

 **p7_11.m**

```
%Przykład 7.11 Całkowanie ODE z opcjami
function ODEopts
clear all; clc
t1=cputime;
% warunek początkowy
x0=[0; 2];
opts=odeset('AbsTol',1e-5,'stats','on','Jacobian',@Jac)
[t x]=ode23s(@row,[1 2],x0,opts);
t2=cputime;
% wizualizacja wyników
```



```

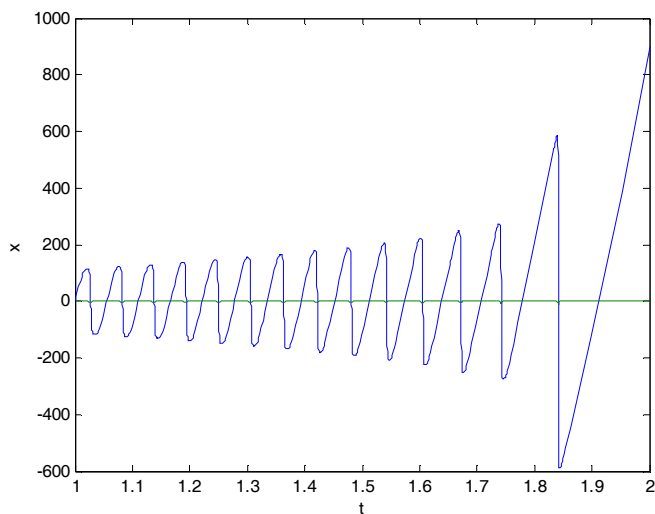
plot(t,x)
xlabel('t'); ylabel('x')
disp(t2-t1);
% funkcja definiująca pochodną
function y = row(t,x)
y = [2.*x(1)+999*x(2).^3
      1.895*x(2).*x(1)-4.*x(1)];
%funkcja definiująca Jakobian
function y = Jac(t,x)
y= [ 2.   (3*999*x(2).^2)
      (1.895*x(2)-4) (1.895*x(1))];

```

Układ równań zdefiniowany w funkcji `row` jest dość sztywny. W opcjach zmieniono zmienną `Abstol` (błąd bezwzględny) z domyślnej wartości $1e-3$ na $1e-5$ oraz polecono obliczanie Jakobianu z osobno napisanej funkcji `Jac` – inaczej `ode23s` oblicza ją numerycznie. Pozwala to na przyspieszenie obliczeń. Dodatkowo polecono wydrukować statystykę obliczeń i czas wykonania.

Rezultat 7.11

Program wydrukował następujący wykres:



Rys. 7.32. Obliczone rozwiązanie

oraz następującą statystykę

```
opts =  
  
      AbsTol: 1.0000e-005  
      BDF: []  
      Events: []  
      InitialStep: []  
      Jacobian: @Jac  
      JConstant: []  
      JPattern: []  
      Mass: []  
      MassConstant: []  
      MassSingular: []  
      MaxOrder: []  
      MaxStep: []  
      NonNegative: []  
      NormControl: []  
      OutputFcn: []  
      OutputSel: []  
      Refine: []  
      RelTol: []  
      Stats: 'on'  
      Vectorized: []  
      MStateDependence: []  
      MvPattern: []  
      InitialSlope: []  
818 successful steps  
115 failed attempts  
2686 function evaluations  
818 partial derivatives  
933 LU decompositions  
2799 solutions of linear systems  
  
1.7784
```

Wydrukowany jako pierwszy wektor `opts` zawiera stan wszystkich opcji integratora. Czas wykonania (s) jest podany na końcu. Dla porównania czas bez opcji 'Jacobian' wyniósł 1.9344 s.

Inne przykłady wywoływania integratorów z opcjami ilustrują dwie kolejne sekcje.

7.3.3.1. Rozwiązywanie zagadnienia początkowego z obsługą wydarzeń

Często nie jest wcześniej znany przedział zmiennej niezależnej (zwykle czasu), w którym całkujemy równanie ODE, natomiast jest znany warunek, którego spełnienie powinno spowodować przerwanie obliczeń. Obliczenia przerywa się gdy funkcja osiąga zadaną wartość. Dzieje się tak np. w czasie projektowania aparatu, kiedy należy określić jego długość, którą odczytamy gdy dany parametr procesowy osiągnie oczekiwaną wartość.

MATLAB umożliwia przerwanie obliczeń gdy dany warunek jest spełniony. Służą do tego opcje wywołania integratora. Przydatna forma wywołania (na przykładzie ode45) ma postać

```
[x, y, xe, ye, ie] = ode45(@dydx,xr,y0,options)
```

gdzie **xe** i **ye** to wartości zmiennych w chwili wystąpienia wydarzenia a **ie** to wskaźnik wystąpienia wydarzenia w przedziale całkowania **xr** (1 jeśli wystąpiło, 0 jeśli nie wystąpiło)

Opcje ustawia się wcześniej za pomocą funkcji **odeset** np.

```
options=odeset('Events',@events)
```

gdzie wywołanie funkcji **events** ma postać

```
[value, isterminal, dircn]=events(x,y)
```

i gdzie **value** to wartość różnicy wartości funkcji **y** i oczekiwanej wartości, **isterminal** określa czy proces ma być przzerwany (0 – nie, 1 – tak), **dircn** warunek pomocniczy, który określa znak pochodnej w tym punkcie (1 rośnie, -1 maleje). Zastosowanie tej opcji w projektowaniu ilustruje Projekt 7.6.

PROJEKT 7.7

Temat: Wyznaczenie wysokości kolumny absorpcyjnej przeciwprądowej

Toolbox: -

Zaprojektować wysokość przeciwprądowej kolumny absorpcyjnej wypełnionej do absorpcji CO₂ z powietrza w roztworze wodnym monoetanoloaminy. Bilanse różniczkowe masy CO₂ w każdej z faz mają postać (Pakowski, Głębowski, 2001, s. 279)

$$\frac{dY}{dh} = -\frac{k_G a}{w_G} (Y - Y_{eq}(X))$$

$$\frac{dX}{dh} = -\frac{k_G a}{w_L} (Y - Y_{eq}(X)).$$
(7.73)

Oznaczmy dół kolumny numerem 1, a górę kolumny 2.

Znane są stężenia wlotowe $Y_1 = 0.379$ kg/kg i $X_2 = 0.007$ kg/kg. Stężenie w fazie gazowej musi zostać obniżone do $Y_2 = 0.015$ kg/kg. Szybkość masowa przepływu fazy gazowej wynosi $w_G = 0.6$ kg/(m²s). Dobrano szybkość masową zraszania jako $w_L = 2.12 w_G$. Wyznaczyć wysokość kolumny, która zapewnia ten stopień desorpcji z fazy gazowej. Równowaga jest dana odpowiednim równaniem. Pozostałe dane są zawarte w pliku głównym projektu.

P7_7.m

%PROJEKT 7.7

%Wyznaczenie wysokości kolumny absorpcyjnej przeciwprądowej

%Zadanie projektowe

```
function kol_abs
clear all; close all; clc
global kG a wG wL Y2;
kG=2e-3; %współczynnik wnikania masy w fazie gazowej, kg/(m2 s)
a=260; %powierzchnia właściwa kontaktu faz, m2/m2
wG=0.6; %prędkość masowa fazy gazowej, kg/(m2 s)
wL=2.12*wG; %natężenie zraszania, kg/(m2 s)

% 1 - dół kolumny, 2 - góra kolumny
Y1=0.379;%stężenie skł.A w fazie gazowej na dole kolumny, kg/kg
Y2=0.015;%wymagane stęż. skł.A f fazy gazowej na wylocie, kg/kg
X2=0.007;%stężenie skł.A w fazie ciekłej na górze kolumny, kg/kg

X1=wG/wL*(Y1-Y2)+X2;
%obliczone stężenie skł.A w fazie ciekłej na dole, kg/kg

y0=[Y1; X1];
hh=[0 10];
%określenie opcji przerywania
options=odeset('Events',@events);
%wywołanie solvera
```

```

[h, y, he, ye, ie]=ode45(@dydh,hh,y0,options);
%wizualizacja wyników
plot(h,y(:,1),'b-','LineWidth',2);
hold on
plot(h,y(:,2),'r--','LineWidth',2);
xlabel('h, m'); ylabel('Y, X, kg/kg');
napis=['Znaleziona wysokość kolumny = ',num2str(he),' m'];
text(2,0.3,napis);

figure(2)
plot(y(:,2),y(:,1),'b-');
hold on
plot(y(:,2),Yeq(y(:,2)),'m--');
xlabel('X, kg/kg');
ylabel('Y, kg/kg');

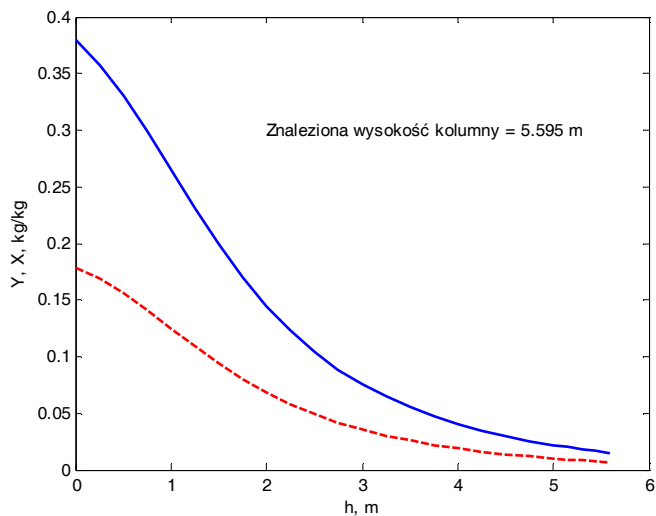
function f=dydh(h,y)
%funkcja określająca układ równań
global kG a wG wL
kGa=kG*a;
dY=y(1)-Yeq(y(2));
f=[-kGa/wG*dY;
-kGa/wL*dY];

function [value,isterminal,dircn]=events(h,y)
%funkcja obsługująca warunek przerwania obliczeń jeśli Y=Y2
%isterminal=1 oznacza przerwanie obliczeń gdy value=0
global Y2
value=y(1)-Y2;
isterminal=1;
dircn=-1;

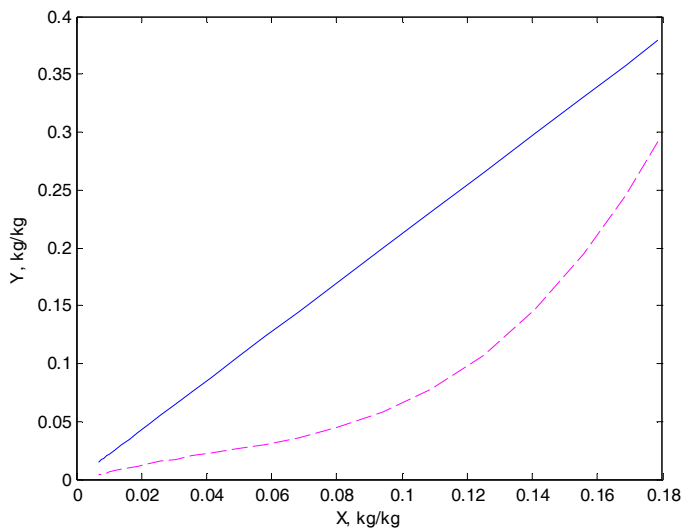
function y=Yeq(X)
%funkcja obliczająca stężenie równowagowe
d=79.607;
c=-9.795;
b=0.852;
a=-1.6e-3;
y=a+b*X+c*X.^2+d*X.^3;

```

Wyniki 7.7



Rys. 7.33. Rozkłady stężeń obu faz wzdłuż wysokości kolumny



Rys. 7.34. Linia operacyjna i linia równowagi

PROJEKT 7.8

Temat:	Selektywność odparowywania kropli dwuskładnikowej
Toolbox:	-

Kropla dwuskładnikowa złożona izopropanolu (1) i wody (2) odparowuje do strumienia powietrza (3) o stałych parametrach. Jak zmienia się skład kropli w zależności od czasu procesu i składu początkowego? Czy zawsze izopropanol odparowuje preferencyjnie jako bardziej lotny? Odpowiedź na to pytanie nie jest oczywista (*Pakowski, 1994*) – o szybkości odparowania danego składnika decyduje tu, w przeciwieństwie do destylacji, nie tylko lotność względna, ale i dyfuzyjność składnika w powietrzu. Izopropanol ma większą lotność, ale mniejszą dyfuzyjność niż woda, należy się więc spodziewać, że w pewnych warunkach będzie odparowywać preferencyjnie izopropanol, a w innych woda. Aby opisać to zjawisko, należy ułożyć odpowiednie równania bilansowe.

Bilans masy i-tego składnika ($i = 1, 2$)

$$\frac{dm_i}{d\tau} = -j_i \cdot A. \quad (7.74)$$

Bilans ciepła:

$$\sum_i m_i c_{Li} \frac{dT_L}{d\tau} = q - \sum_i j_i \Delta h_i. \quad (7.75)$$

Bilans objętości (przy założeniu, że objętość kropli oblicza się addytywnie):

$$\frac{dv}{d\tau} = -\sum_i \frac{j_i}{\rho_{Li}} \cdot A. \quad (7.76)$$

co daje wzór na pochodną promienia kropli po czasie:

$$\frac{dr_k}{d\tau} = -\sum_i \frac{j_i}{\rho_{Li}}. \quad (7.77)$$

W równaniach tych A – powierzchnia kropli [m^2], m_i – masa składnika i w kropli [kg], j_i – strumień masy składnika i [$\text{kg}/(\text{m}^2\text{s})$], Δh_i – entalpia przemiany fazowej [kJ/kg], q – strumień ciepła namacalnego [kW/m^2], r_k – promień kropli [m], ρ_{Li} – gęstość ciekłego składnika i [kg/m^3], v – objętość kropli [m^3].

Strumienie ciepła i masy należy obliczyć wg odpowiednich równań konstytutywnych (zob. plik **rate.m**).

Wykresy rozwiązywania tych równań można pokazać jako zależności od czasu bądź na płaszczyźnie fazowej. Tutaj pokazano wyniki na płaszczyźnie fazowej: stopień odparowania (stosunek aktualnej liczby moli do początkowej liczby moli w kropli) od ułamka molowego składnika 1.

Główny plik projektu zawiera solver równań ODE rozwiązujący układ równań 7.73-7.76 dla kolejnych dwudziestu składników początkowych kropli. Zawiera on obsługę wydarzeń polegającą na przerywaniu obliczeń, jeśli aktualny promień kropli będzie mniejszy niż 0.3 promienia początkowego.

P7_8.m

%PROJEKT 7.8 Selektywność odparowania kropli dwuskładnikowej

%Program rysuje krzywe selektywności odparowania kropli
%dwuskładnikowej do strumienia inerty zawierającego pary
%obu składników

```
clear all; clc; close all;
global P0 Tg ug rk0 nm0
% Dane procesowe-----
n=[1 2 3];          %numeracja składników:
                    %1-izopropanol, 2-woda, 3-powietrze
P0 = 101.325;        %ciśnienie ogólne, kPa
ug = 0.1;            %prędkość gazu, m/s
Tg = 20.0;           %temperatura gazu, C
rk0=1e-3./2;         %początkowy promień kropli, m
Tl0=20;              %początkowa temperatura kropli, C

%wilgotności względne, -
%fi1=0.0; fi2=0.0; tmax=250; % 1 zestaw
%fi1=0.0; fi2=0.1; tmax=250; % 2 zestaw
fi1=0.5; fi2=0.5; tmax=500; % 3 zestaw

propdata;
y(1)=fi1*prop('Ps',1,Tg)./P0;
y(2)=fi2*prop('Ps',2,Tg)./P0;
y(3)=1-y(1)-y(2);
%Wykresy selektywności-----
figure(1)
hold on
for J=1:21
    x(1)=0.05*(J-1);
```



```

x(2)=1-x(1);
m=mass(x,rk0,Tl0);
w0=[Tl0 m(1) m(2) rk0]';
nm0=w0(2)./mol(1)+w0(3)./mol(2);
tspan=linspace(0,tmax,200);
options=odeset('Events',@events);
[t,w,te,we,ie]=ode23s(@eqs,tspan,w0,options);
nm=w(:,2)./mol(1)+w(:,3)./mol(2);
x1=w(:,2)./mol(1)./nm;
nr=nm./nm0;
plot(x1,nr);
axis([0.0 1.0 0.0 1.0])
grid on
end

```

Układ równań bilansowych zawarty jest w pliku **eqs.m**

 **eqs.m**

```

function y=eqs(~,w)
%ODE set
global mol Tg
Tl =w(1); %droplet temperature, C
ml1=w(2); %mass of component 1, kg
ml2=w(3); %mass of component 2, kg
rk =w(4); %droplet radius, m

f1=ml1./mol(1)+ml2./mol(2); %sum n
x1=ml1./mol(1)./f1;
x2=ml2./mol(2)./f1;
if x1<0
    x1=0;
end
if x1>1
    x1=1;
end
if x2<0
    x2=0;
end
if x2>1
    x2=1;
end

a=4*pi*rk*rk; %droplet surface area

```

```

x=[x1 x2];

[code,q,j]=rate(x,Tl,rk);
if code
    y=[0 0 0 0]';
    return;
end
ro1=prop('r1',1,Tl); ro2=prop('r1',2,Tl);
cg1=prop('cg',1,Tg); cg2=prop('cg',2,Tg);
cg10=prop('cg',1,0); cg20=prop('cg',2,0);
cl1=prop('cl',1,Tl); cl2=prop('cl',2,Tl);
cl10=prop('cl',1,0); cl20=prop('cl',2,0);
hv10=prop('hv',1,0); hv20=prop('hv',2,0);

hg1=hv10+Tg*(cg1+cg10)./2;
hl1=Tl*(cl1+cl10)./2;
hg2=hv20+Tg*(cg2+cg20)./2;
hl2=Tl*(cl2+cl20)./2;
f1=j(1)*(hg1-hl1);
f2=j(2)*(hg2-hl2);
y(1)=a*(q-f1-f2)./(m11*c11+m12*c12); %dTl/dt
y(2)=-j(1)*a; %dm1/dt
y(3)=-j(2)*a; %dm2/dt
y(4)=-(j(1)./ro1+j(2)./ro2); %drk/dt
y=y';

```

Funkcja **rate**, która oblicza strumień ciepła oraz strumienie masy na powierzchni kropli zgodnie z zasadami wieloskładnikowego ruchu masy zawarta jest w pliku **rate.m**.

rate.m

```

function [code,q,j]=rate(x,Tl,rk)
%heat and mass transfer rates
global y Tg D0 nd T0 mol P0 Tc ug
R=8.314;
code=0; q=0; j=[0 0];
% break condition-----
Tb1=prop('tb',1,P0);
Tb2=prop('tb',2,P0);
if Tl>=Tb1
    Tl=Tb1-0.01;
end
if Tl>=Tb2

```

```

        Tl=Tb2-0.01;
    end
    Tm=Tl+0.3333*(Tg-Tl); %mean boundary layer temperature
    Tr=(Tm+T0)/T0;
    code=not((abs(x(1)+x(2)-1)<1e-9)&(rk>0));
    if code
        return;
    end

    i=[1 2 3];
% diffusivities-----
    d12=D0(1,2).*Tr.^nd(1,2);
    d13=D0(1,3).*Tr.^nd(1,3);
    d23=D0(2,3).*Tr.^nd(2,3);
    dm(1)=1./((y(2)./d12+y(3)./d13)./(1-y(1)));
    dm(2)=1./((y(1)./d12+y(3)./d23)./(1-y(2)));
% transfer coefficients-----
    mmol=sum(y.*mol);
    C=P0./(R*(Tm+T0)); %{kmol/m3}
    cg=prop('cg',i,Tm);
    ch=sum(cg'.*y.*mol)./mmol;
    %gas mixture viscosity
    a=y.*sqrt(mol.*(Tc+T0));
    mi=prop('mi',i,Tm);
    mim=sum(mi'.*a)./sum(a);
    %gas mixture heat conductivity
    b=y.*mol.^0.333;
    lb=prop('lb',i,Tm);
    lbm=sum(lb'.*b)./sum(b);
    rog=mmol*C;          nim=mim./rog;
    Re=2*ug*rk./nim;      Pr=ch*mim./lbm;
    Sc=nim./dm;
    if ug>0
        Re=0.552*sqrt(Re);          % Froessling equation
        Nu=2+Re*Pr.^0.333;
        Sh=2+Re*Sc.^0.333;
    else
        Nu=2;      Sh=[2 2];
    end
% heat flux-----
    alfa =(Nu./2)*lbm/rk;      % {kW/m2K}
    q=alfa*(Tg-Tl);           % {kW/m2K}
% mass fluxes -----

```

```

yr =equil(x,T1);
if y(3)~=1                                %{ Wilke, Hougen formula}
    dm3=1./((y(1)./d13+y(2)./d23)./(1-y(3)));
    cd=(1-y(1)-y(2))./(1-yr(1)-yr(2));
    code=not(cd>0);
    if code
        return;
    end
    cd=log(cd);          f=C.*dm3.*cd./rk;
    beta=Sh./2.*f;      cz=exp(cd.*dm3./dm);
    code=not(cz(1)~=1)&(cz(2)~=1);
    if code
        return;
    end
    j=mol(1:2).*beta.*(yr.*cz-y(1:2))./(cz-1); %kg/m2s
    else
    % in dry gas the above formula is non-applicable
    cz01=ch.*mmol; cz02=lbm./(cz01.*C);
    Le=(cz02./dm).^(-0.667);
    beta=alfa.*Le./cz01;
    cz1=(1-y(1))./(1-yr(1));
    cz2=(1-y(2))./(1-yr(2));
    lncz=[log(cz1) log(cz2)];
    j=mol(1:2).*beta.*lncz;
    code=0;
end

%Ackermann correction to heat flux
ack=(j(1)*cg(1)+j(2)*cg(2))./alfa;
if (abs(ack)<20)&&(abs(ack)>1e-5)
    Ac=ack./(exp(ack)-1);
else
    Ac=1;
end
q=Ac.*q;

```

Warunek przerywania obliczeń solvera ode23s jest zawarty w pliku **events.m**. Obliczenia przerywamy, gdy promień kropki osiągnie 0.3 wartości początkowej, co oznacza 97.3% odparowanej objętości początkowej. Dalsze prowadzenie obliczeń jest możliwe, ale wymaga lepszej kontroli kroku całkowania. Do celów badania selektywności nie jest to potrzebne.

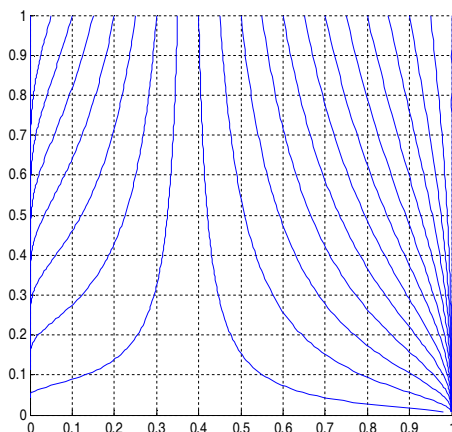
events.m

```
function [value, isterminal, dirc]=events(~,w)
global rk0
value=w(4)-0.3*rk0;
isterminal=1;
dirc=-1;
```

Pozostałe pliki zawierają bazę danych właściwości składników układu oraz metodę NRTL obliczania równowagi ciecz-para (VLE) oraz funkcję obliczającą masy składników w kropli na podstawie jej promienia i ułamków molowych.

Wyniki 7.8

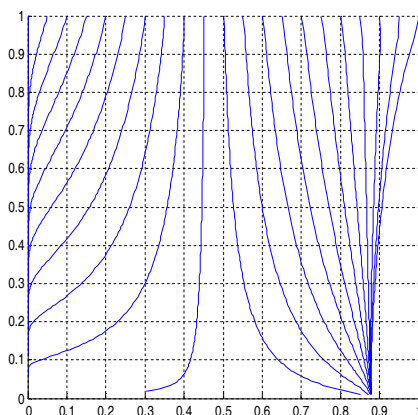
Przy odparowywaniu do strumienia suchego powietrza (Rys. 7.35) można stwierdzić, że z punktu widzenia dynamiki procesu istnieją tu trzy węzły, przy czym dwa są stabilne (stany stacjonarne), a jeden niestabilny. Stany stacjonarne odpowiadają składom czystych składników, a węzeł niestabilny $x_1 = 0.37$ dzieli obszar wszystkich składów początkowych na strefy wpływu obu tych stanów stacjonarnych. Można więc powiedzieć, że z kropli zawierającej mniej niż 0.37 ułamka molowego izopropanolu istotnie będzie odparowywał preferencyjnie izopropanol, jednakże z kropli bogatszej w izopropanol odparowywać będzie preferencyjnie woda, a więc składnik mniej lotny.



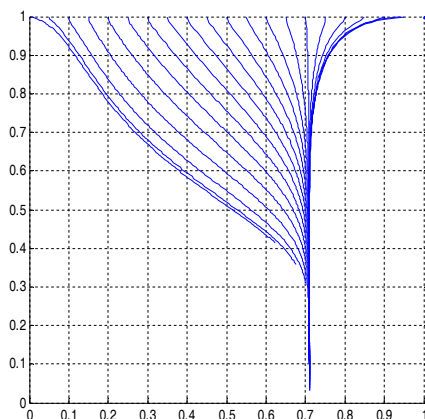
Rys. 7.35. Kropla odparowuje do suchego powietrza
Węzły stabilne są w punktach $x_1 = 0$ i $x_1 = 1$ a niestabilny w punkcie $x_1 = \text{ok. } 0.37$

Wzbogacenie powietrza w parę jednego ze składników powoduje przesunięcie jednego ze stanów stacjonarnych w obszar dwuskładnikowy jak na Rys. 7.36. Wzbogacenie o parę obu składników może spowodować powstanie jednego wspólnego stanu stacjonarnego jak na Rys. 7.37.

Zawartość par składnika w powietrzu wyraża wilgotność względna $\varphi_i = p_i/p_{si}$ (p – prężność cząstkowa [Pa], p_s – prężność pary nasyconej [Pa])



Rys. 7.36. Kropla odparowuje do powietrza o $\varphi_1 = 0$ i $\varphi_2 = 0.1$
Węzły stabilne są w punktach $x_1 = 0$ i $x_1 = 0.88$ a niestabilny w punkcie $x_1 = \text{ok. } 0.47$



Rys. 7.37. Kropla odparowuje do powietrza o $\varphi_1 = 0.5$ i $\varphi_2 = 0.5$
Jest tylko jeden węzeł stabilny w punkcie $x_1 = 0.71$

7.3.3.2. Rozwiązywanie zagadnienia początkowego z przekazywaniem dodatkowych parametrów

Wiele funkcji służących do obliczeń numerycznych w tym solvery równań mają ściśle określoną listę parametrów przekazywanych przez nagłówek. Jeśli chcemy przekazać do nich dodatkowe parametry pozostaje zadeklarowanie ich jako zmiennych globalnych. Ten sposób nie jest jednak zalecany, gdyż nazw zmiennych globalnych nie można już wykorzystywać lokalnie.

Istnieją jeszcze inne sposoby, można polecić wykorzystanie funkcji anonimowych (zob. Rozdz. 2.8.6). Sposób ten ilustruje Projekt 7.9. Dotyczy on przypadku reaktora rurowego pracującego w stanie ustalonym w pętli recyrkulacji. Model reaktora wymaga czterech parametrów, które zostaną przekazane do integratora równań ODE poprzez funkcję anonimową.

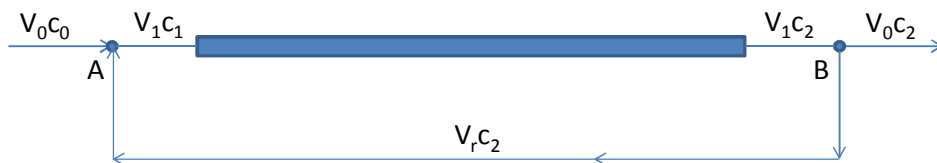
PROJEKT 7.9

Temat: Stan ustalony pracy reaktora w pętli recyrkulacji

Toolbox: -

W reaktorze rurowym zachodzi reakcja I rzędu zużywająca reagent dopływający do układu. Ze względu na silną egzotermiczność reakcji, strumień zasilający jest częściowo rozcieńczany strumieniem przereagowanym zawracanym do obiegu ze stopniem recyrkulacji r . Należy obliczyć stężenia na wlocie i wylocie z reaktora w stanie ustalonym, jeśli natężenie objętościowe na wlocie do układu wynosi V_0 [m^3/s], a stężenie wlotowe c_0 [kmol/m^3]. Stała szybkości reakcji wynosi k [$1/\text{s}$], a pole przekroju reaktora wynosi S [m^2].

Schemat reaktora przedstawia poniższy rysunek.



Rys. 7.38. Schemat strumieni reaktora rurowego

Napiszmy równania bilansowe dla węzła A

$$\begin{aligned} V_0 + V_r &= V_1 \\ V_0 c_0 + V_1 r c_2 &= V_1 c_1, \end{aligned} \quad (7.78)$$

gdzie $V_r = V_1 r$.

Stopień recyrkulacji r jest zdefiniowany jako stosunek natężenia strumienia zwracanego do strumienia skąd został pobrany. Na podstawie tych równań otrzymujemy

$$(1-r)c_0 + r c_2 - c_1 = 0. \quad (7.79)$$

Ponieważ c_2 nie jest znane, zostanie ono obliczone z modelu reaktora na podstawie stężenia wlotowego c_1 , które posłuży jako warunek początkowy w prostym modelu reaktora postaci

$$\frac{dc}{dl} = -\frac{k}{V} S c. \quad (7.80)$$

Stężenie wlotowe do reaktora znajdziemy, rozwiązując równanie 7.78 za pomocą solvera równań algebraicznych **fzero**. Ma on tylko dwa argumenty wejściowe: punkt startowy i funkcję celu – tutaj jest to lewa strona równania 7.78. Będzie nią tu funkcja anonimowa f .

P7_9.m

```
%PROJEKT 7.9
%Reaktor w pętli recyklu
clear all; clc;
global c2
%dane
L=10;           %długość reaktora
S=0.1 ;         %pole przekroju reaktora [m2]
k=0.01;         %stała reakcji I rzędu [1/s]
r=0.3;          %stopień recyklu [-]
c0=2.5;         %stężenie na wlocie [kmol/m3]
V0=0.01;        %natężenie objętościowe dopływu [m3/s]
c1guess=1.;
f=@(x)paramfun(x,c0,L,S,k,r,V0); %funkcja anonimowa
[x,fval]=fzero(f,c1guess);
c1=x;
disp('Obliczone stężenie na wlocie do reaktora'); disp(c1);
disp('Obliczone stężenie na wylocie z reaktora'); disp(c2);
delta=c1-(1-r)*c0-c2*r;
```



```
disp('Sprawdzenie bilansu: L-P =');disp(delta);
```

Funkcja obliczająca lewą stronę równania 7.78 na podstawie równania 7.79 scałkowanego w przedziale $\langle 0, L \rangle$ ma postać.

paramfun.m

```
function w = paramfun(c1,c0,L,u,k,r,V0)
    %funkcja oblicza bilans składnika dla węzła A
global c2
[~,c]=ode23(@RHS,[0 L],c1);
[n,~]=size(c);
c2=c(n,1);
w=c1-(1-r)*c0-r*c2;
    function v=RHS(~,c)
        u=V0/(1-r);
        v=-k*c(1)/u;
    end
end
```

Jak widać, do tej funkcji przekazujemy przez nagłówek wszystkie parametry reaktora bez potrzeby użycia zmiennych globalnych. Jedyną zmienną globalną, której trzeba było użyć, jest c_2 – stężenie wylotowe z reaktora.

Wyniki 7.9

Obliczone stężenie na wlocie do reaktora

1.7505

Obliczone stężenie na wylocie z reaktora

0.0016

Sprawdzenie bilansu: L-P =

5.9197e-017

7.4. Rozwiązywanie dwupunktowych zagadnień brzegowych

Częstym przypadkiem, który zdarza się przy określaniu warunków brzegowych dla równania zwyczajnego drugiego rzędu lub układu równań pierwszego rzędu, jest sytuacja, gdy warunki te są określone na przeciwległych krańcach obszaru całkowania. Stan taki występuje np. w przypadku symulacji przeciwpądowego

kontaktu dwu faz lub w przypadku wymiany ciepła lub masy w przepływie z dyspersją wzdłużną. Zamiast klasycznej metody strzałów MATLAB oferuje m.in. wyspecjalizowane funkcje `bvp4c` i `bvp5c` o następującej składni

```
sol = bvp4c(odefun,bcfun,solinit,options)
```

gdzie:

`odefun` funkcja zwracająca wektor prawych stron układu równań

`bcfun` funkcja zwracająca wektor residuów warunków brzegowych na obu krańcach przedziału (wpisuje się je tylko dla znanych warunków jako różnicę oczekiwanej i znanej wartości funkcji)

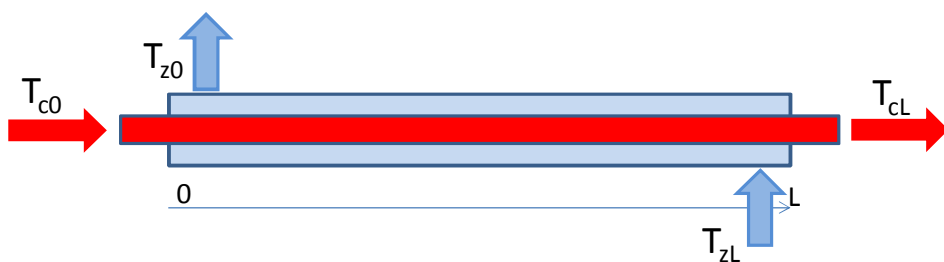
`solinit = bvpinit(x, yinit, params)` – macierz zawierająca: w pierwszym wierszu kolejne punkty z domeny funkcji, w następnych – odpowiadające im założenia oczekiwanej wartości każdej z funkcji

`options, params` – opcje dodatkowe (zob. pomoc programu).

Działanie funkcji `bvp4c` ilustruje przykład symulacji przeciwprądowego wymiennika ciepła typu rura w rurze.

Przykład 7.12. Przeciwproudowy wymiennik ciepła – dwupunktowe zagadnienie brzegowe

Dane są wymiary wymiennika i strumienie masy obu czynników. Czynnik ciepły wpływa na początku wymiennika, i tu jest określona jego temperatura, a czynnik zimny wpływa na drugim końcu wymiennika, gdzie także określona jest jego temperatura wlotowa. Pokazano to na Rys. 7.39.



Rys. 7.39. Schemat przeciwprądowego wymiennika ciepła

Bilanse ciepła dla elementu różniczkowego objętości wymiennika o długości dl mają postać:

$$W_c c_c T_c - W_c c_c \left(T_c + \frac{dT_c}{dl} dl \right) - \pi DK (T_c - T_z) = 0 \quad (7.81)$$

$$W_z c_z \left(T_z + \frac{dT_z}{dl} dl \right) - W_z c_z T_z + \pi DK (T_c - T_z) = 0, \quad (7.82)$$

co po przekształceniu daje

$$\frac{dT_c}{dl} = -\pi \frac{DK}{W_c c_c} (T_c - T_z) \quad (7.83)$$

$$\frac{dT_z}{dl} = -\pi \frac{DK}{W_z c_z} (T_c - T_z). \quad (7.84)$$

Ten układ równań ma następujące warunki brzegowe:

$$\begin{aligned} \text{dla } l = 0 \quad T_c &= T_{c0} \\ \text{dla } l = L \quad T_z &= T_{zL}. \end{aligned} \quad (7.85)$$

Zagadnienie rozwiązano w następujący sposób:

p7_12.m

```
%Przykład 7.12 Dwupunktowe zagadnienie brzegowe
function bvp_wymiennik
% wymiennik typu rura w rurze
% zadanie symulacyjne - dwupunktowe zagadnienie brzegowe
clear all; close all; clc
global K D Wc Wz cc cz Tc0 TzL
K=2.2;      %współczynnik przenikania ciepła, kW/(m2 K)
D=0.2;      %średnica rury wewnętrznej
Wc=1;       %natężenie masowe przepływu cieczy cieplej, kg/s
```

```

Wz=1.5;      %natężenie masowe przepływu cieczy zimnej, kg/s
cc=4.19;     %ciepło właściwe cieczy ciepłej, kJ/(kg K)
cz=1.1;     %ciepło właściwe cieczy zimnej, kJ/(kg K)
Tc0=90;     %temperatura wlotowa cieczy ciepłej, kJ/(kg K)
TzL=20;     %temperatura wlotowa cieczy zimnej, kJ/(kg K)
L=4;        %długość wymiennika, m

%założenie rozwiązania
guess=[60 40];
insol = bvpinit(linspace(0,L,10),guess);

%wywołanie solvera
sol=bvp4c(@rownODE,@bcs,insol);

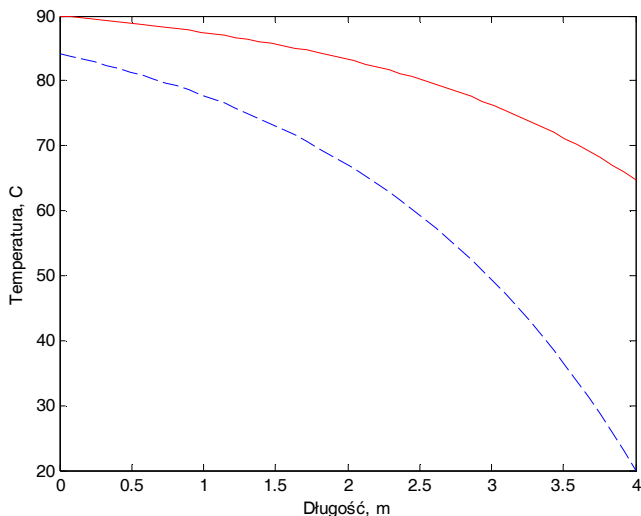
%wykres obliczonych profili temperatur
x=linspace(0,L,50);
T=deval(sol,x)';
plot(x,T(:,1),'r-');
hold on
plot(x,T(:,2),'b--');
xlabel('Długość, m');
ylabel('Temperatura, C');

function res = bcs(ya,yb)
%funkcja określa residua na krańcach przedziału
global Tc0 TzL
res = [ ya(1)-Tc0;
        yb(2)-TzL];

function dydx=rownODE(x,y)
%funkcja zwraca prawe strony układu równań
global K D Wc Wz cc cz
fc=pi*D*K/Wc/cc;
fz=pi*D*K/Wz/cz;
dydx=[ -fc*(y(1)-y(2));
        -fz*(y(1)-y(2))];

```

Rezultat 7.12



Rys. 7.40. Rozwiązanie dwupunktowego zagadnienia brzegowego symulacji wymiennika ciepła typu rura w rurze

7.5. Rozwiązanie uwikłanych równań różniczkowych

Uwikłane zwyczajne równanie różniczkowe to takie, które nie da się przekształcić do postaci (7.1) lub (7.2). Stosunkowo rzadkie przypadki tego typu równań rozwiązuje solver `ode15i`, co ilustruje poniższy przykład dostępny w pomocy MATLABa.

Uwikłane równanie Weissingera ma postać

$$ty^2 \left(\frac{dy}{dt} \right)^3 - y^3 \left(\frac{dy}{t} \right)^2 + t(t^2 + 1) \frac{dy}{dt} - t^2 y = 0 \quad (7.86)$$

z warunkiem początkowym

$$\text{dla } t_0 = 1 \quad y(t_0) = \sqrt{3/2} \quad (7.87)$$

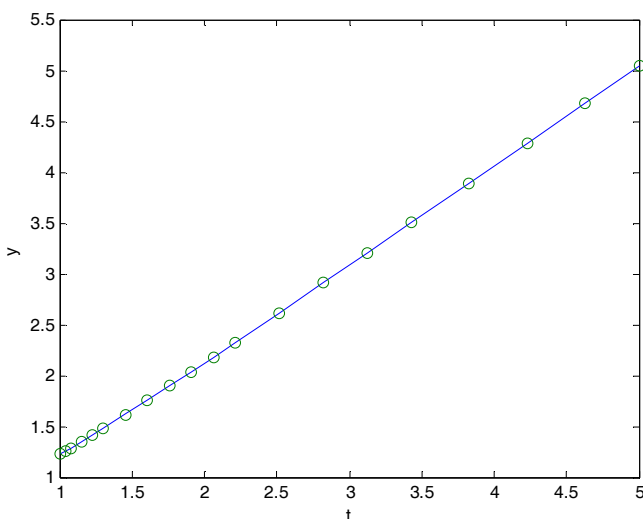
MATLAB ma wbudowaną funkcję `@weissinger`, która oblicza lewą stronę równania (7.37). Potrzebny warunek brzegowy dla $y'(t_0)$ można oszacować za pomocą funkcji `decic`. Jest to specjalna funkcja przypisana integratorowi `ode15i`, służąca obliczaniu spójnych warunków brzegowych dla równania uwiakłanego (zob. więcej w pomocy MATLABa). Program i graficzna forma rozwiązania pokazane są poniżej. Wyrażenie na zmienną `ytrue` to rozwiązanie analityczne.

Przykład 7.13.

 **p7_13.m**

```
%Przykład 7.13
%równanie Weissingera
t0 = 1;
y0 = sqrt(3/2);
yp0 = 0;
[y0,yp0] = decic(@weissinger,t0,y0,1,yp0,0);
[t,y] = ode15i(@weissinger,[1 5],y0,yp0);
ytrue = sqrt(t.^2 + 0.5);
plot(t,y,t,ytrue,'o');
xlabel('t'); ylabel('y');
```

Rezultat 7.13



Rys. 7.41. Rozwiązanie równania Weissingera uzyskane za pomocą solvera `ode15i`.
o – rozwiązanie numeryczne, – – rozwiązanie analityczne

LITERATURA

- Finlayson B., „Nonlinear Analysis in Chemical Engineering”, McGraw-Hill 1980.
- Krzystek L., „Stechiometria i kinetyka bioprocessów”, Wyd. PŁ, Łódź 2010.
- Lorenz E.N., 1963, *Deterministic nonperiodic flow*, J. Atmos. Sci, 20, 130-141.
- Moler C., *Numerical Computations with MATLAB*, SIAM, 2004, rozdz. 7, s. 19.
- Orlik M., „Reakcje oscylacyjne porządek i chaos” WNT, Warszawa 1996.
- Ott E., „Chaos w układach dynamicznych”, WNT, Warszawa 1997, s. 24.
- Pakowski Z., „Stany stacjonarne w procesie odparowywania kropli wieloskładnikowych do strumienia inerty”, Wyd. PŁ, 1994.
- Pakowski Z. Głębowski M., „Symulacja procesów inżynierii chemicznej”, Wyd. PŁ, Łódź 2001.
- Pakowski Z. Druzdel A. Drwiega J., 2002. *Validation of an expanding superheated steam flash dryer for cut tobacco based on processing data*. IDS 2002, Beijing, Vol. B, 828-836.
- Pakowski Z., „Projektowanie suszarek do suszenia parą przegrzaną”, Wyd. PŁ, Łódź 2012.
- Ramirez W.F., *Computational Methods for Process Simulation*, 2nd ed., Butterworth-Heinemann, 1997.
- Tabiś B., „Zasady inżynierii reaktorów chemicznych”, WNT, Warszawa 2000, ss. 136-137.
- Walas S.M., „Modeling with Differential Equations in Chemical Engineering”, Butterworth-Heinemann, 1991.
- Wesselingh J.A., R. Krishna, „Mass Transfer in Multicomponent Mixtures”, VSSD, Delft, 2006.

8. ROZWIĄZYWANIE RÓWNAŃ RÓŻNICZKOWYCH CZĄSTKOWYCH

Równaniami różniczkowymi cząstkowymi (PDE) opisujemy obiekty o parametrach rozłożonych. W obszarze przenoszenia pędu, ciepła i masy równania te są rzędu drugiego, jak np. drugie równanie Fouriera czy Ficka i są typu parabolicznego, gdy chodzi o opis stanu nieustalonego, lub eliptyczne, gdy mowa o stanie ustalonym. Równania bilansowe obiektów o parametrach rozłożonych mogą być również pierwszego rzędu jak np. w przepływie krzyżowym dwu faz.

Dla najczęściej występujących równań cząstkowych typu parabolicznego (nieustalona dyfuzja i przewodzenie) można użyć łatwą do zaprogramowania metodę linii lub posłużyć się, dostępnym w MATLABie, solverem `pdepe`. Zostaną tu omówione obie te metody.

Do rozwiązywania równań typu eliptycznego jak i równań PDE rzędu pierwszego zostaną zaprezentowane metody uproszczone, nadające się do samodzielnego zaprogramowania. Równania typu hiperbolicznego bardzo rzadko pojawiają się w inżynierii procesowej i nie będą tu omawiane.

MATLAB oferuje skrzynkę narzędziową Partial Differential Equations Toolbox, gdzie znajduje się szereg narzędzi do rozwiązywania podstawowych typów równań PDE oraz do tworzenia siatek dyskretizacyjnych zarówno metodą różnic skończonych jak i metodą elementów skończonych (FEM).

8.1. Klasyfikacja równań różniczkowych cząstkowych

8.1.1. Równania rzędu pierwszego

Równania PDE rzędu pierwszego to, z reguły, równania bilansowe dla obiektów o parametrach rozłożonych w stanie ustalonym, dla przypadku co najmniej dwuwymiarowego, lub dla obiektu jednowymiarowego o parametrach rozłożonych gdy drugą zmienną niezależną jest czas. W równaniach tych nie może być jednak członów II rzędu, takich jak przewodzenie ciepła, dyfuzja czy dyspersja.

Równanie cząstkowe rzędu I ma następującą postać

$$A \frac{\partial u}{\partial x} + B \frac{\partial u}{\partial y} + C \frac{\partial u}{\partial z} + D \frac{\partial u}{\partial \tau} + Eu = F(x, y, z, \tau, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial u}{\partial z}, \frac{\partial u}{\partial \tau}) \quad (8.1)$$

gdzie x, y, z – współrzędne przestrzenne, τ – czas. Współczynniki A, B, C, D, E mogą w przypadku ogólnym być funkcjami wszystkich zmiennych, tak jak wyraz wolny F . Jeśli we współczynnikach brak pochodnych mówimy, że równanie

jest *quasiliniowe*, jeśli brak w nich również zmiennej zależnej mówimy, że jest ono *liniowe*. Równanie bez wyrazu wolnego nazywamy *homogenicznym*.

W podstawowej wersji MATLABa brak wyspecjalizowanych narzędzi do rozwiązywania numerycznego równań PDE rzędu pierwszego. Przedstawimy dalej dwie proste metody rozwiązywania tych równań, oparte o metodę Eulera.

8.1.2. Równania rzędu drugiego

Kanoniczną postać równania PDE rzędu drugiego (dla dwu zmiennych niezależnych) przedstawia wzór:

$$A \frac{\partial^2 u}{\partial x^2} + B \frac{\partial^2 u}{\partial x \partial y} + C \frac{\partial^2 u}{\partial y^2} = F\left(x, y, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right) \quad (8.2)$$

gdzie: u – to zmienna zależna (szukana funkcja), x i y – zmienne niezależne.

Równania PDE można podzielić na równania eliptyczne, paraboliczne i hiperboliczne. Rodzaj równania, charakter rozwiązania i klasa problemu zależy od wyróżnika równania danego wzorem

$$\Delta = B^2 - 4AC. \quad (8.3)$$

♦ Jeśli $\Delta < 0$, równanie jest typu eliptycznego. Typowym równaniem eliptycznym jest równanie Poissona:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = F(x, y), \quad (8.4)$$

i równanie Laplace'a, które dla przewodzenia ciepła w dwuwymiarowym ciele anizotropowym, w stanie ustalonym ma postać:

$$\lambda_x \frac{\partial^2 T}{\partial x^2} + \lambda_y \frac{\partial^2 T}{\partial y^2} = 0, \quad (8.5)$$

gdzie λ_x i λ_y to przewodność cieplna [W/(m K)], odpowiednio w kierunku x i y .

♦ Jeśli $\Delta = 0$, równanie jest typu parabolicznego. Typowym równaniem parabolicznym jest równanie przewodzenia ciepła lub dyfuzji w stanie nieustalonym:

$$\frac{\partial c_A}{\partial \tau} = D \frac{\partial^2 c_A}{\partial x^2}, \quad (8.6)$$

gdzie c_A – stężenie składnika A [kmol/m³], D – dyfuzyjność [m²/s].

♦ Kiedy $\Delta > 0$ równanie staje się hiperboliczne. Typowym równaniem hiperbolicznym jest równanie falowe:

$$\frac{\partial^2 u}{\partial \tau^2} = v^2 \frac{\partial^2 u}{\partial x^2}, \quad (8.7)$$

gdzie v jest prędkością propagacji fali. W procesach transportu pędu, ciepła i masy tego typu równania występują bardzo rzadko.

Należy dodać, że jeśli współczynniki równania 8.2 nie są stałe, możliwe są równania typu *mieszanego* np. paraboliczno-eliptyczne czy paraboliczno-hyperboliczne.

Równanie należy uzupełnić wymaganą liczbą warunków brzegowych⁷. Liczbę warunków brzegowych (BC) wyznacza suma rzędów pochodnych po wszystkich zmiennych niezależnych.

8.2. Zagadnienia początkowe i brzegowe

W zagadnieniach, z którymi mamy do czynienia w inżynierii procesowej występują cztery typy warunków brzegowych.

♦ Warunek brzegowy I rodzaju (warunek Dirchleta).

Dana jest wartość funkcji na brzegu przedziału

$$u|_{x=0} = u_0. \quad (8.8)$$

Warunek ten, w przypadku ruchu ciepła i masy, stosuje się na powierzchni międzyfazowej kontaktu faz, wtedy gdy opór wnikania w fazie zewnętrznej jest bardzo mały (liczba Biota > 100). Wtedy na granicy rozdziału faz wartość funkcji osiąga wartość równowagową np.

$$c|_{x=L} = c^*. \quad (8.9)$$

♦ Warunek brzegowy II rodzaju (warunek Neumanna).

Dana jest pochodna w kierunku normalnym na krawędzi przedziału

$$\left. \frac{\partial u}{\partial x} \right|_{x=0} = p. \quad (8.10)$$

⁷ Warunek brzegowy dotyczący czasu to warunek początkowy.

Najczęściej warunek ten stosuje się dla krawędzi obszaru, która jest izolowana lub jest osią (płaszczyzną) symetrii. Pochodna ma wtedy wartość zero np.:

$$\left. \frac{\partial T}{\partial x} \right|_{x=0} = 0 . \quad (8.11)$$

♦ Warunek brzegowy III rodzaju (warunek Robina). Pochodna na granicy obszaru jest funkcją liniową:

$$-a \left. \frac{\partial u}{\partial x} \right|_{x=L} = c + bu . \quad (8.12)$$

Zwykle warunek ten stosuje się na powierzchni międzyfazowej, gdy w fazie zewnętrznej występuje wnikanie, a w fazie rozpatrywanej przewodzenie ciepła bądź dyfuzja

$$-\lambda \left. \frac{\partial T}{\partial x} \right|_{x=L} = \alpha(T - T_{\infty}) \quad (8.13)$$

♦ Warunek brzegowy IV rodzaju. Ten specyficzny warunek stosuje się w przypadku dwu stykających się faz (np. A i B), jeśli w obu odbywa się przewodzenie ciepła bądź dyfuzja. Wtedy strumień przechodzący z fazy do fazy jest taki sam

$$-\lambda_A \left. \frac{\partial T_A}{\partial x} \right|_{x=L} = -\lambda_B \left. \frac{\partial T_B}{\partial x} \right|_{x=L} . \quad (8.14)$$

8.3. Rozwiązywanie równań cząstkowych pierwszego rzędu

8.3.1. Stan nieustalony

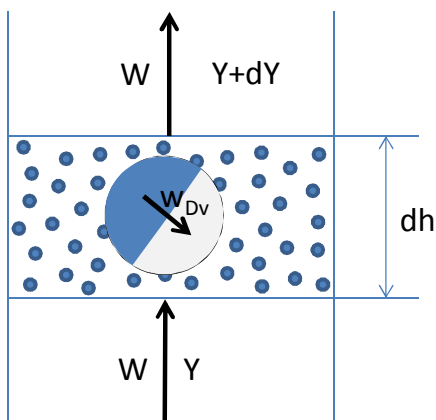
Prostym przypadkiem opisanym układem równań PDE I rzędu z pochodnymi po czasie i współrzędnej przestrzennej jest wymiana ciepła lub masy w złożu ziarnistym, z oporem wnikania wyłącznie w fazie gazowej. Metodykę rozwiązywania tego przypadku przedstawia poniższy projekt (*Pakowski i Głębowski, 2001*, s. 368).

PROJEKT 8.1

Temat: Desorpcja z głębokiego złoża przedmuchiwane gazem
 Toolbox: -

Kolumna o wysokości H jest wypełniona materiałem ziarnistym zawierającym składnik zaadsorbowany A. Jego ułamek masowy wynosi X_0 . W chwili $\tau = 0$ od dołu kolumny jest tłoczone powietrze, do którego składnik ten desorbuje. Obliczyć przebiegi czasowe ułamka masowego składnika A w fazie stałej wzdłuż wysokości złoża. Określić przybliżony czas, po jakim proces się zakończy. Pozostałe dane zawarte są w treści programu.

Rozważmy element różniczkowej objętości stacjonarnego złoża. Ponieważ przepływ gazu jest jednokierunkowy, przypadek staje się jednowymiarowy.



Rys. 8.1. Schemat różniczkowego elementu objętości złoża

Oznaczenia:

- a powierzchnia właściwa złoża [m^2/m^3]
- H całkowita wysokość złoża [m]
- h wysokość bieżąca złoża [m]
- k_Y współczynnik wnikanía masy [$\text{kg}/\text{m}^2\text{s}$]
- S powierzchnia przekroju złoża [m^2]
- W natężenie przepływu gazu inertynego [kg/s]
- X uł. masowy składnika dyfundującego w fazie stałej [kg/kg]
- Y uł. masowy składnika dyfundującego w fazie gazowej [kg/kg]
- τ czas [s]
- Y_r równowagowy uł. masowy składnika A w fazie gazowej [kg/kg]

- ε porowatość złoża [-]
- ψ sferyczność cząstki [-]
- ρ_g gęstość gazu [kg/m^3]
- ρ_m gęstość materiału [kg/m^3]

Bilans masy składnika dyfuzyjnie czynnego w różniczkowej objętości złoża

- faza gazowa

$$WY - W(Y + dY) + w_{Dv} S dh = \varepsilon \rho_g S dh \frac{dY}{d\tau} \quad (8.15)$$

- faza stała

$$- w_{Dv} S dh = (1 - \varepsilon) \rho_m S dh \frac{dX}{d\tau}, \quad (8.16)$$

gdzie

$$w_{Dv} = k_Y a (Y_r - Y) \quad (8.17)$$

jest objętościową szybkością desorpcji [$\text{kg}/(\text{m}^3\text{s})$]. Wartość równowagową ułamka masowego Y_r oblicza się na podstawie izotermy desorpcji materiału ziarnistego. W tym przypadku ma ona postać

$$Y_r = 5.5 \left[1 - \exp(-0.05 X^{0.6}) \right]. \quad (8.18)$$

Współczynnik wnikania masy można obliczyć z równania

$$Sh = 0.91 Re^{0.49} Sc^{0.33} \psi. \quad (8.19)$$

Ponieważ akumulacja w fazie gazowej jest zaniedbywane mała w stosunku do pozostałych składników równania, dlatego można ją pominąć. Ostatecznie równania przyjmują postać:

$$\begin{aligned} \frac{\partial Y}{\partial h} &= \frac{S}{W} w_{Dv} \\ \frac{\partial X}{\partial \tau} &= - \frac{w_{Dv}}{(1 - \varepsilon) \rho_m} \end{aligned} \quad (8.20)$$

z warunkami brzegowymi

$$\begin{aligned} \text{dla } \tau = 0 \quad X(h) &= X_0 \\ \text{dla } h = 0 \quad Y(\tau) &= Y_0 \end{aligned} \quad (8.21)$$

Tego układu równań różniczkowych nie da się rozwiązać przy użyciu standardowych narzędzi MATLABa. Należy użyć znanych metod numerycznych, z których najprostsza to jawna metoda Eulera. W tym celu należy podzielić wysokość złoża na N warstw o grubości ΔH . Znając początkową wartość X_0 oraz wartość Y_0 na wlocie do złoża dla każdego kroku czasowego Δt , można obliczyć profil Y z równania metody Eulera:

$$Y_{i+1} = Y_i + \frac{\partial Y}{\partial h} \Delta h \quad (8.22)$$

$$(X_i)^{\tau+\Delta\tau} = (X_i)^\tau + \frac{\partial X}{\partial \tau} \Delta\tau \quad (8.23)$$

P8_1.m

```
%PROJEKT 8.1
%Desorpcja z głębokiego złoża bez oporu wewnętrznego
function desorpcja
clear all; close all
global ky a W S rom eps H
% dane
d=0.006;           % średnica cząstki [m]
rom=1550;          % gęstość materiału [kg/m^3]
sf=1;              % sferyczność cząstki [-]
eps=0.45;          % porowatość złoża [-]
W=0.2;             % natężenie przepływu powietrza [kg/s]
S=1;               % powierzchnia przekroju złoża [m^2]
H=0.8;             % głębokość złoża [m]
X0=1;              % stężenie początkowe w cząstkach [kg/kg]
Y0=0.01;           % stężenie początkowe w powietrzu [kg/kg]

% własności fizykochemiczne
rog=1.21;          % gęstość powietrza [kg/m^3]
mig=18*10^(-6);    % lepkość powietrza [Pas]
Dab=15*10^(-6);    % dyfuzyjność składnika zaadsorbowanego w
powietrzu [m^2/s]
```

```

% obliczenia
u=W/(rog*S);           % prędkość powietrza na pusty aparat [m/s]
Re=u*d*rog/mig;        % liczba Reynoldsa
Sc=mig/(rog*Dab);      % liczba Schmidta
Sh=0.91*Re.^0.49*Sc.^0.33*sf;
                        % liczba Sherwooda
ky=Sh*rog*Dab/d;       % współczynnik wnikania masy [kg/m^2*s]
a=6*(1-eps)/(d*sf);    % powierzchnia właściwa [m^2/m^3]

N=100;
dH=H/N;                % krok przestrzenny [m]
dt=10;                 % krok czasowy [s]
t=4*3600;
X=zeros(N,1);
Y=zeros(N,1);
X(:)=X0;
Y(1,:)=Y0;

%całkowanie układu równań
for k=1:round(t./dt)
    for j=1:N-1
        Y(j+1)=Y(j)+dYdH(X(j),Y(j))*dH;
    end
    for j=1:N
        X(j)=X(j)+dXdH(X(j),Y(j))*dt;
        if X(j) < 0
            X(j)=0; %zabezpieczenie przed wartościami ujemnymi
        end
    end
    if mod(k,90)==0 %pokaż co 90 profil tj. co 900 s = 15 min
        plot(linspace(0,H,N),X(:,1))
        hold on
    end
end
plot(linspace(0,1,N),X(:,1))
axis([0 H 0 X0])
title('Profil uł. masowego składnika A w fazie stałej')
xlabel('H [m]')
ylabel('X [kg/kg]')
%-----
function y3=dYdH(X,Y)
global S W

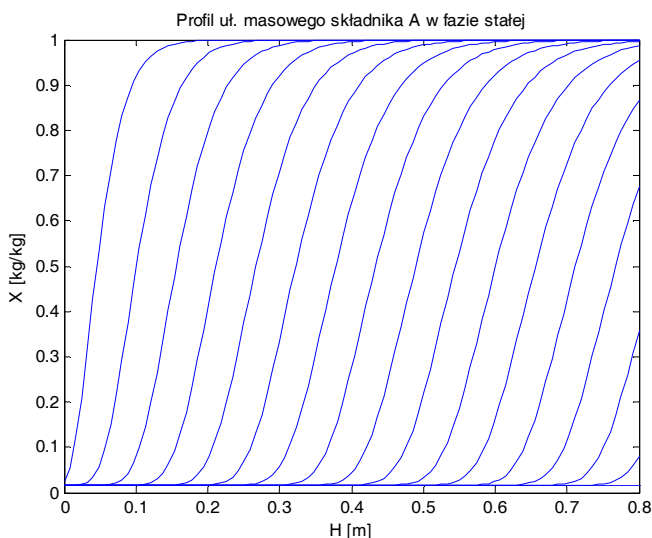
```

```

y3=S/W*wDV(X,Y);
%-----
function y4=dXdt(X,Y)
global rom eps
y4=-wDV(X,Y)/(rom*(1-eps));
%-----
function y1=Yr(X);
% izoterma sorpcji
s1=0.05; s2=0.8; s3=5.5;
y1=s3*(1-exp(-s1*X.^s2));
%-----
function y2=wDV(X,Y)
global ky a
y2=ky*a*(Yr(X)-Y);

```

Wyniki 8.1



Rys. 8.2. Przebieg zmian ułamka masowego składnika A w materiale wzdłuż wysokości złoża. Krzywe pokazane co 15 min trwania procesu

Orientacyjny czas całkowitej desorpcji ze złoża wynosi nieco ponad 4 godziny. Rozszerzeniem tego zadania jest Projekt 8.14, gdzie dodatkowo wprowadzono opór wnikanía masy wewnątrz cząstek stałych.

8.3.2. Stan ustalony

W stanie ustalonym jako zmienne niezależne pozostają jedynie współrzędne przestrzenne. Prostim przypadkiem dwuwymiarowym jest przypadek wymiany ciepła lub masy w przepływie krzyżowym, który zilustrujemy w Projekcie 8.2 (na podstawie pracy Pakowski i Głębowski, 2001, s. 295).

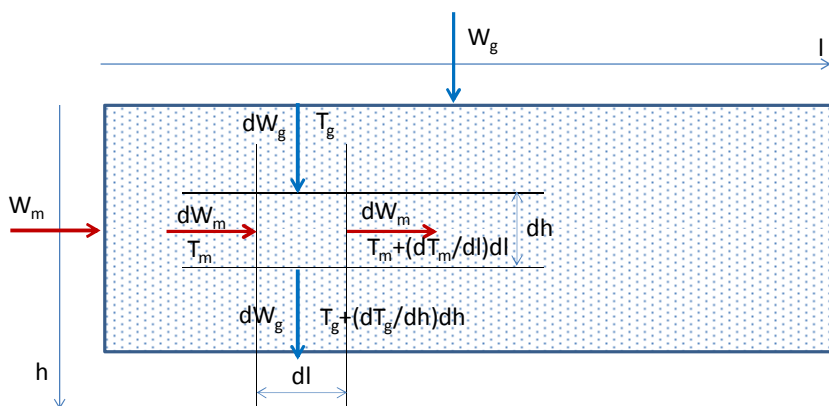
PROJEKT 8.2

Temat: Wymiana ciepła w przepływie krzyżowym

Toolbox: -

Warstwa materiału ziarnistego o grubości H i szerokości B jest chłodzona powietrzem przedmuchiwanym z góry na dół. Materiał porusza się na taśmie siatkowej o długości L . Natężenia masowe przepływu i ciepła właściwe wynoszą odpowiednio: dla materiału W_s i c_s a dla powietrza W_G i c_G . Objętościowy współczynnik wnikania ciepła wynosi α_v . Temperatura wlotowa materiału wynosi T_{m0} , a powietrza T_{g0} . Obliczyć rozkłady przestrzenne temperatur obu mediów. Pozostałe dane zawarte są w treści programu.

Opiszemy przypadek bezprzeponowego kontaktu fazy gazowej i ziarnistej fazy stałej. Przypadek przepływu krzyżowego z tłokowym przepływem obu faz jest przypadkiem dwuwymiarowym, dlatego bilansowanie przeprowadza się dla elementu różniczkowego o wymiarach $dl \times dh$, jak pokazano na rysunku. Warto zauważyć, że do każdego elementu dopływa tylko różniczkowa część łącznego przepływu w danym kierunku.



Rys. 8.3. Schemat bezprzeponowej wymiany ciepła podczas przepływu krzyżowego

Bilanse ciepła dla obu faz przedstawiono poniżej:

- faza stała

$$dW_m c_m T_m - dW_m c_m \left[T_m + \frac{dT_m}{dl} dl \right] - \alpha_v dl dh B(T_m - T_g) = 0 \quad (8.24)$$

- faza gazowa

$$dW_g c_g T_g - dW_g c_g \left[T_g + \frac{dT_g}{dh} dh \right] + \alpha_v dl dh B(T_m - T_g) = 0 \quad (8.25)$$

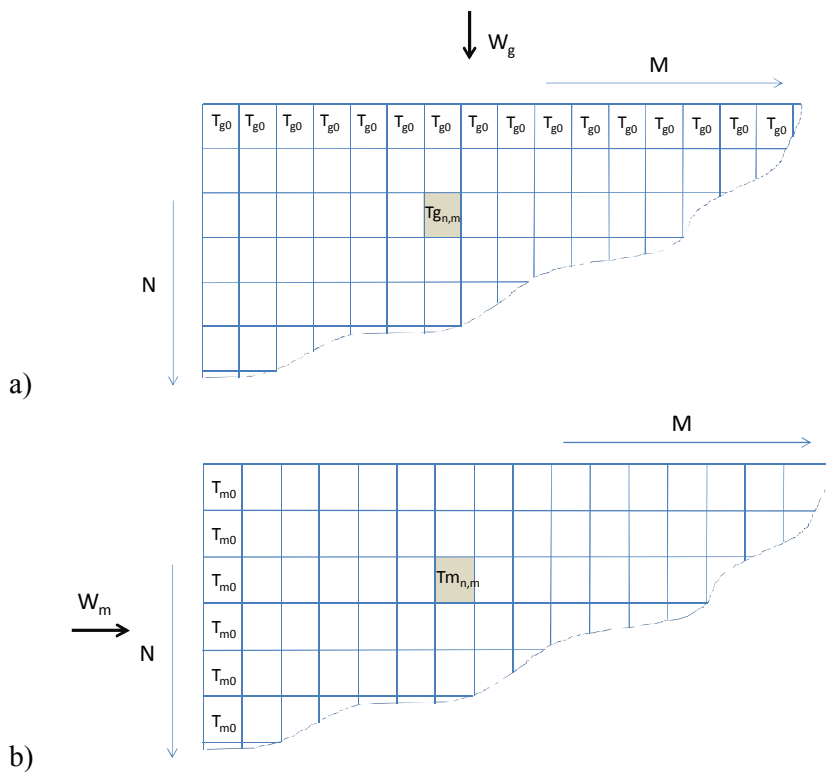
Podstawiając następujące oznaczenia

$$\sigma_H = \frac{dW_m}{dh} \quad i \quad \sigma_L = \frac{dW_g}{dl} \quad (8.26)$$

po podzieleniu przez iloczyn $dh dl$ i podstawieniu powyższych wyrażeń równania (8-23 i 8-24), można zapisać jako:

$$\begin{aligned} \frac{\partial T_m}{\partial l} &= \frac{\alpha_v B(T_m - T_g)}{\sigma_H c_m} \\ \frac{\partial T_g}{\partial h} &= \frac{\alpha_v B(T_m - T_g)}{\sigma_L c_g}. \end{aligned} \quad (8.27)$$

Z uwagi na prostą budowę tych równań można je rozwiązać, stosując metodę jawną Eulera. W tym celu podzielimy obszar na M części w kierunku osi l i na N części w kierunku osi h . Do powstałych komórek przypiszemy macierz temperatur ciała stałego T_m i gazu T_g . W odpowiednich ich komórkach umieścimy warunek brzegowy, jak pokazano na Rys. 8.4.



Rys. 8.4. Macierz: a) temperatur gazu, b) temperatur ciała stałego

Obliczenia należy wykonywać dla kolejnych kolumn obu macierzy obliczając kolumnę macierzy temperatur gazu ze wzoru Eulera postaci:

$$Tg_{n,m} = Tg_{n-1,m} + \frac{\alpha_v B}{\sigma_L c_g} (Tm_{n,m} - Tg_{n-1,m}) \Delta h \quad \text{dla } n = 1, 2 \dots N, \quad (8.28)$$

a następnie kolumnę temperatur materiału ze wzoru Eulera postaci:

$$Tm_{n,m+1} = Tm_{n,m} - \frac{\alpha_v B}{\sigma_H c_m} (Tm_{n,m} - Tg_{n,m}) \Delta l \quad \text{dla } n = 1, 2 \dots N. \quad (8.29)$$

P8_2D.m

```
clear all; close all; clc
%geometria
```

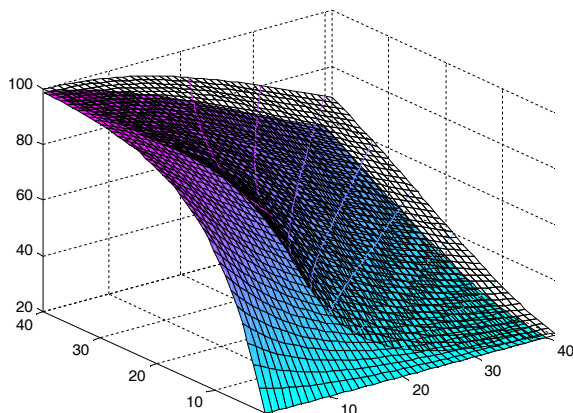
```

H=0.1;          % głębokość złoża [m]
B=1;           % szerokość złoża [m]
L=20;          % długość złoża [m]
% ciało stałe
Wm=3;          % natężenia przepływu [kg/s]
cm=0.8;        % ciepło właściwe [kJ/kg*K]
tm0=100;       % temperatura wlotowa [C]
av=5;          % objętościowy współczynnik wnikania ciepła
[kW/m^3*K]
% gaz
Wg=2;          % natężenia przepływu [kg/s]
cg=1.2;        % ciepło właściwe [kJ/kg*K]
tg0=20;        % temperatura wlotowa [C]
% rozwiązanie
sigh=Wm/H;     sigl=Wg/L;
N=40;          M=40;
dh=H/N;        dl=L/M;
av=av*B;
tm=zeros(N,M);
tg=zeros(N,M);
tm(:,1)=tm0;
tg(1,:)=tg0;
% obliczenie temperatury gazu i ciała stałego metodą Eulera
for m=1:M
    for n=2:N
        tg(n,m)=tg(n-1,m)+((dh*av)/(sigl*cg))*(tm(n,m)-tg(n-
1,m));
    end
    for n=1:N
        tm(n,m+1)=tm(n,m)-((dl*av)/(sigh*cm))*(tm(n,m)-tg(n,m));
    end
end
% wizualizacja wyników
figure(1)
contour3(tm)
surface(tm,'EdgeColor',[0 0 0],'FaceColor','none')
colormap cool
hold on
surface(tg)
hold off
figure(2)
[C,h] = contourf(tm);
clabel(C,h)

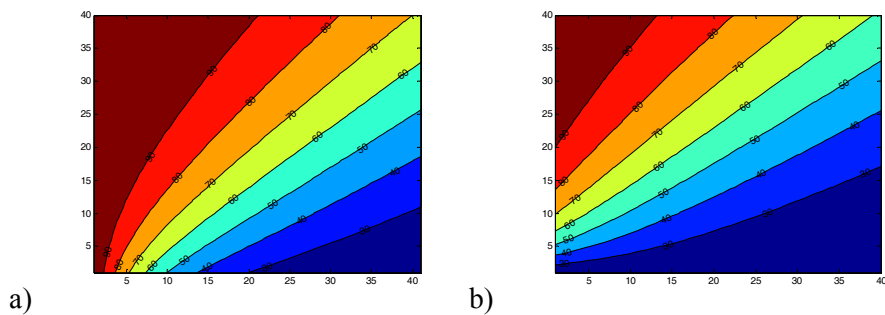
```

```
figure(3)
[C,h] = contourf(tg);
clabel(C,h)
```

Wyniki 8.2



Rys. 8.5. Wykres przestrzenny temperatury materiału i powietrza. Na osi pionowej temperatura w °C



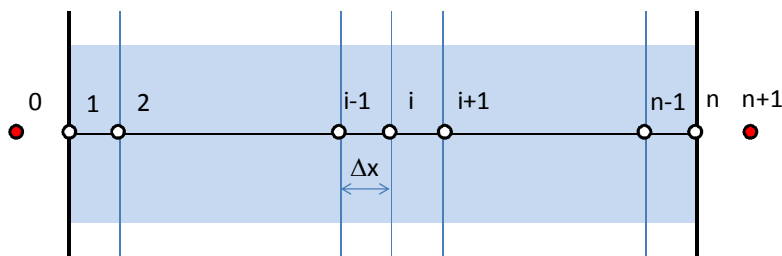
Rys. 8.6. Mapa zmian: a) temperatury materiału, b) temperatury powietrza

Komentarz: metoda powyższa jest metodą przybliżoną. Dla przepływu krzyżowego stosowane są również dokładniejsze metody macierzowe (zob. *Zaleski, 1987*).

8.4. Rozwiązywanie równań typu parabolicznego

8.4.1. Metoda linii

Metoda linii służy do rozwiązywania równań typu parabolicznego, które występują np. w procesach nieustalonego przepływu, przewodzenia ciepła lub dyfuzji. Polega ona na przekształceniu równania PDE na układ równań ODE przez dyskretyzację (zamiannę na iloraz różnicowy określony w węzłach siatki przestrzennej) pochodnych cząstkowych po współrzędnych przestrzennych. Pochodną po czasie pozostawia się bez zmian. MATLAB dysponuje dobrymi integratorami równań ODE, które należy zastosować do rozwiązania powstałego układu równań. Na Rys. 8.7 pokazano numerację węzłów dla przypadku jednowymiarowego.



Rys. 8.7. Sposób dyskretyzacji przestrzeni jednowymiarowej

Powstała jednowymiarowa siatka jest najczęściej równomierna o stałym odstępem między węzłami Δx . Węzły 1 i n leżą na powierzchniach ograniczających obszar. Dla tych węzłów formułuje się odpowiednie warunki brzegowe. Do tego celu służą pomocnicze fikcyjne węzły 0 i $n+1$, leżące poza analizowanym obszarem. Warstwy między węzłami to komórki.

Dla objaśnienia metodyki dalszego postępowania w metodzie linii posłużymy się przypadkiem nieustalonego jednowymiarowego przewodzenia ciepła we współrzędnych kartezjańskich. Równanie Fouriera ma wtedy postać:

$$\frac{\partial T}{\partial \tau} = a \frac{\partial^2 T}{\partial x^2}. \quad (8.30)$$

W dowolnym i -tym węźle utworzonej siatki zastosujemy następujące centralne trójpunktowe przybliżenie drugiej pochodnej

$$\frac{\partial^2 T_i}{\partial x^2} = \frac{T_{i+1} - 2T_i + T_{i-1}}{(\Delta x)^2}. \quad (8.31)$$

Otrzymuje się wówczas równanie ODE

$$\frac{dT_i}{d\tau} = a \frac{T_{i+1} - 2T_i + T_{i-1}}{(\Delta x)^2} \quad (8.32)$$

ślusne dla węzłów siatki o numerach $i = 2, 3, \dots, n-1$. Dodatkowo należy je uzupełnić równaniami wynikającymi z warunków brzegowych dla węzłów o numerach 1 i n .

Przyjmijmy, że powierzchnia z węzłem 1 jest izolowana, a powierzchnia z węzłem n wymienia ciepło z otoczeniem w sposób konwekcyjny.

Komórka izolowana – gradient temperatury w ciele wynosi tutaj zero, a więc, korzystając z przybliżenia pierwszej pochodnej ilorazem różnicowym centralnym, mamy:

$$\frac{dT_1}{dx} = \frac{T_0 - T_2}{2\Delta x} = 0, \quad (8.33)$$

stąd

$$T_0 = T_2. \quad (8.34)$$

Wstawiając do wzoru na drugą pochodną (8.31), mamy

$$\frac{dT_1}{d\tau} = 2a \frac{T_2 - T_1}{(\Delta x)^2} \quad (8.35)$$

Komórka wymieniająca ciepło z otoczeniem – warunek brzegowy ma tu postać

$$-\lambda \frac{dT_n}{dx} = \alpha(T_n - T_f). \quad (8.36)$$

Tak więc

$$\frac{dT_n}{dx} = -\frac{\alpha}{\lambda}(T_n - T_f) = \frac{T_{n+1} - T_{n-1}}{2\Delta x}, \quad (8.37)$$

stąd

$$T_{n+1} = T_{n-1} - 2\Delta x \frac{\alpha}{\lambda}(T_n - T_f). \quad (8.38)$$

Wstawiając do (8.31), mamy

$$\frac{dT_n}{d\tau} = 2a \frac{T_{n-1} - T_n - \frac{\Delta x \alpha}{\lambda} (T_n - T_f)}{(\Delta x)^2}. \quad (8.39)$$

Powstały układ równań ODE po rozwiązaniu daje linie przebiegów czasowych poszukiwanej funkcji dla każdego węzła siatki. Stąd nazwa – metoda linii. Tę metodę można prześledzić w poniższych projektach.

PROJEKT 8.3

Temat: Nieustalony ruch ciepła w płycie nieskończonej – metoda linii
Toolbox: -

Nieskończona płyta płaska o grubości $L = 0.1$ m, przewodności cieplnej $\lambda = 0.00375$ kW/(m K), cieple właściwym $c_p = 1.5$ kJ/(kg K) i gęstości $\rho = 2500$ kg/m³, wymienia z jednej strony ciepło z gazem o temperaturze $t_g = 100^\circ\text{C}$. Współczynnik wnikania ciepła wynosi $\alpha = 0.05$ kW/(m² K). Z drugiej strony płyta jest izolowana. Początkowa temperatura płyty wynosi $t_0 = 20^\circ\text{C}$. Obliczyć rozkłady temperatur w płycie w miarę upływu czasu.

Wykorzystamy metodę linii. Obliczenia wykonamy na zmiennych wymiarowych. Obliczymy najpierw dyfuzyjność cieplną

$$a = \frac{\lambda}{c_p \rho} = \frac{3.75 \cdot 10^{-3}}{1.5 \cdot 2500} = 1 \cdot 10^{-6} \text{ m}^2 / \text{s} \quad (8.40)$$

Równanie przewodzenia ciepła dla płyty płaskiej ma postać

$$\frac{\partial T}{\partial \tau} = a \frac{\partial^2 T}{\partial x^2} \quad (8.41)$$

z warunkami brzegowymi

$$\left. \frac{\partial T}{\partial \tau} \right|_{x=0} = 0 \quad (8.42)$$

$$-\lambda \left. \frac{\partial T}{\partial \tau} \right|_{x=L} = \alpha(T - T_f)$$

i warunkiem początkowym

$$T_0 = 20 \quad \text{dla} \quad 0 < x < L \quad \text{i} \quad \tau = 0. \quad (8.43)$$

W celu dyskretyzacji równania musimy rozciągnąć siatkę na rozpatrywanym obszarze z węzłami ponumerowanymi tak, aby pierwszy węzeł wypadł na powierzchni izolowanej, a n-ty węzeł na powierzchni wymieniającej ciepło. Położenie punktu opisujemy zależnością

$$x_i = (i - 1)\Delta x, \quad (8.44)$$

gdzie

$$\Delta x = \frac{L}{n} \quad (8.45)$$

oraz i – numer danego węzła, n – liczba komórek siatki, Δx – szerokość komórki, L – szerokość obszaru.

Następnym krokiem jest zamiana pochodnych na ilorazy różnicowe

$$\begin{aligned} \frac{\partial^2 T_i}{\partial x^2} &\approx \frac{(T_{i+1} - T_i) - (T_i - T_{i-1}))}{(\Delta x)^2} = \frac{T_{i+1} - 2T_i + T_{i-1}}{(\Delta x)^2} \\ \frac{\partial T_i}{\partial x} &\approx \frac{\Delta T}{\Delta x} = \frac{T_{i+1} - T_{i-1}}{x_{i+1} - x_{i-1}} = \frac{T_{i+1} - T_{i-1}}{2\Delta x}. \end{aligned} \quad (8.46)$$

Wyprowadźmy teraz równania dla węzła 1 (powierzchnia izolowana) oraz węzła n (powierzchnia międzyfazowa).

Dla komórki izolowanej:

$$\left. \frac{\partial^2 T}{\partial x^2} \right|_{x=0} \approx \frac{T_2 - 2T_1 + T_0}{(\Delta x)^2} = \frac{T_2 - 2T_1 + T_2}{(\Delta x)^2} = 2 \frac{T_2 - T_1}{(\Delta x)^2}. \quad (8.47)$$

Dla komórki nagrzewanej:

$$-\lambda \left. \frac{T_{n+1} - T_{n-1}}{2\Delta x} \right|_{x=L} = \alpha(T_n - T_f), \quad (8.48)$$

co można przekształcić do

$$T_{n+1} = T_{n-1} - \frac{2\Delta x \alpha}{\lambda} (T_n - T_f), \quad (8.49)$$

gdzie $n+1$ oznacza numer fikcyjnego zewnętrznego węzła. Wstawiając tę temperaturę do wzoru na drugą pochodną dla węzła n mamy

$$\left. \frac{\partial^2 T}{\partial x^2} \right|_{x=L} = 2 \frac{T_{n-1} - T_n - \frac{\Delta x \alpha}{\lambda} (T_n - T_f)}{(\Delta x)^2}. \quad (8.50)$$

Implementację tej metody zawiera przedstawiony poniżej plik.

P8_3.m

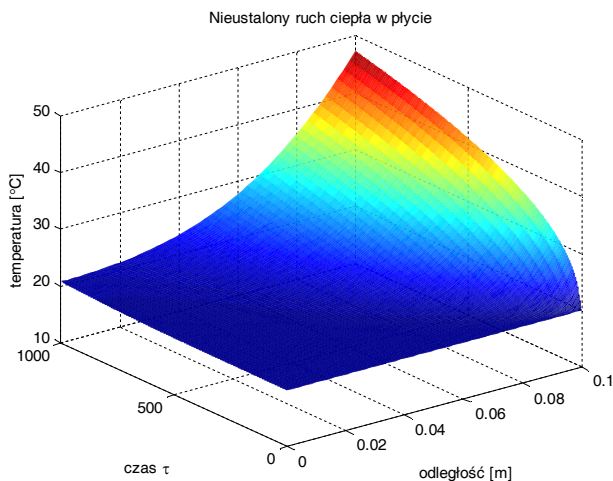
```
%PROJEKT 8.3
%Przewodzenie ciepła w płycie rozwiązane metodą linii
function plyta_lin
clear all; close all; clc
global n L
% podział siatki
n=50;
% warunek początkowy
t0=20;
i=[1:n];
temp(i)=t0;
tmax=1000;
% wywołanie solvera ODE
[t,T]=ode45(@r,[0 tmax],temp);
% prezentacja wyników
xi=linspace(0,L,n);
[w,k]=size(T);
yi=linspace(0,tmax,w);
[XI,YI]=meshgrid(xi,yi);
figure(1)
surf(XI,YI,T)
title('Nieustalony ruch ciepła w płycie')
xlabel('odległość [m]')
ylabel('czas [s]')
zlabel('temperatura [\textcircled{C}])')
hidden on
shading interp
```

```

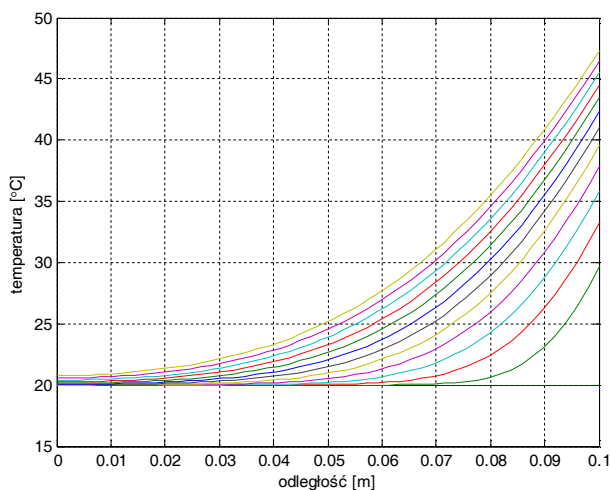
figure(2)
[w,k]=size(T)
plot(xi,T([1:100:w],:))
xlabel('odległość [m]')
Ylabel('temperatura [\text{C}]')
grid on
%-----
function dtdtemp=r(t,temp)
% zdefiniowany układ równań metody linii
global n L
L=0.1;           %grubość płyty
a=10^(-6);       %dyfuzyjność cieplna, m2/s
alfa=0.05;       %współczynnik wnikanie ciepła, kW/(m2 K)
tg=100;          %temperatura zewnętrzna, C
cp=1.5;          %ciepło właściwe ciała, kJ/(kg K)
ro=2500;         %gęstość ciała, kg/m3
dL=L/n; f=a/dL.^2;
p(1)=2*f*(temp(2)-temp(1));
for i=2:n-1
    p(i)=f*(temp(i+1)-2*temp(i)+temp(i-1));
end
p(n)=2*f*(temp(n-1)-temp(n))+2*alfa./(dL*cp*ro)*(tg-temp(n));
dtdtemp=[p'];

```

Wyniki 8.3



Rys. 8.8. Rozkład temperatur w płycie – wykres 3D



Rys. 8.9. Rozkład temperatur w płycie dla czasów rosnących co 100 s.

PROJEKT 8.4

Temat: Nieustalony ruch ciepła w walcu nieskończonym – metoda linii
 Toolbox: -

Nieskończony walec o promieniu $R = 0.1$ m, przewodności cieplnej $\lambda = 0.00375$ kW/(m K), ciepłe właściwym $c_p = 1.5$ kJ/(kg K) i gęstości $\rho = 2500$ kg/m³, wymienia ciepło z gazem o temperaturze $t_g = 100^\circ\text{C}$. Współczynnik wnikanie ciepła wynosi $\alpha = 0.05$ kW/(m² K). Temperatura początkowa walca wynosi $t_0 = 20^\circ\text{C}$. Obliczyć rozkłady temperatur T w walcu w miarę upływu czasu τ .

Równanie przewodzenia ciepła dla walca

$$\frac{\partial T}{\partial \tau} = a \left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r} \frac{\partial T}{\partial r} \right), \quad (8.51)$$

korzystając z zależności z poprzedniego zadania, wyprowadźmy równania użyteczne w metodzie linii.

– dla komórki w osi symetrii

$$\frac{\partial T}{\partial \tau} = 2a \frac{T_1 - T_0}{(\Delta r)^2}, \quad (8.52)$$

– dla komórki wewnętrznej

$$\frac{\partial T}{\partial \tau} = a \left(\frac{T_{i+1} - 2T_i + T_{i-1}}{(\Delta r)^2} + \frac{1}{i \Delta r} \frac{T_{i+1} - T_{i-1}}{2\Delta r} \right), \quad (8.53)$$

– dla komórki zewnętrznej nagrzewanej

$$-\lambda \frac{T_{n+1} - T_{n-1}}{2\Delta r} \Big|_{r=R} = \alpha (T_n - T_f). \quad (8.54)$$

Po przekształceniu otrzymujemy

$$T_{n+1} = T_{n-1} - \frac{2\Delta r \alpha}{\lambda} (T_n - T_f) \quad (8.55)$$

oraz

$$\frac{\partial T}{\partial \tau} = a \left(\frac{T_{n-1} - \frac{2\Delta r \alpha}{\lambda} (T_n - T_f) - 2T_n + T_{n-1}}{(\Delta r)^2} + \right. \\ \left. + \frac{1}{N\Delta r} \frac{T_{n-1} - 2\frac{\Delta r \alpha}{\lambda} (T_n - T_f) - T_{n-1}}{2\Delta r} \right). \quad (8.56)$$

Ostatecznie otrzymujemy

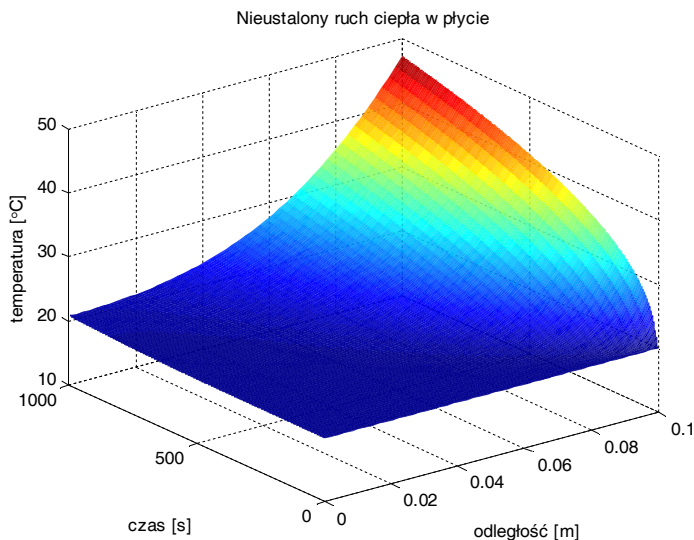
$$\frac{\partial T}{\partial \tau} = a \left(2 \frac{T_{n-1} - \frac{\Delta r \alpha}{\lambda} (T_n - T_f) - T_n}{(\Delta r)^2} + \right. \\ \left. + \frac{1}{N\Delta r} \frac{-\frac{\Delta r \alpha}{\lambda} (T_n - T_f)}{\Delta r} \right) \quad (8.57)$$

P8_4.m

```
%PROJEKT 8.4
%Przewodzenie ciepła w walcu rozwiązane metodą linii
function walec_lin
clear all; close all; clc
global n R
% podział siatki
n=50;
% warunek początkowy
t0=20;
for i=1:n
temp(i)=t0;
end
% wywołanie solvera ODE
[t,T]=ode45(@r,[0 1000],temp);
% prezentacja wyników
xi=linspace(0,R,n);
[w,k]=size(T);
yi=linspace(0,200,w);
[XI,YI]=meshgrid(xi,yi);
figure(1)
surf(XI,YI,T)
title('Nieustalony ruch ciepła w walcu')
xlabel('promień [m]')
ylabel('czas [s]')
zlabel('temperatura [\textcircled{C}])')
hidden on
shading interp
%-----
function dtdtemp=r(t,temp)
% zdefiniowany układ równań 'metody linii'
global n t0 R
R=0.1; a=10^(-6); alfa=0.05; tg=100; cp=1.5; ro=2500;
dR=R/n; f=a/dR.^2; lamb=a*cp*ro;

p(1)=a.*((2.*temp(2)-2.*temp(1))./(dR.^2));
for i=2:n-1
    p(i)=a.*(((temp(i+1)-2*temp(i)+temp(i-1))./(dR.^2))+...
        (1/(i.*dR)).*(temp(i+1)-temp(i-1))./(2.*dR));
end
wT=((2.*dR.*alfa)./lamb).*(temp(n)-tg);
p(n)=a.*((2.*temp(n-1)-2.*temp(n)-wT)./(dR.^2)+(1/(n.*dR)).*(-
wT)./(2.*dR));
```

dtdtemp=[p'];



Rys. 8.10. Rozkład temperatur w walcu

PROJEKT 8.5

Temat: Nieustalony ruch ciepła w kuli – metoda linii

Toolbox: -

Kula o promieniu $R = 0.1$ m, przewodności cieplnej $\lambda = 0.00375$ kW/(m K), cieple właściwym $c_p = 1.5$ kJ/(kg K) i gęstości $\rho = 2500$ kg/m³, wymienia ciepło z gazem o temperaturze $T_g = 100^\circ\text{C}$. Współczynnik wnikania ciepła wynosi $\alpha = 0.05$ kW/(m² K). Temperatura początkowa kuli wynosi $T_g = 20^\circ\text{C}$. Obliczyć rozkłady temperatur w kuli w miarę upływu czasu.

Równanie przewodzenia ciepła dla kuli

$$\frac{\partial T}{\partial \tau} = a \left(\frac{\partial^2 T}{\partial r^2} + \frac{2}{r} \frac{\partial T}{\partial r} \right), \quad (8.58)$$

po dyskretyzacji otrzymujemy

– dla komórki w osi symetrii

$$\frac{\partial T}{\partial \tau} = a \frac{2T_1 - 2T_0}{(\Delta r)^2}, \quad (8.59)$$

– dla komórki wewnętrznej

$$\frac{\partial T}{\partial \tau} = a \left(\frac{2T_{i+1} - 2T_i + T_{i-1}}{(\Delta r)^2} + \frac{2}{i \Delta r} \frac{T_{i+1} - T_{i-1}}{2\Delta r} \right), \quad (8.60)$$

– dla nagrzewanej komórki zewnętrznej

$$-\lambda \left(\frac{T_{n+1} - T_{n-1}}{2\Delta r} \right) \Big|_{x=R} = \alpha (T_n - T_f) \quad (8.61)$$

co daje

$$T_{n+1} = T_{n-1} - \frac{2\Delta r \alpha}{\lambda} (T_n - T_f) \quad (8.62)$$

$$\frac{\partial T}{\partial \tau} = a \left(\frac{T_{n-1} - \frac{2\Delta r \alpha}{\lambda} (T_n - T_f) - 2T_n + T_{n-1}}{(\Delta r)^2} + \frac{2}{N \Delta r} \frac{T_{n-1} - 2 \frac{\Delta r \alpha}{\lambda} (T_n - T_f) - T_{n-1}}{2\Delta r} \right) \quad (8.63)$$

$$\frac{\partial T}{\partial \tau} = a \left(\frac{2 \left(T_{N-1} \frac{\Delta r \alpha}{\lambda} (T - T_f) - T_N \right)}{(\Delta r)^2} + \frac{2}{N \Delta r} \frac{-\frac{\Delta r \alpha}{\lambda} (T - T_f)}{\Delta r} \right). \quad (8.64)$$

Ostatecznie otrzymujemy

$$\frac{\partial T}{\partial \tau} = 2a \left(\frac{T_{n-1} - \frac{\Delta r \alpha}{\lambda} (T_n - T_f) - T_n}{(\Delta r)^2} + \frac{1}{N \Delta r} - \frac{\Delta r \alpha}{\lambda} \frac{(T_n - T_f)}{\Delta r} \right) \quad (8.65)$$

P8_5.m

```
%PROJEKT 8.5
%Przewodzenie ciepła w kuli rozwiązane metodą linii
function kula_lin
clear all; close all; clc
global n R
% podział siatki
n=50;
% warunek początkowy
t0=20;
for i=1:n
temp(i)=t0;
end
% wywołanie solvera ODE
[t,T]=ode45(@r,[0 200],temp);
% prezentacja wyników
xi=linspace(0,R,n);
[w,k]=size(T);
yi=linspace(0,200,w);
[XI,YI]=meshgrid(xi,yi);
figure(1)
surf(XI,YI,T)
title('Nieustalony ruch ciepła w kuli')
xlabel('odległość x')
ylabel('czas t')
zlabel('temperatura T')
hidden on
shading interp

%-----
function dtdtemp=r(t,temp)
% zdefiniowany układ równań metody linii
```

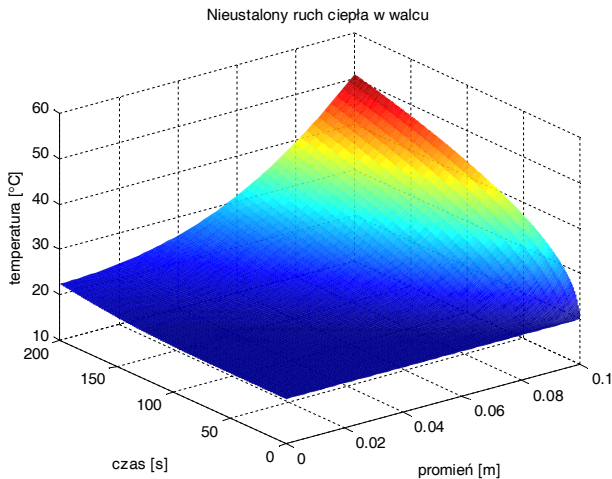
```

global n R
R=0.1; a=10^(-6); alfa=0.05; tg=100; cp=1.5; ro=2500;
dR=R/n; f=a/dR.^2; lamb=a*cp*ro;

p(1)=a.*((2.*temp(2)-2.*temp(1))./(dR.^2));
for i=2:n-1
    p(i)=a.*(((temp(i+1)-2*temp(i)+temp(i-1))./(dR.^2))+...
        (2/(i.*dR)).*(temp(i+1)-temp(i-1))./(2.*dR)));
end
wT=((2.*dR.*alfa)./lamb).*(temp(n)-tg);
p(n)=a.*((2.*temp(n-1)-2.*temp(n)-wT)./(dR.^2)+(2/(n.*dR)).*(-
wT)./(2.*dR));
dtdtemp=[p'];

```

Wyniki 8.5



Rys. 8.11. Rozkład temperatury w kuli

8.4.2. Wykorzystanie integratora pdepe

Integrator ten jest przeznaczony do rozwiązywania zagadnień początkowych dla równań różniczkowych cząstkowych typu mieszanego paraboliczno-eliptycznego w przestrzeni jednowymiarowej. Rozwiązuje on jednowymiarowe równania PDE o ogólnej postaci:

$$c\left(x, \tau, u, \frac{\partial u}{\partial x}\right) \frac{\partial u}{\partial \tau} = x^{-m} \frac{\partial}{\partial x} \left(x^m f\left(x, \tau, u, \frac{\partial u}{\partial x}\right) \right) + s\left(x, \tau, u, \frac{\partial u}{\partial x}\right), \quad (8.66)$$

gdzie $\tau_0 \leq \tau \leq \tau_f$, $a \leq x \leq b$. Wykładnik m może przyjmować wartości 0, 1 lub 2 w zależności od geometrii (płyta płaska, walec, kula). Występujący po prawej stronie człon

$$s\left(x, \tau, u, \frac{\partial u}{\partial x}\right) \quad (8.67)$$

jest członem źródłowym. Funkcja f to strumień wielkości u na jednostkę czasu i jednostkę powierzchni.

W kodzie programu praktycznie definiujemy tylko funkcje c , f i s . Następnie określamy warunek początkowy

$$u(x, \tau_0) = u_0(x) \quad (8.68)$$

oraz warunek brzegowy postaci

$$p(x, \tau, u) + q(x, \tau) \cdot f\left(x, \tau, u, \frac{\partial u}{\partial x}\right) = 0. \quad (8.69)$$

Przykładowe wywołanie integratora ma postać

```
sol = pdepe(m,pdefun,icfun,bcfun,xmesh,tspan)
```

gdzie:

- m** określa geometrię: dla płyty $m = 0$, dla walca $m = 1$, dla kuli $m = 2$
- pdefun** funkcja, w której zdefiniowane jest równanie różniczkowe cząstkowe lub układ równań różniczkowych cząstkowych (8.66). Uwaga: definiujemy tylko zmienne c , f , s . Jeśli rozwiązujemy układ równań to zmienne c , f , s muszą być zdefiniowane jako wektory pionowe.
- icfun** funkcja, w której zdefiniowane są warunki początkowe (8.68).
- bcfun** funkcja, w której zdefiniowane są warunki brzegowe (8.69).
- tspan** wektor zawierający punkty czasowe, w których chcemy uzyskać wartość obliczonej funkcji; w przypadku braku **tspan** solver ustali je samoczynnie.
- xmesh** wektor zawierający węzły siatki - umożliwia to zagęszczenie siatki w miejscach o dużej zmienności funkcji; w przeciwnym razie solver użyje siatki równomiernej.

Działanie tej funkcji ilustrują dwa kolejne projekty.

PROJEKT 8.6

Temat:	Nieustalone przewodzenie wzdłużne ciepła w pręcie chłodzonym na obwodzie
Toolbox:	-

Do pieca o temperaturze 1020°C włożono jednym końcem okrągły pręt stalowy o średnicy 20 mm i długości 0.5 m. Dane materiałowe w listingu. Pręt jest chłodzony przez opływające go prostopadle powietrze o temperaturze 20°C. Określić rozkład temperatur pręta po upływie 1 godz (*Kudra, 1985*, ss. 321-324).

Przy założeniu, że średnica pręta jest mała, równanie bilansowe dla tego przypadku jest łatwe do wyprowadzenia i przyjmuje postać

$$\frac{\partial T}{\partial \tau} = a \frac{\partial^2 T}{\partial x^2} - \frac{4\alpha}{c_p \rho D} (T - T_f), \quad (8.70)$$

gdzie a – dyfuzyjność cieplna [m^2/s], α – współczynnik wnikania ciepła [$\text{kW}/(\text{m}^2\text{K})$], D – średnica pręta [m], c_p – ciepło właściwe [$\text{kJ}/(\text{kg K})$], ρ – gęstość [kg/m^3].

Równanie 8.70 należy zapisać w postaci stosowanej w pdepe

$$c_p \rho \frac{\partial T}{\partial \tau} = \frac{\partial}{\partial x} \left(\lambda \frac{\partial T}{\partial x} \right) - \frac{4\alpha}{D} (T - T_f), \quad (8.71)$$

wtedy mamy

$$c = c_p \rho \quad f = \lambda \frac{\partial T}{\partial x} \quad s = -\frac{4\alpha}{D} (T - T_f). \quad (8.72)$$

Warunki brzegowe należy przedstawić w postaci 8.69.

Na końcu lewym (l) zanurzonym w piecu temperatura wynosi $T_1 = 1020^\circ\text{C}$, znika więc człon zawierający strumień czyli $q_l = 0$ natomiast $p_l = u_l - T_1$, bo funkcja szukana u na lewym krańcu (u_l) ma wartość T_1 . Na prawym końcu (r) mamy warunek Robina bo koniec pręta jest chłodzony powietrzem przez konwekcję. Wtedy więc

$$-\lambda \frac{\partial T}{\partial \tau} = \alpha (T - T_f). \quad (8.73)$$

Przenosząc strumień na prawą stronę, mamy

$$0 = \alpha(T - T_f) + \lambda \frac{\partial T}{\partial \tau}. \quad (8.74)$$

Tak więc $pr = \alpha(T - T_f)$ a $qr = 1$ bo drugi człon to strumień f.

P8_6.m

%PROJEKT 8.6

%Przewodzenie ciepła w pręcie z członem źródłowym

function pret

clear all; close all

global cp ro lambda alfa D T1 Tf

% płyta m=0; walec m=1; kula m=2

m=0;

n=50;

%dane

cp=460; %ciepło właściwe, J/(kg K)

ro=7850; %gęstość, kg/m3

lambda=48; %przewodność cieplna, W/(m K)

alfa=5; %współczynnik wnikanía ciepła, W/(m2 K)

D=0.02; %średnica pręta, m

T1=1020; %temperatura końca w piecu, C

Tf=20; %temperatura powietrza, C

L=0.5; %długość pręta, m

t=3600; %czas nagrzewania, s

x = linspace(0,L,n); %długość [m]

t = linspace(0,t,n); %czas [s]

% wywołanie solvera PDE

sol = pdepe(m,@pdex1pde,@pdex1ic,@pdex1bc,x,t);

u = sol(:,:,1);

%pokaż temperaturę na końcu pręta po zakończeniu procesu

disp('Temperatura na końcu pręta wynosi [C]');

disp(u(end,end))

% prezentacja wyników

```

figure(1)
surf(x,t,u)
title('Nagrzewanie pręta w piecu')
xlabel('Odległość [m]')
ylabel('Czas [s]')
zlabel('Temperatura [C]')

figure(2)
[xi,ti]=meshgrid(x,t);
[C,h] = contour(xi,u,ti);
set(h,'ShowText','on','LineWidth',2,'TextStep',get(h,'LevelStep')
)*2)
grid on
ylabel('Temperatura [C]')
xlabel('Odległość [m]')

% -----
function [c,f,s] = pdex1pde(x,t,u,DuDx)
global cp ro lambda alfa D Tf

% zdefiniowanie równania
c = cp*ro;
f = lambda*DuDx;
s = -4*alfa/D*(u-Tf);
% -----
function u0 = pdex1ic(x)
% warunek początkowy
u0 = 20; %Tf
% -----
function [pl,q1,pr,qr] = pdex1bc(xl,ul,xr,ur,t)
global alfa T1 Tf
% warunki brzegowe l-lewy, r-prawy
pl = T1-ul;
q1 = 0;
pr = alfa*(ur-Tf);
qr = 1;

```

Wyniki 8.6

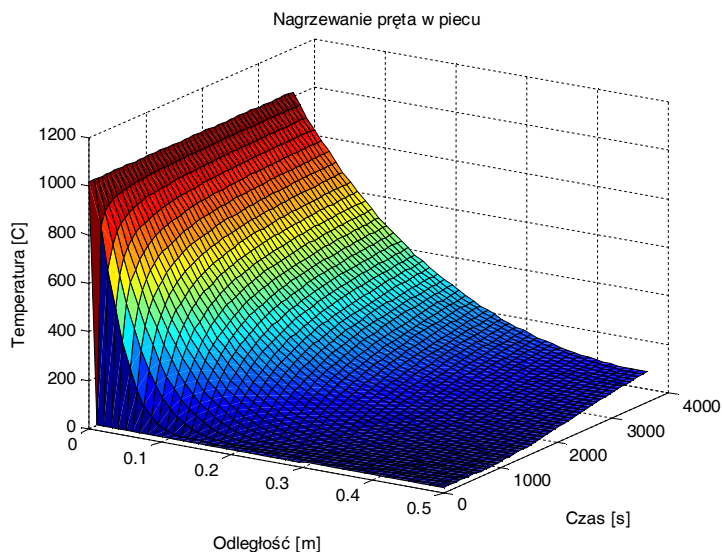
Program wydrukował na ekranie głównym:

```

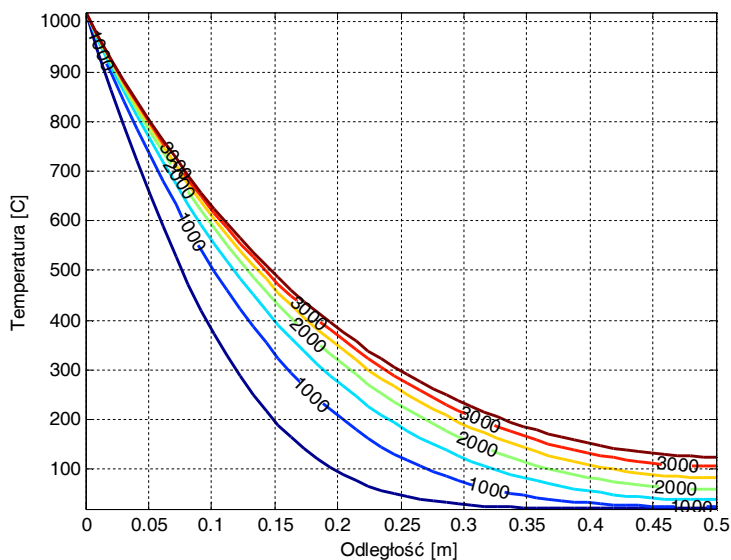
Temperatura na końcu pręta wynosi [C]
128.2363

```

Rozkłady temperatur pokazują rysunki:



Rys. 8.12. Nieustalone nagrzewanie pręta chłodzonego powietrzem



Rys. 8.13. Rozkład temperatury dla poszczególnych czasów pokazanych na krzywych w sekundach

PROJEKT 8.7

Temat:	Reakcja nieodwracalna w reaktorze rurowym z dyspersją wzdłużną
Toolbox:	-

Do strumienia płynącego przez reaktor rurowy o długości L zaczęto doprowadzać reagent o stężeniu c_0 . Przereagowuje on w reakcji nieodwracalnej I rzędu o stałej szybkości k . Ciecz w reaktorze płynie z prędkością u_t . Współczynnik dyspersji wzdłużnej wynosi E . Oblicz rozkłady stężeń wzdłuż długości reaktora w czasie $\tau_{\max}$ od momentu rozruchu. Dane liczbowe są zawarte w kodzie pokazanym poniżej.

Równanie bilansowe elementu objętości reaktora o długości dx przyjmuje postać

$$\frac{\partial c}{\partial \tau} = E \frac{\partial^2 c}{\partial x^2} - u_t \frac{\partial c}{\partial x} - kc \quad (8.75)$$

gdzie c – stężenie reagenta [kmol/m^3], E – współczynnik dyspersji wzdłużnej [m^2/s], k – współczynnik szybkości reakcji [$1/\text{s}$], u_t – prędkość cieczy w reaktorze [m/s]. Poszczególne człony w tym równaniu to (od lewej): akumulacja, człon dyfuzyjny, człon adwekcyjny, człon upustowy. Ponieważ w kanonicznej postaci równania rozwiązywanego przez integrator `pdepe` nie ma członu adwekcyjnego musi on wejść do członu źródłowego s . Wobec tego

$$c = 1 \quad f = E \frac{\partial c}{\partial x} \quad s = -u_t \frac{\partial c}{\partial x} - kc, \quad (8.76)$$

Warunek początkowy jest następujący

$$\begin{array}{lll} \text{dla } \tau = 0 & \text{ i } & \begin{array}{ll} \text{dla } x = 0 & c = c_0 \\ \text{dla } x > 0 & c = 0 \end{array} \end{array} \quad (8.77)$$

Warunki brzegowe są następujące

$$\begin{array}{lll} & \text{dla } x = 0 & c = c_0 \\ \text{dla } \tau > 0 & \text{ i } & \begin{array}{ll} \text{dla } x = L & \frac{\partial c}{\partial x} = 0 \end{array} \end{array} \quad (8.78)$$

Ostatni warunek, zwany warunkiem Danckwertsza, jest warunkiem najczęściej stosowanym na wylocie z reaktora rurowego.

P8_7.m

```
%PROJEKT 8.7
%Rozwiązanie równania adwekcji z dyspersją wzdłużną
function adwekcja
clear all; close all
global c0 E k ut
%dane
ut=4.233e-4;           %prędkość przepływu, m/s
E=4.115e-5;           %współczynnik dyspersji wzdłużnej, m2/s
k=1.157e-5;           %stała szybkości reakcji,1/s
c0=0.1;               %stężenie wlotowe, kmol/m3
L=38.1;               %długość reaktora, m
t=1800*60;            %czas, s

% płyta m=0; walec m=1; kula m=2
m = 0;
ns=200;
nt=250;
x = linspace(0,L,ns); %długość [m]
t = linspace(0,t,nt); %czas [s]

% wywołanie solvera PDE
sol = pdepe(m,@pdex1pde,@pdex1ic,@pdex1bc,x,t);
u = sol(:, :, 1);
disp(u(end,end)) %pokaż stężenie końcowe na wylocie
% prezentacja wyników
figure(1)
surf(x,t,u)
shading interp
title('Rozkład stężenia w reaktorze')
xlabel('Długość [m]')
ylabel('Czas [s]')
zlabel('Stężenie [kmol/m3]')
figure(2)
[xi,ti]=meshgrid(x,t);
[C,h] = contour(xi,u,ti);
set(h,'ShowText','on','LineWidth',2,'TextStep',get(h,'LevelStep')*2)
grid on
ylabel('Stężenie [kmol/m3]')
xlabel('Długość [m]')
```

```

% -----
function [c,f,s] = pdex1pde(x,t,u,DuDx)
global E k ut
% zdefiniowanie równania
c = 1;
f = E*DuDx;
s = -ut*DuDx-k*u;
% -----
function u0 = pdex1ic(x)
global c0
% warunek początkowy
u0 = 0;
if x==0
    u0=c0;
end;
% -----
function [pl,ql,pr,qr] = pdex1bc(xl,ul,xr,ur,t)
global c0
% warunki brzegowe l-lewy, r-prawy
pl = c0-ul;
ql = 0;
pr = 0;
qr = 1;

```

Wyniki 8.7

Jako stężenie końcowe program wydrukował 0.0354 kmol/m^3 . Rozkłady przestrzenne i czasowe stężenia w reaktorze pokazują poniższe rysunki.

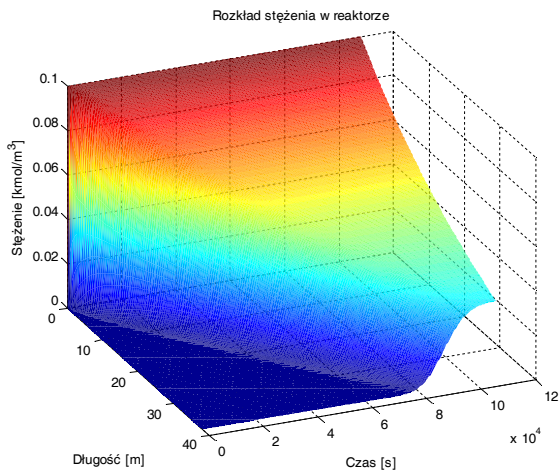
Uwaga: aby rozwiązanie było numerycznie stabilne musi być spełniony warunek:

$$\sigma^2 < \lambda < 1, \quad (8.79)$$

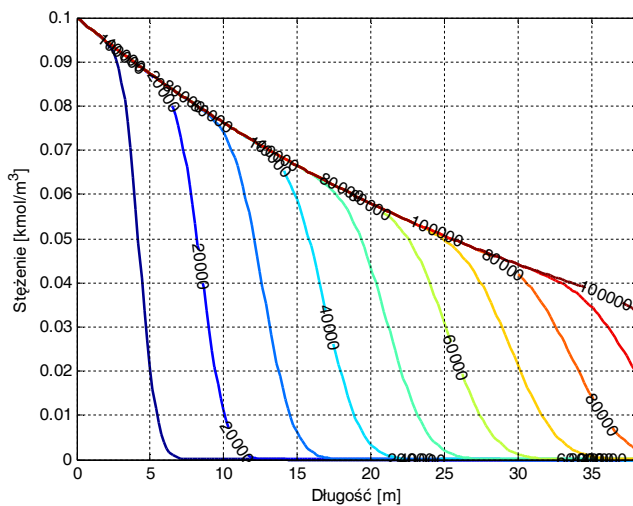
gdzie:

$$\sigma^2 = \left(u_t \frac{\Delta \tau}{\Delta x} \right)^2 \quad \lambda = 2E \frac{\Delta \tau}{(\Delta x)^2} \quad (8.80)$$

gdzie Δx to krok przestrzenny, a $\Delta \tau$ krok czasowy. W naszych warunkach dla liczby kroków przestrzennych 200 i czasowych 250 mamy $\sigma^2 = 0.921$ oraz $\lambda = 0.980$, co gwarantuje, że schemat jest stabilny.



Rys. 8.14. Wykres 3D rozkładów stężenia w reaktorze



Rys. 8.15. Wykres 2D rozkładów stężenia w reaktorze

8.5. Rozwiązywanie równań typu eliptycznego

Przedstawimy najpierw metodę macierzową, która w zależności od sposobu rozwiązywania powstałego układu równań liniowych bywa nazywana metodą odwracania macierzy lub metodą relaksacji.

Należy ona do metod różnic skończonych i polega na dyskretyzacji przestrzeni siatką prostopadłościenną. W środku otrzymanych elementów przestrzennych umieszcza się węzły skupiające wartość funkcji dla całej komórki siatki. Ponieważ w równaniu eliptycznym brak pochodnych po czasie równanie bilansowe dla komórki otrzymanej siatki staje się równaniem algebraicznym. Po napisaniu równań bilansowych dla wszystkich elementów uzyskuje się układ równań liniowych, który rozwiązuje się metodą odwracania macierzy. Metodę tę zilustrujemy na najprostszym przykładzie dwuwymiarowego przewodzenia ciepła.

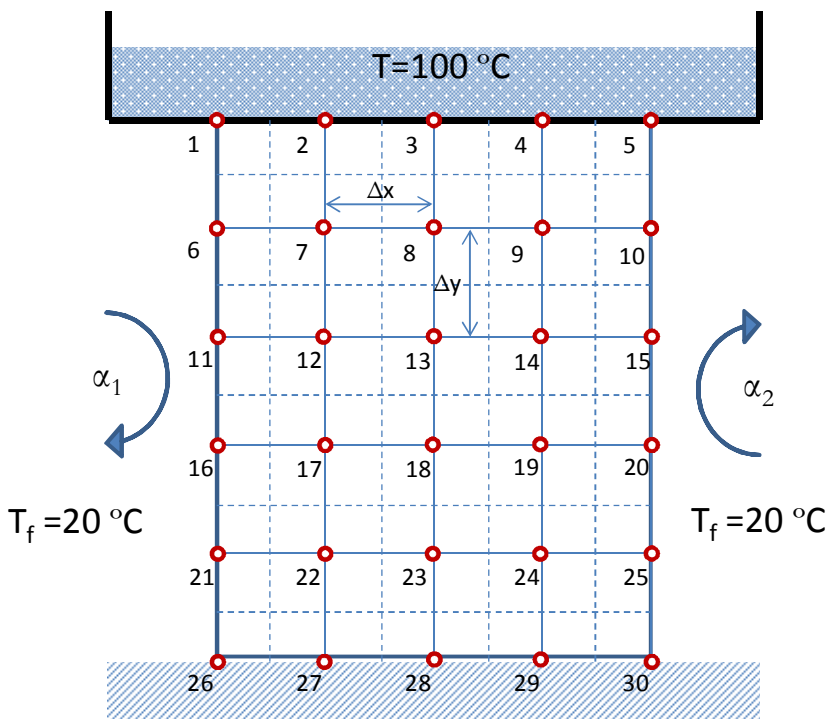
PROJEKT 8.8

Temat:	Ustalony ruch ciepła w podporze o przekroju prostokątnym
Toolbox:	-

Na Rys. 8.15 przedstawiono schemat podpory zbiornika z gorącą cieczą, posadowionej na izolowanym cieplnie podłożu. Ściana jest chłodzona z obu stron powietrzem o tej samej temperaturze, lecz współczynniki wnikania są różne po obu stronach. Dane zawarte są w programie.

Zasady dyskretyzacji tu użyte opisują *Incropera i in., 2007*. Na przekrój podpory naniesiono siatkę o kroku Δx w kierunku poziomym i Δy w kierunku pionowym. Na przecięciach linii siatki (linie ciągłe) umieszczono węzły ponumerowane od 1 do 30. Węzły te należą do odpowiednich komórek ograniczonych liniami przerywanymi o tym samym skoku Δx i Δy . Dla każdej z komórek należy ułożyć równanie bilansowe ciepła dopływającego do komórki. Dla ułatwienia pracy zauważmy, że występuje tu kilka rodzajów komórek np. komórki 1, 2, 3, 4 i 5 leżą na powierzchni o stałej temperaturze i sąsiadują na pozostałych ścianach z innymi komórkami. Należy więc wyprowadzić odpowiednie równania dla każdej grupy osobno. Zastosujemy następujące indeksowanie: i – oznacza daną komórkę, a indeksy l , p , g , d – odpowiednio komórki z lewej, prawej, u góry i u dołu. Dla uproszczenia przyjmiemy, że podpora jest izotropowa i ma tę samą przewodność cieplną w obu kierunkach.

Aby móc obliczyć powierzchnię wymiany ciepła między komórkami przyjmiemy, że podpora ma długość jednostkową.



Rys. 8.16. Schemat dyskretyzacji powierzchni przekroju ściany przewodzącej ciepło

- Komórka powierzchniowa o stałej temperaturze ($i = 1$ do 5)

$$T_i = \text{const} . \quad (8.81)$$

- Komórka powierzchniowa wymieniająca ciepło przez wnikanie ($i = 6, 10, 11, 15, 16, 20, 21, 25$)

$$\lambda \frac{T_g - T_i}{\Delta y} \frac{\Delta x}{2} + \lambda \frac{T_d - T_i}{\Delta y} \frac{\Delta x}{2} + \lambda \frac{T_{p/l} - T_i}{\Delta x} \Delta y + \alpha (T_f - T_i) \Delta y = 0 . \quad (8.82)$$

Indeks p/l oznacza komórkę z lewej bądź z prawej.

- Komórka wewnętrzna ($i = 7, 8, 9, 12, 13, 14, 17, 18, 19, 22, 23, 24$)

$$\lambda \frac{T_l - T_i}{\Delta x} \Delta y + \lambda \frac{T_p - T_i}{\Delta x} \Delta y + \lambda \frac{T_g - T_i}{\Delta y} \Delta x + \lambda \frac{T_d - T_i}{\Delta y} \Delta x = 0 . \quad (8.83)$$

- Komórka powierzchniowa izolowana ($i = 27, 28, 29$)

$$\lambda \frac{T_l - T_i}{\Delta x} \frac{\Delta y}{2} + \lambda \frac{T_p - T_i}{\Delta x} \frac{\Delta y}{2} + \lambda \frac{T_l - T_i}{\Delta y} \Delta x = 0. \quad (8.84)$$

- Komórka narożna z jednej strony wymieniająca ciepło przez wnikanie, z drugiej izolowana ($i = 26, 30$)

$$\lambda \frac{T_g - T_i}{\Delta y} \frac{\Delta x}{2} + \lambda \frac{T_{l/p} - T_i}{\Delta x} \frac{\Delta y}{2} + \alpha(T_f - T_i) \frac{\Delta y}{2} = 0. \quad (8.85)$$

W tym miejscu należy dodać, że podobne równania łatwo ułożyć dla innych rodzajów węzłów jak np. węzeł w narożniku wklęsłym czy węzeł narożny izolowany na dwu powierzchniach bocznych. Można też uwzględnić objętościową generację ciepła.

Łatwo zauważyć, że równania te są liniowe ze względu na T . Można je więc przekształcić do postaci macierzowej

$$A \cdot T = b, \quad (8.86)$$

gdzie A – macierz współczynników a b – macierz wyrazów wolnych. Przykładowo równanie dla węzła 26 ma po przekształceniu postać:

$$(\Delta x)^2 T_{21} + \left(-(\Delta x)^2 - (\Delta y)^2 - \frac{\alpha_1 (\Delta y)^2 \Delta x}{\lambda} \right) T_{26} + (\Delta y)^2 T_{27} = -\frac{\alpha_1 (\Delta y)^2 \Delta x}{\lambda} T_f. \quad (8.87)$$

Dla wszystkich węzłów w powyższy sposób zbudujemy kolejne wiersze macierzy A i wektora b .

Na koniec wystarczy obliczyć wektor T przez przekształcenie

$$T = A^{-1}b. \quad (8.88)$$

Obliczenia te zawarte są w programie poniżej. Budowę macierzy wykonano w znacznej części ręcznie. Zbudowanie automatycznego generatora równań na podstawie osiatkowania jest pracochłonne i zwykle nieopłacalne dla jednostkowych przypadków. Zwykle w takich przypadkach korzystać będziemy z komercyjnego oprogramowania.

P8_8.m

```
%PROJEKT 8.8
%Rozwiązanie eliptycznego równania przewodzenia ciepła
function heat_elip
clear all; clc;
%Dane
H=0.1;           %wysokość podpory, m
B=0.08;          %grubość podpory, m
lam=2;           %przewodność cieplna podpory, W/(m K)
Tf=20;           %temperatura otoczenia, C
T0=100;          %temperatura górnej krawędzi, C
alfa1=120;       %współczynnik wnikania ciepła po lewej, W/(m2 K)
alfa2=50;        %współczynnik wnikania ciepła po prawej, W/(m2 K)
N=30;            %liczba węzłów
nk=5;            %liczba kolumn węzłów
nw=6;            %liczba wierszy węzłów

Dx=B/(nk-1);     %krok siatki w kierunku x
Dy=H/(nw-1);     %krok siatki w kierunku y
Dx2=Dx^2;
Dy2=Dy^2;

%Macierz współczynników dla wszystkich węzłów, łącznie
%z tymi, dla których znamy temperaturę.

A=zeros(N);
b= linspace(0,0,N)';

%Wszystkie węzły w pierwszym wierszu
A(1,1)=1; A(2,2)=1; A(3,3)=1; A(4,4)=1; A(5,5)=1;
b(1)=T0; b(2)=T0; b(3)=T0; b(4)=T0; b(5)=T0;

%Węzły środkowe w pierwszej kolumnie
c1=-2*(Dx2+Dy2)-2*Dx*Dy2*alfa1/lam;
z1=-2*Dx*Dy2*alfa1*Tf/lam;
A( 6, 1)=Dx2; A( 6, 6)=c1; A( 6,11)=Dx2; A( 6, 7)=2*Dy2;
b(6)=z1;
A(11, 6)=Dx2; A(11,11)=c1; A(11,16)=Dx2; A(11,12)=2*Dy2;
b(11)=z1;
A(16,11)=Dx2; A(16,16)=c1; A(16,21)=Dx2; A(16,17)=2*Dy2;
b(16)=z1;
A(21,16)=Dx2; A(21,21)=c1; A(21,27)=Dx2; A(21,22)=2*Dy2;
b(21)=z1;
```

```

%Węzły środkowe w ostatniej kolumnie
c2=-2*(Dx2+Dy2)-2*Dx*Dy2*alfa2/lam;
z2=-2*Dx*Dy2*alfa2*Tf/lam;
A(10,5)=Dx2; A(10,10)=c2; A(10,15)=Dx2; A(10, 9)=2*Dy2;
b(10)=z2;
A(15,5)=Dx2; A(15,15)=c2; A(15,20)=Dx2; A(15,14)=2*Dy2;
b(15)=z2;
A(20,5)=Dx2; A(20,20)=c2; A(20,25)=Dx2; A(20,19)=2*Dy2;
b(20)=z2;
A(25,5)=Dx2; A(25,25)=c2; A(25,30)=Dx2; A(25,24)=2*Dy2;
b(25)=z2;

%Dolny lewy węzeł narożny
c5=-Dx2-Dy2-Dx*Dy2*alfa1/lam;
A(26,21)=Dx2; A(26,26)=c5; A(26,27)=Dy2;
b(26)=-Dx*Dy2*alfa1*Tf/lam;

%Dolny prawy węzeł narożny
c6=-Dx2-Dy2-Dx*Dy2*alfa2/lam;
A(30,25)=Dx2; A(30,30)=c6; A(30,29)=Dy2;
b(30)=-Dx*Dy2*alfa2*Tf/lam;

%Dolne węzły środkowe
c3=-2*(Dx2+Dy2);
A(27,22)=2*Dx2; A(27,27)=c3; A(27,28)=Dy2;
b(27)=0;
A(28,23)=2*Dx2; A(28,28)=c3; A(28,29)=Dy2;
b(28)=0;
A(29,24)=2*Dx2; A(29,29)=c3; A(29,30)=Dy2;
b(29)=0;

%Wszystkie węzły wewnętrzne
c4=-2*(Dx2+Dy2);
ri=[7 8 9 12 13 14 17 18 19 22 23 24]; %tablica numerów węzłów
wewn.
for k=1:12
    j=ri(k);
    A(j,j-nk)=Dx2; A(j,j-1)=Dy2; A(j,j)=c4; A(j,j+1)=Dy2;
    A(j,j+nk)=Dx2;
    b(j)=0;
end;
%Rozwiązanie
T=inv(A)*b

```



```

%Konwersja wektora T na dwuwymiarową macierz TT
TT(1,1:5)=[T( 1) T( 2) T( 3) T( 4) T( 5)];
TT(2,1:5)=[T( 6) T( 7) T( 8) T( 9) T(10)];
TT(3,1:5)=[T(11) T(12) T(13) T(14) T(15)];
TT(4,1:5)=[T(16) T(17) T(18) T(19) T(20)];
TT(5,1:5)=[T(21) T(22) T(23) T(24) T(25)];
TT(6,1:5)=[T(26) T(27) T(28) T(29) T(30)];

%Wydrukuj macierz temperatur
disp(TT);

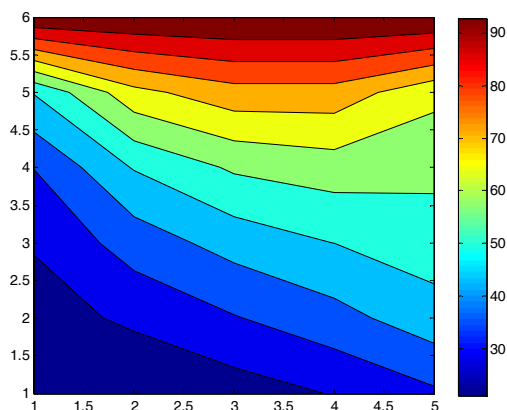
%Wykres warstwiczny
TT=flipud(TT); %obróć macierz wokół osi poziomej
figure(1);
contourf(TT,10);

```

Wyniki

100.0000	100.0000	100.0000	100.0000	100.0000
50.2660	69.0508	75.6360	75.5423	65.9918
35.6005	50.3012	57.9508	60.5414	58.8743
28.9749	38.6028	45.3246	49.7983	53.2889
24.6334	29.8103	34.9466	40.0381	46.8479
21.7787	21.0585	24.6132	28.5598	34.1631

Wyniki 8.8



Rys. 8.17. Wykres warstwiczny rozkładu temperatury w podporze

W tym miejscu należy przestrzec Czytelnika przed bezkrytycznym przyjmowaniem wyników tych obliczeń. Jakość rozwiązania zależy od gęstości osiatkowania. W powyższym przypadku wystarczy zwiększyć rozmiary podpory, aby uzyskać w niektórych węzłach temperaturę niższą niż temperatura otoczenia! Aby upewnić się, że rozwiązanie jest poprawne, należy wykonać obliczenie sprawdzające przy dwukrotnie zagęszczonej siatce w obu kierunkach. Jeśli otrzymane wyniki są takie same, mówimy, że rozwiązanie jest *niezależne od siatki*. Przy ręcznej generacji równań (jak wyżej) jest to oczywiście bardzo pracochłonne.

Inną metodą rozwiązano równanie eliptyczne ustalonego przepływu płynu ruchem laminarnym w kanale prostokątnym. Przedstawia to Projekt 8.9. Metoda ta jest mniej pracochłonna jak chodzi o dyskretyzację.

PROJEKT 8.9

Temat:	Ustalony profil prędkości w laminarnym przepływie przez kanał o przekroju prostokątnym
Toolbox:	-

W rurociągu o przekroju prostokątnym płynie ciecz ustalonym ruchem laminarnym. Należy wyznaczyć dwuwymiarowy profil prędkości cieczy w tym kanale. Problem opisany jest równaniem Naviera-Stokesa, które w tym przypadku upraszcza się do postaci

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = \frac{1}{\mu} \frac{dp}{dz}. \quad (8.89)$$

Podstawiając następujące simpleksy

$$X = x/B, \quad Y = y/H, \quad U = u/u_c, \quad (8.90)$$

gdzie u_c jest prędkością w osi przewodu, otrzymujemy równanie bezwymiarowe

$$\frac{\partial^2 U}{\partial X^2} + \frac{\partial^2 U}{\partial Y^2} = \frac{H^2}{u_c \mu} \frac{dp}{dz}. \quad (8.91)$$

Domena obliczeniowa pokazana na Rys. 8.18 obejmuje jedynie $\frac{1}{4}$ przekroju kanału z uwagi na symetrię, stąd warunki brzegowe dla tego równania są następujące:

Na ścianach kanału	$X = 0.5H/B$	$U = 0$
	$Y = 0.5$	$U = 0$
Na osiach symetrii	$X = 0$	$\frac{\partial U}{\partial X} = 0$
	$Y = 0$	$\frac{\partial U}{\partial Y} = 0$
W osi kanału	$X = 0$ i $Y = 0$	$U = 1$

Zaprezentowany tu sposób rozwiązywania równania (8.89) zaproponowali *Osthuizen i Naylor, 1999*. Polega on na następujących krokach.

W pierwszej kolejności rozwiązujemy równanie, w którym człon źródłowy (po prawej) jest równy -1. W tym celu dyskretyzujemy obszar siatką prostokątną o krokach ΔX i ΔY . Równanie przedstawiamy w postaci dyskretnej, stosując trójpunktowy schemat centralny do przedstawienia pochodnych drugiego rzędu

$$\frac{U_{i+1,j} - 2U_{i,j} + U_{i-1,j}}{\Delta X^2} + \frac{U_{i,j+1} - 2U_{i,j} + U_{i,j-1}}{\Delta Y^2} = -1, \quad (8.92)$$

stąd po przekształceniu mamy wyrażenie na $U_{i,j}$

$$U_{i,j} = \frac{\frac{U_{i+1,j} + U_{i-1,j}}{\Delta X^2} + \frac{U_{i,j+1} + U_{i,j-1}}{\Delta Y^2} + 1}{\frac{2}{\Delta X^2} + \frac{2}{\Delta Y^2}}. \quad (8.93)$$

Wypełniamy teraz cały obszar, poczynając od ścian gdzie za U podstawiamy zero, przez wszystkie punkty wewnętrzne (z wyjątkiem osi symetrii podstawiając jakieś założenie początkowe np.

$$U(X, Y) = (1 - X^2)(1 - Y^2). \quad (8.94)$$

W komórkach leżących na osiach symetrii wpisujemy wartości w sąsiednich komórkach

$$U_{i,1} = U_{i,2} \text{ oraz } U_{1,j} = U_{2,j} \quad (8.95)$$

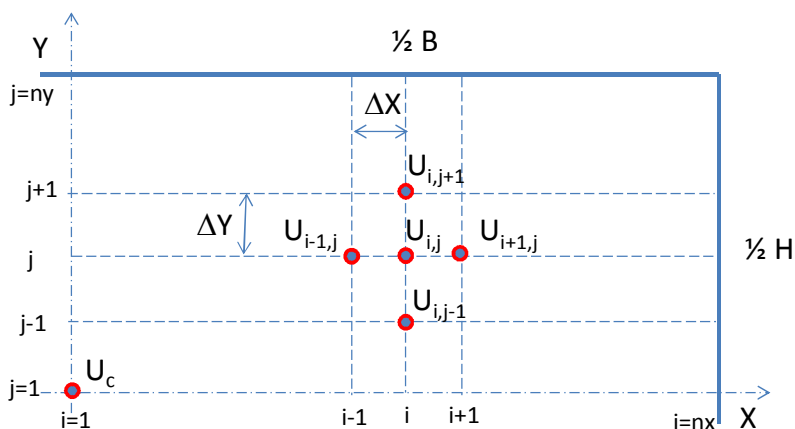
Dokonujemy kolejnych iteracji, przeliczając wszystkie komórki wewnętrzne, dopóki w wybranej komórce zmiana wartości w dwu kolejnych przeliczeniach nie będzie mniejsza niż zadana dokładność ϵ .

W wyniku tych obliczeń uzyskujemy w osi kanału wartość U_c różną niż zero, gdyż rozwiązywane równanie to różni się od właściwego członem źródłowym.

Odczytujemy wartość obliczoną U_c . Ostateczne rozwiązanie musi dawać $U = 1$ w osi przewodu (trzeci warunek brzegowy), stąd musi zachodzić zależność

$$\frac{H^2}{u_c \mu} \frac{dp}{dz} = \frac{1}{U_c}. \quad (8.96)$$

Na podstawie tego równania można policzyć wartość u_c w osi na podstawie danych wartości dp/dz , H , μ oraz odczytanej wartości U_c . Ostateczne rozwiązanie wymiarowe otrzymamy po przekształceniu wszystkich zmiennych na wymiarowe.



Rys. 8.18. Schemat dyskretyzacji $\frac{1}{4}$ powierzchni przekroju kanału

Obliczenia te zawarte są w programie poniżej. Dane są zawarte w programie.

P8_9.m

```
%PROJEKT 8.9
```

```
%Profil ustalony prędkości w kanale prostokątnym w ruchu laminarnym
```

```
clear all; close all; clc;
```

```
%Równanie rownania bezwymiarowego
```

```
nx=20; dx=1./nx;
```

```
ny=20; dy=1./ny;
```

```
U=zeros(nx,ny);
```

```
%BC na ścianie
```

```
U(ny,:)=0;
```

```
U(:,nx)=0;
```

```

%Pierwsze przybliżenie
for i=2:ny-1
    for j=2:nx-1
        X=(j-1)*dx;
        Y=(i-1)*dy;
        U(i,j)=(1-X.^2)*(1-Y.^2);
    end
end
%BC na płaszczyznach symetrii
U(1,:)=U(2,:);
U(:,1)=U(:,2);
%rysunek kontrolny
figure(1)
surf(U)
%Iteracyjne uzyskanie rozwiązania
dx2=dx.^2; dy2=dy.^2; %zmiennne pomocnicze
den=2*(1./dx2+1./dy2);
U0=10; w=1; nit=0; %wartości początkowe
while (w>1.e-3) && nit<1000
    for i=ny-1:-1:2
        for j=nx-1:-1:2
            U(i,j)=((U(i,j+1)+U(i,j-1))/dx2+(U(i+1,j)+U(i-1,j))/dy2+1)/den;
        end
    end
    end
    U(1,:)=U(2,:);
    U(:,1)=U(:,2);
    w=abs(U(2,2)-U0);
    U0=U(2,2);
    nit=nit+1;
end
hold on
surf(U) %rozwiązanie
title('Wykres bezwymiarowy: założenie i rozwiązanie')
Uc=U(1,1); %określenie Uc
%Wykres bezwymiarowy
U=U./Uc;
figure(2)
surfc(U)
title('Wykres bezwymiarowy znormalizowany')
%Wykres wymiarowy
B=0.05; %X Szerokość [m]
H=0.02; %Y Wysokość [m]

```

```

miu=1.0e-3; %lepkość [Pa s]
ro=1000.0; %gęstość [kg/m3]
dpdz=-0.02; %gradient ciśnienia [Pa/m]
uc=-Uc*H.^2*dpdz/miu
Rec=uc*ro*H/miu
u=U*uc*1000;
x=linspace(0,B/2,nx);
y=linspace(0,H/2,ny);
figure(3)
surf(x,y,u)
xlabel('Szerokość [m]')
ylabel('Wysokość [m]')
zlabel('Prędkość [mm/s]')
title('Profil prędkości w kanale prostokątnym')

```

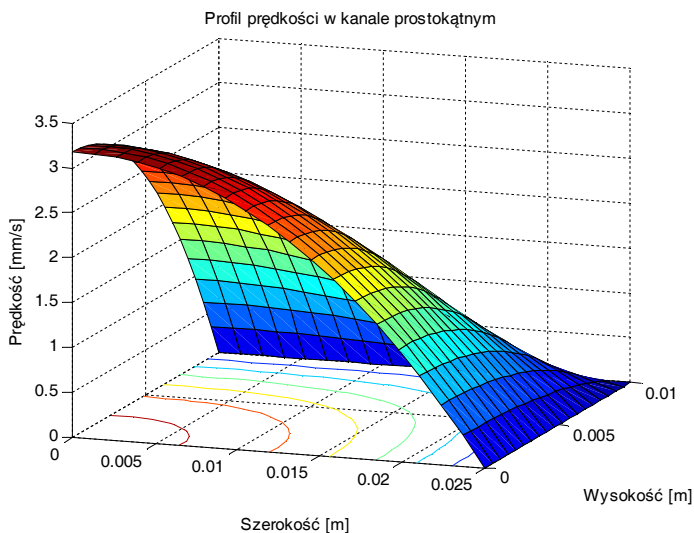
Wyniki 8.9

Program wydrukował następującą prędkość [m/s] w osi kanału i liczbę Reynoldsa

```

uc =    0.0032
Rec =   63.4631

```



Rys. 8.19. Wykres profilu prędkości w kanale

8.6. Problemy obliczeń numerycznych przy rozwiązywaniu PDE

Pierwszym ważnym problemem w rozwiązywaniu numerycznym PDE jest problem zbieżności. Powstaje zawsze pytanie czy rozwiązanie numeryczne odpowiada rozwiązaniu ścisłemu, tj. czy rozwiązanie numeryczne jest zbieżne, co ma miejsce wtedy, gdy dąży ono do rozwiązania ścisłego, kiedy krok całkowania dąży do zera. Rozwiązaniem ścisłym jest rozwiązanie analityczne, to zaś można uzyskać zwykle jedynie dla prostych przypadków. I tak np. istnieją analityczne rozwiązania równania dyfuzji nieustalonej w jednowymiarowej płycie, walcu i kuli (zob. *Crank, 1956*), lecz uzyskane dla stałej wartości dyfuzyjności i stałych warunków brzegowych.

Drugim podstawowym problemem jest czy dany schemat różnicowy (sposób reprezentacji pochodnych ilorazem różnicowym) będzie stabilny przy danej gęstości siatki dyskretizacyjnej. Jest to problem stabilności. Łatwo zaobserwować, również na przykładach z tej książki, że przy zmianie kroku całkowania uzyskuje się czasem oscylacje uzyskanych funkcji lub wręcz przekroczenie zakresu liczbowego.

Prostą metodą sprawdzenia zbieżności rozwiązania numerycznego jest porównanie go z rozwiązaniem teoretycznym dla przypadku stałych współczynników. Można wtedy sądzić, że będzie on zbieżny również dla przypadku zmiennych współczynników. Jeśli rozwiązania analitycznego brak, pozostaje zmniejszanie kroku całkowania do chwili, gdy rozwiązanie przestanie w sposób istotny zależeć od kroku całkowania.

Podstawowym sposobem na zmniejszenie niestabilności schematu różnicowego jest rozwiązywanie równania w postaci znormalizowanej, tj. przekształconego z użyciem zmiennych bezwymiarowych. Zmienne w równaniach różnią się zakresami wielkości, które zresztą zależą od jednostek. Różnice te sięgają kilku rzędów wielkości. W formie bezwymiarowej większość zmiennych zmienia się w zakresie $\langle 0, 1 \rangle$, co zmniejsza oscylacje. Upraszczają się również postaci współczynników równań kanonicznych solvera **pdepe**.

Stosując następujące zmienne bezwymiarowe

$$\Phi = \frac{T - T_f}{T_0 - T_f}, \quad Fo = \frac{a\tau}{L^2} \quad \text{ i } \quad \chi = \frac{x}{L}, \quad (8.97)$$

gdzie Fo to liczba Fouriera, możemy przekształcić równanie (8-30) do postaci bezwymiarowej:

$$\frac{\partial \Phi}{\partial Fo} = \frac{\partial^2 \Phi}{\partial \chi^2}. \quad (8.98)$$

Również do bezwymiarowej postaci przekształcimy warunki brzegowe (8.11) i (8.13). Otrzymamy odpowiednio:

$$\left. \frac{\partial \Phi}{\partial \chi} \right|_{\chi=0} = 0 \quad (8.99)$$

oraz

$$-\left. \frac{\partial \Phi}{\partial \chi} \right|_{\chi=1} = Bi_H \Phi, \quad (8.100)$$

gdzie Bi_H to liczba Biota dla ruchu ciepła dana wzorem

$$Bi_H = \frac{\alpha L}{\lambda}. \quad (8.101)$$

Dla schematu różnicowego (8.31), gdzie dodatkowo zdyskretyzujemy pochodną po czasie, dla formy bezwymiarowej otrzymamy:

$$\frac{\Phi^{(j+1)} - \Phi^{(j)}}{\Delta Fo} = \frac{\Phi_{i+1} - 2\Phi_i + \Phi_{i-1}}{(\Delta \chi)^2}. \quad (8.102)$$

Można udowodnić, że stabilność numeryczna jest zapewniona, gdy

$$\Delta Fo > \frac{(\Delta \chi)^2}{2}. \quad (8.103)$$

Oczywiście w metodzie linii stosujemy bardziej złożoną reprezentację pochodnej po czasie, którą dostarcza użyty solver ODE i warunek powyższy należy traktować z przybliżeniem.

Rozwiązanie problemu stygnięcia płyty jednowymiarowej dla równania sprawdzonego do postaci bezwymiarowej przedstawia Projekt 8.10.

PROJEKT 8.10

Temat: Nieustalony ruch ciepła w płycie nieskończonej. Równanie w wersji znormalizowanej

Toolbox: -

Scenariusz i dane identyczne jak w Projekcie 8.3. Rozwiązano równanie przewodzenia w wersji znormalizowanej.

P8_10.m

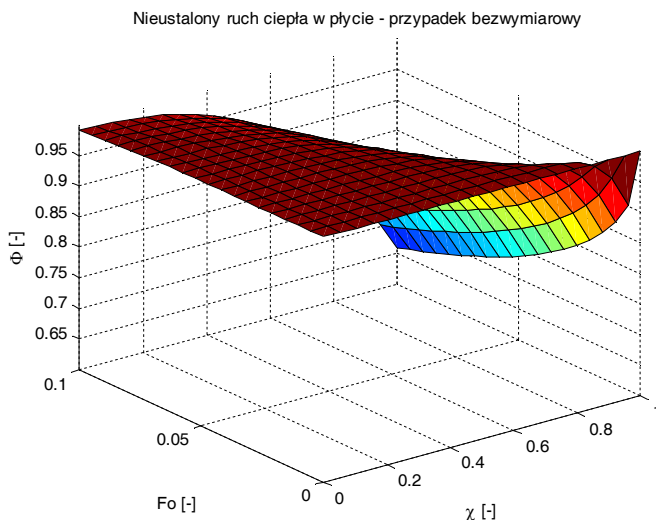
```
%PROJEKT 8.10
%Znormalizowane równanie przewodzenia ciepła w płycie
function plyta_dim
clear all; close all; clc
global Bi
%-----dane
t0=20; tg=100;
lam=0.00375;
a=1e-6;
L=0.1;
alfa=0.05;
Bi=alfa*L/lam;
%---rozwiązanie bezwymiarowe
m = 0; %płyta
x = linspace(0,1,20); % grubość bezwymiarowa, -
Fo = linspace(0,0.1,20); % czas bezwymiarowy, -
% wywołanie solvera PDE
sol = pdepe(m,@pdex1pde,@pdex1ic,@pdex1bc,x,Fo);
u = sol(:,:,1);
% ----przeliczenie na wymiarowe
T=u.*(t0-tg)+tg;
t=Fo.*L^2/a;
l=x*L;
% ---- prezentacja wyników
figure(1)
surf(x,Fo,u)
title('Nieustalony ruch ciepła w płycie - przypadek bezwymiarowy')
xlabel('\chi [-]')
ylabel('Fo [-]')
zlabel('\Phi [-]')
figure(2)
surf(l,t,T)
```

```

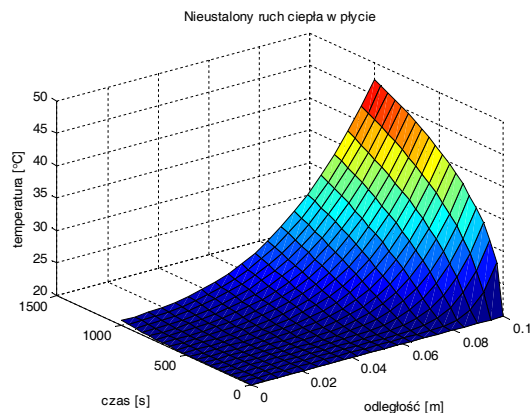
title('Nieustalony ruch ciepła w płycie')
xlabel('odległość [m]')
ylabel('czas [s]')
zlabel('temperatura [\text{C}]')
% -----
function [c,f,s] = pdex1pde(x,t,u,DuDx)
% zdefiniowanie równania
c = 1;
f = DuDx;
s = 0;
% -----
function u0 = pdex1ic(x)
% warunek początkowy
u0 = 1;
% -----
function [pl,ql,pr,qr] = pdex1bc(xl,ul,xr,ur,t)
% warunki brzegowe l-lewy, p-prawy
global Bi
pl = 0;
ql = 1;
pr = Bi*ur;
qr = 1;

```

Wyniki 8.10



Rys. 8.20. Rozkład temperatur w płycie we współrzędnych bezwymiarowych



Rys. 8.21. Rozkład temperatur w płycie we współrzędnych wymiarowych

8.7. Przykłady obliczeniowe

Projekty tu przedstawione ilustrują różne aspekty obliczeń opartych na rozwiązaniu równań PDE typu parabolicznego w zastosowaniach dotyczących inżynierii procesowej. Zawierają one następujące zagadnienia:

- Projekt 8.11 przedstawia problem nieustalonego ruchu ciepła w ścianie płaskiej z warunkiem początkowym określonym równaniem. Zawiera również obsługę wydarzeń.
- Projekt 8.12 przedstawia problem nieustalonego ruchu ciepła w ścianie płaskiej z warunkiem początkowym określonym równaniem i zmiennym w czasie warunkiem brzegowym na brzegu.
- Projekt 8.13 przedstawia problem nieustalonego ruchu ciepła w walcu wielowarstwowym, z różnymi właściwościami każdej z warstw.
- Projekt 8.14 przedstawia problem suszenia kuli opisanego układem równań jednoczesnego ruchu ciepła i masy.
- Projekt 8.15 przedstawia problem suszenia warstwy materiału ziarnistego z oporem wewnętrznym ruchu ciepła i masy w każdym ziarnie materiału.
- Projekt 8.16 przedstawia problem suszenia parą przegrzaną próbki materiału w kształcie walca.

Warto też zajrzeć do Projektu 9.5, który zawiera jest rozwiązanie problemu suszenia walca z ruchomą granicą – walec kurczy się w czasie procesu.

PROJEKT 8.11

Temat: Nieustalony ruch ciepła w ścianie budynku. Dochodzenie do stanu ustalonego po zmianie warunku brzegowego

Toolbox: -

Wskutek strat ciepła od ogrzewanego wnętrza budynku do otoczenia temperatura zewnętrznej i wewnętrznej powierzchni jednej ze ścian budynku ustaliła się na poziomie odpowiednio 10 i 21°C. Gdy temperatura otoczenia spadła nagle do 0°C podniesiono temperaturę wewnątrz budynku do 22°C. Określić, po jakim czasie temperatura wewnętrznej powierzchni ściany ustali się i jaką wartość osiągnie (*Kudra, 1985 ss. 324-327*).

Posłużono się tu obsługą wydarzeń (*events*- por. Projekt 7.7) w celu określenia czasu, po którym osiągnięty zostaje stan ustalony w ścianie, czyli stanu gdy profil temperatury zbliżony jest do linii prostej. W funkcji *zdarzenie* określono warunek, kiedy profil temperatury w ścianie jest już zbliżony do prostej, obliczając pochodne dT/dx we wszystkich punktach siatki przestrzennej. Następnie obliczono odchylenie standardowe s tych pochodnych i przyjęto, że gdy osiągnie ono wartość 2 (*value= s-2*), podczas gdy ta różnica maleje (*dircn = -1*), to profil temperatury jest już wystarczająco płaski i obliczenia należy przerwać (*isterminal=1*). Ponieważ zaczynaliśmy od płaskiego profilu temperatury dlatego na obliczenia zezwolono dopiero gdy $t > \Delta t$.

sciana1.m

```
%PROJEKT 8.11
%Ogrzewanie ściany budynku do stanu ustalonego
function sciana1
clear all; close all; clc;
global cp ro lambda alfa_z alfa_w L Tw1 Tz1 Tws Tzs n dt
% płyta m=0; walec m=1; kula m=2
m=0;
n=50;
%-----dane
cp=800;           % ciepło właściwe ściany [J/(kg K)]
ro=1600;          % gęstość ściany [kg/m3]
lambda=0.5;       % przewodność ściany [W/(m K)]
alfa_z=20;        % zewn. wsp. wnikania ciepła [W/(m2 K)]
alfa_w=5;         % wewn. wsp. wnikania ciepła [W/(m2 K)]
Tw1=22;           % nowa temp. wewn. [C]
Tz1=0;            % nowa temp. zewn. [C]
Tzs=10;           % pocz. temp. zewn. ściany [C]
Tws=21;           % pocz. temp. zewn. ściany [C]
```

```

L=0.5;           % grubość ściany [m]
t=80;           % czas procesu [godz.]
dt=t*3600./n;
x = linspace(0,L,n); % grubość [m]
t = linspace(0,t*3600,n); % czas [s]
% wywołanie solvera PDE
options=odeset('Events',@zdarzenie);
[sol,tsol,sole,te,ie] = pde-
pe(m,@pdex1pde,@pdex1ic,@pdex1bc,x,t,options);
u = sol(:, :, 1);
disp('Temperatura ściany wewn. [C]'); disp(u(end,end))
disp('Czas ustalnia się profilu temp. [hr]'); disp(te/3600);
% -----prezentacja wyników
figure(1)
[w,k]=size(u);
surf(x,t(1:w)/3600,u)
title('Temperatura ściany')
xlabel('grubość ściany [m]')
ylabel('czas [hr]')
zlabel('temperatura [\circ C]')
figure(2)
tf=u(w,:);
plot(x,tf),grid on
title('Rozkład temperatur w nowym stanie ustalonym')
ylabel('T [\circ C]'), xlabel('L [m]')
% -----
function [c,f,s] = pdex1pde(x,t,u,DuDx)
global cp ro lambda
% zdefiniowanie równania
c = cp*ro;
f = lambda*DuDx;
s = 0;
% -----
function u0 = pdex1ic(x)
global Tws Tzs L
% warunek początkowy
u0 = (Tws-Tzs)*x/L+Tzs; %profil liniowy
% -----
function [pl,ql,pr,qr] = pdex1bc(xl,ul,xr,ur,t)
% warunki brzegowe l-lewy, p-prawy
global alfa_z alfa_w Tw1 Tz1
pl = -alfa_z*(ul-Tz1);
ql = 1;

```

```

pr = alfa_w*(ur-Tw1);
qr = 1;
function [value,isterminal,dircn]=zdarzenie(m,t,xmesh,umesh)
global n dt
if t>dt
    dl=xmesh(2)-xmesh(1);
    du=diff(umesh)./dl;
    du(n)=du(n-1);
    md=sum(du)./n;
    s=sum(abs(du-md)./n);
else
    s=1;
end
value=s-2;    %gdy ta wartość osiągnie zero
isterminal=1;%przerwij obliczenia
dircn=-1;    %jeśli wartość ta malała;

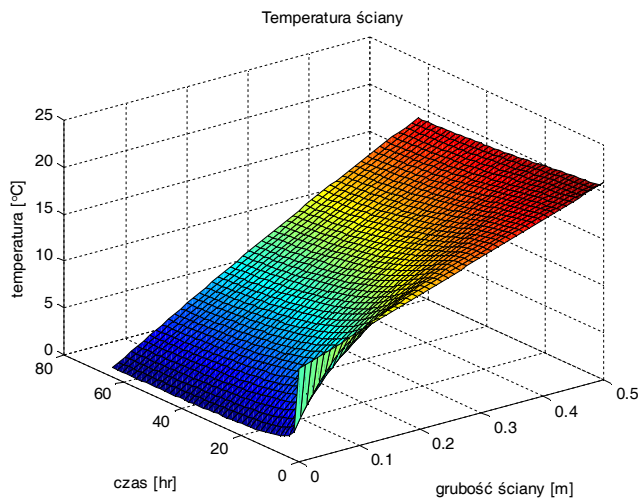
```

Wyniki 8.11

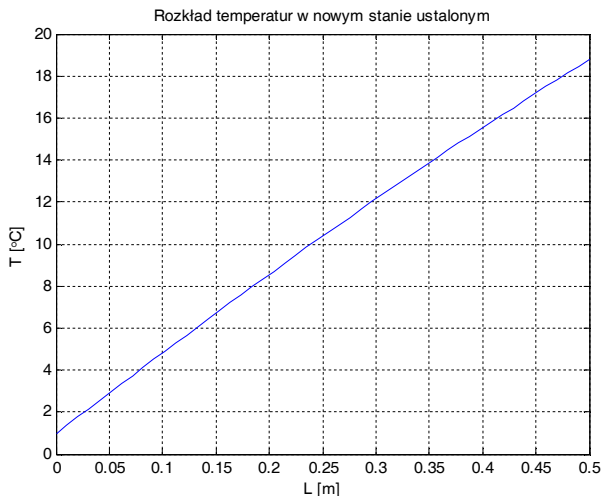
```

Temperatura ściany wewn. [C]
18.7995
Czas ustalnia się profilu temp. [hr]
62.2716

```



Rys. 8.22. Rozkład temperatur w ścianie



Rys. 8.23. Ustalony nowy rozkład temperatur

PROJEKT 8.12

Temat: Nieustalone nagrzewanie i chłodzenie ściany budynku.
Dowolny warunek początkowy. Zmienny warunek brzegowy
Toolbox: -

Wskutek strat ciepła do otoczenia temperatura zewnętrznej i wewnętrznej powierzchni jednej ze ścian budynku ustaliła się na poziomie odpowiednio 10 i 21°C. Od tej chwili temperatura na zewnątrz zmieniała się w cyklu dobowym, osiągając -5 stopni w nocy i 10 stopni w dzień. Określić wahania temperatury ściany wewnętrznej, jeśli powietrze w budynku miało stałe 22 stopnie.

Cykl dobowy temperatury opisano równaniem:

$$T(\tau) = T_0 + A \sin\left(2\pi \frac{\tau}{24}\right), \quad (8.104)$$

gdzie amplituda $A = (T_{\max} - T_{\min})/2$, a T_0 to punkt środkowy sinusoidy $T_0 = T_{\max} - A$ lub $T_{\min} + A$.

P8_2.m

```
%PROJEKT 8.12
%Nieustalone ogrzewanie ściany budynku - oscylujący warunek
brzegowy
function sciana2
clear all; close all;
global cp ro lambda alfa_z alfa_w L Tw1 Tws Tzs
% płyta m=0; walec m=1; kula m=2
m=0;
n=50;
%-----dane
cp=800;           % ciepło właściwe ściany [J/(kg K)]
ro=1600;          % gęstość ściany [kg/m3]
lambda=0.5;       % przewodność ściany [W/(m K)]
alfa_z=20;        % zewn. wsp. wnikań ciepła [W/(m2 K)]
alfa_w=5;         % wewn. wsp. wnikań ciepła [W/(m2 K)]
Tw1=22;           % temp. wewn. [C]
Tzs=10;           % pocz. temp. zewn. ściany [C]
Tws=21;           % pocz. temp. zewn. ściany [C]
L=0.5;            % grubość ściany [m]
t=48;             % czas procesu [godz.]
x = linspace(0,L,n); % grubość [m]
t = linspace(0,t*3600,n); % czas [s]
% wywołanie solvera PDE
sol = pdepe(m,@pdex1pde,@pdex1ic,@pdex1bc,x,t);
u = sol(:, :, 1);
disp('Temperatura ściany wewn. [C]'); disp(u(end,end))
% -----prezentacja wyników
figure(1)
surf(x,t/3600,u)
title('Temperatura ściany')
xlabel('grubość ściany [m]')
ylabel('czas [s]')
zlabel('temperatura [\text{C}]')
% -----
function [c,f,s] = pdex1pde(x,t,u,DuDx)
global cp ro lambda
% zdefiniowanie równania
c = cp*ro;
f = lambda*DuDx;
s = 0;
% -----
function u0 = pdex1ic(x)
```

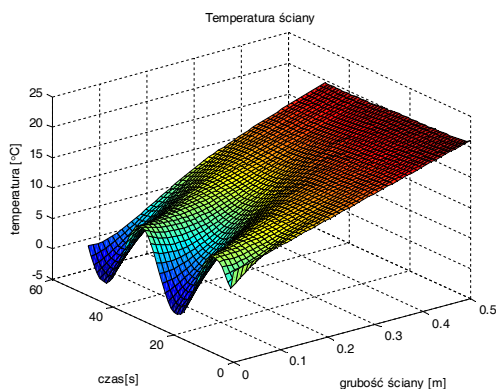


```

global Tws Tzs L
% warunek początkowy
u0 = (Tws-Tzs)*x/L+Tzs; %profil liniowy
% -----
function [pl,q1,pr,qr] = pdex1bc(xl,ul,xr,ur,t)
% warunki brzegowe l-lewy, p-prawy
global alfa_z alfa_w Tw1
Tz=3.5+(10-(-5))/2*sin(2*pi*t/(24*3600));%cykl zmian temp. zewn.
pl = -alfa_z*(ul-Tz1);
q1 = 1;
pr = alfa_w*(ur-Tw1);
qr = 1;

```

Wyniki 8.12



Rys. 8.24. Rozkład temperatur w ścianie w poszczególnych krokach czasowych

Jak widać wahania temperatury zewnętrznej praktycznie nie wpływają na temperaturę ściany wewnętrznej.

PROJEKT 8.13

Temat: Nieustalony ruch ciepła w walcu wielowarstwowym
 Toolbox: -

Trójwarstwowy walec o stałej temperaturze początkowej $T_0 = 20^\circ\text{C}$ został wystawiony na działanie temperatury $T_f = 800^\circ\text{C}$. Obliczyć rozkłady temperatur w tym walcu, jeśli współczynnik wnikania ciepła wynosi $\alpha = 1.25 \text{ W}/(\text{m}^2 \text{ K})$.

Właściwości warstw podane są poniżej:

Nr warstwy	Średnica [cm]	c_p [kJ/(kg K)]	ρ [kg/m ³]	a [m ² /s]
1	2	1.5	2500	1e-8
2	4	0.7	1800	1e-5
3	6	2.1	2600	1e-7

P8 13.m

%PROJEKT 8.13

%Nieustalone ogrzewanie walca wielowarstwowego

```
clear all; close all; clc
global n t0 R1 R2 R3 tg
% ----- podział siatki
n=50;          % liczba węzłów siatki dla każdej warstwy
t0=20;         % temperatura początkowa [C]
tk=4*3600;     % czas [s]
% ----- warunek początkowy
for i=1:3.*n
temp(i)=t0;
end
% wywołanie solvera ODE
[t,T]=ode15s(@uk_row,[0 tk],temp);
% ----- prezentacja wyników
xi=linspace(0,R3,3*n);
[w,k]=size(T);
yi=linspace(0,tk/3600,w);
[XI,YI]=meshgrid(xi,yi);

figure(1) %wykres 3D
surf(XI,YI,T)
title('Nieustalony ruch ciepła w walcu trójwarstwowym')
xlabel('odległość [m]')
ylabel('czas [godz]')
zlabel('temperatura [C]')
hidden on
shading interp
alpha 0.8
hold on
plot3([R2; R2],[0; tk/3600],[t0; t0],'k','LineWidth',2)
```

```

plot3([R1; R1],[0; tk/3600],[t0; t0],'k','LineWidth',2)
figure(2) %wykres 2D
[C,h] = contour(XI,T,YI,20);
set(h,'ShowText','on','LineWidth',2,'TextStep',get(h,'LevelStep')*1)
grid on
title('Nieustalony ruch ciepła w walcu trójwarstwowym')
xlabel('odległość [m]')
ylabel('temperatura [\textcircled{C}]')
hold on
plot([R1; R1],[0 tg],'m','LineWidth',2);
plot([R2; R2],[0 tg],'m','LineWidth',2);

```

Pozostałe dane zadeklarowano i zdefiniowano układ równań w pliku **uk_row.m**:

 **uk_row.m**

```

function dtdtemp=uk_row(t,temp)
% zdefiniowany układ równań 'metody linii'
global n R1 R2 R3 tg
% promienie [m]
R1=0.01; R2=0.02; R3=0.03;
% ciepła właściwe poszczególnych warstw [kJ/kg]
cp1=1.5; cp2=0.7; cp3=2.1;
% gęstości materiałów poszczególnych warstw [kg/m^3]
ro1=2500; ro2=1800; ro3=2670;
% dyfuzyjności cieplne poszczególnych warstw [m^2/s]
a1=1e-8; a2=1e-5; a3=1e-7;
% współczynnik wnikanía ciepła [kW/m^2*K]
alfa=1.25;
% temperatura otoczenia [C]
tg=800;
% wielkość komórki
dR1=R1/n; dR2=(R2-R1)/n; dR3=(R3-R2)/n;
% współczynnik przewodzenia ciepła poszczególnych warstw [W/m*K]
lamb1=a1*cp1*ro1; lamb2=a2*cp2*ro2; lamb3=a3*cp3*ro3;
% układ równań ODE
p(1)=a1.*((2.*temp(2)-2.*temp(1))./(dR1.^2));
for i=2:n-1
    p(i)=a1.*(((temp(i+1)-2*temp(i)+temp(i-1))./(dR1.^2)));
end
for i=n:2*n-1
    p(i)=a2.*(((temp(i+1)-2*temp(i)+temp(i-1))./(dR2.^2)));
end
for i=2*n:3*n-1

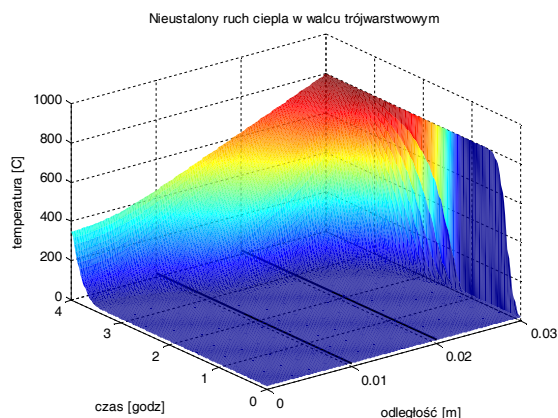
```

```

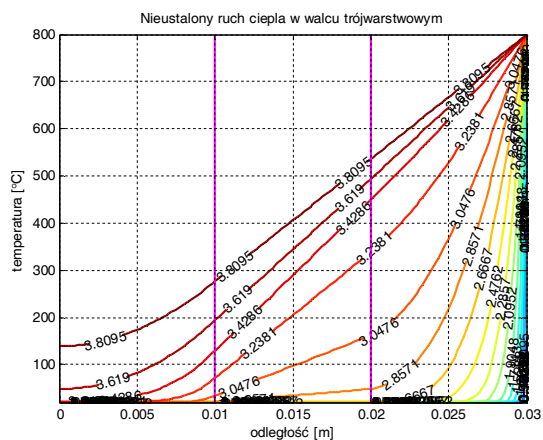
p(i)=a3.*(((temp(i+1)-2*temp(i)+temp(i-1))./(dR3.^2)));
end
wT=((2.*dR3.*alfa)./lamb3).*(temp(3.*n)-tg)./a3;
p(3*n)=a3.*((2.*temp(3.*n-1)-2.*temp(3.*n)-
wT)./(dR3.^2)+1/(n*dR3)*(-wT)/(2*dR3));
% wyjściowy wektor pochodnych
dtdtemp=[p'];

```

Wyniki 8.13



Rys. 8.25. Nieustalony rozkład temperatur w walcu trójwarstwowym. Wykres 3D



Rys. 8.26. Nieustalony rozkład temperatur w czasie w walcu trójwarstwowym
Wykres warstwowy

PROJEKT 8.14

Temat: Suszenie kuli – metoda linii

Plik główny: **suszenie_kuli.m**

Toolbox: -

Podstawą modelu procesu suszenia cząstki kulistej są równania bilansów różniczkowych dla ciepła i masy. Równania te mają postać równań PDE rzędu drugiego. Dodatkowo niezbędne są zależności opisujące warunki brzegowe, w tym równania kinetyczne na strumienie ciepła i masy wymieniane przez powierzchnię zewnętrzną, równania opisujące własności materiału oraz równowagę suszarniczą. (*Pakowski, 2000, Pakowski i in., 2011*)

Zakładając, że wilgoć w materiale przemieszcza się w stanie ciekłym w sposób dyfuzyjny układ równań opisujący jednoczesny dyfuzyjny ruch ciepła i masy w cząstce kulistej ma postać:

– bilans masy wilgoci

$$\frac{\partial X}{\partial \tau} = D \left(\frac{\partial^2 X}{\partial r^2} + \frac{2}{r} \frac{\partial X}{\partial r} \right), \quad (8.105)$$

– bilans ciepła

$$\frac{\partial T}{\partial \tau} = \frac{1}{\rho_m(c_L X + c_S)} \left[\lambda \left(\frac{\partial^2 T}{\partial r^2} + \frac{2}{r} \frac{\partial T}{\partial r} \right) - \frac{\partial X}{\partial \tau} c_L T \right] - \frac{c_L T}{c_L X + c_S} \frac{\partial X}{\partial \tau}. \quad (8.106)$$

Równania te są sprzężone i należy rozwiązywać je jednocześnie.

Należy teraz dokonać ich dyskretyzacji do postaci stosowanej w metodzie linii. Poniżej przedstawiono odpowiednie równania dla komórek w osi symetrii, komórek wewnętrznych i komórek powierzchniowych odpowiednio dla równań ruchu masy i ruchu ciepła. Liczba komórek oznaczona jest symbolem n .

- Równania dyskretyzacyjne dla ruchu masy

– komórka w osi symetrii

$$\frac{\partial X}{\partial \tau} = 2D \frac{X_1 - X_0}{(\Delta r)^2} \quad (8.107)$$

– komórka wewnętrzna

$$\frac{\partial X}{\partial \tau} = D_m \left(\frac{X_{i-1} - 2X_i + X_{i+1}}{(\Delta r)^2} + \frac{2}{i \Delta r} \frac{X_{i+1} - X_{i-1}}{2 \Delta r} \right) \quad (8.108)$$

– komórka powierzchniowa

$$\frac{\partial X}{\partial \tau} = 2D \left[\frac{X_{n-1} + \frac{\Delta r w_D}{\rho_m D} - X_n}{(\Delta r)^2} + \frac{1}{n(\Delta r)^2} \frac{\Delta r w_D}{\rho_m D} \right] \quad (8.109)$$

- Równania dyskretyzacyjne dla ruchu ciepła

– komórka w osi symetrii:

$$\frac{\partial T}{\partial \tau} = \frac{2}{\rho_m (c_L X + c_S)} \left[\lambda \frac{T_1 - T_0}{(\Delta r)^2} - D \frac{X_1 - X_0}{(\Delta r)^2} c_L T \right] - \frac{c_L T}{c_L X + c_S} D \frac{X_1 - X_0}{(\Delta r)^2} \quad (8.110)$$

– komórka wewnętrzna

$$\begin{aligned} \frac{\partial T}{\partial \tau} = A & \left[\lambda \left(\frac{T_{i-1} - 2T_i + T_{i+1}}{(\Delta r)^2} + \frac{2}{i} \frac{T_{i+1} - T_{i-1}}{(\Delta r)^2} \right) - \right. \\ & \left. D \left(\frac{X_{i-1} - 2X_i + X_{i+1}}{(\Delta r)^2} + \frac{1}{i} \frac{X_{i+1} - X_{i-1}}{(\Delta r)^2} \right) c_L T \right] + \\ & + B \left(\frac{X_{i-1} - 2X_i + X_{i+1}}{(\Delta r)^2} + \frac{2}{i} \frac{X_{i+1} - X_{i-1}}{(\Delta r)^2} \right) \end{aligned} \quad (8.111)$$

– komórka powierzchniowa

$$\frac{\partial T}{\partial \tau} = A \left[\lambda \left(\frac{2T_{n-1} + 2\frac{\Delta r}{\lambda} q - 2T_n}{(\Delta r)^2} - \frac{2}{n} \frac{\frac{\Delta r}{\lambda} q}{(\Delta r)^2} \right) - D \left(\frac{2X_{n-1} + 2\frac{\Delta r}{\rho_m D} w_D - 2X_n}{(\Delta r)^2} + \frac{2}{n} \frac{X_{n-1} + \frac{\Delta r}{\rho_m D} w_D}{(\Delta r)^2} \right) c_L T \right] + B \left(D \left(\frac{2X_{n-1} + 2\frac{\Delta r}{\rho_m D} w_D - 2X_n}{(\Delta r)^2} + \frac{2}{n} \frac{X_{n-1} + \frac{\Delta r}{\rho_m D} w_D}{(\Delta r)^2} \right) \right) \quad (8.112)$$

W powyższych równaniach

$$A = \frac{1}{\rho_m (c_L X_i + c_S)} \quad \text{i} \quad B = \frac{c_L T_i}{c_L X_i + c_S} \quad (8.113)$$

oraz

$$q = \alpha (T_n - T_f) \quad (8.114)$$

$$w_D = k_Y (Y_n^* - Y)$$

P8_14.m

```
%PROJEKT 8.14
%Nieustalone konwekcyjne suszenie kuli
clear all; close all; clc;
global n R dr Xcr Xeq tg P0 ts u Y
%-----dane
n=50; % liczba węzłów [szt]
R=0.01/2; % promień kulki [m]
dr=R/n; % szerokość komórki [m]
P0=101.325; % ciśnienie atmosferyczne [kPa]
```

```

b1=0.45; % wilgotność początkowa [%]
b2=0.05; % wilgotność końcowa [%]
tp=20; % początkowa temperatura ciała stałego [C]
X1=btoX(b1); % zawartość wilg. początkowa [kg/kg]
Xeq=btoX(b2); % zawartość wilg. końcowa [kg/kg]
Xcr=0.4; % wilgotność krytyczna [kg/kg]
tg=50; % temperatura czynnika suszącego [C]
czasp=0;
czask=3600; % czas suszenia kulki [s]
u=1.78; % prędkość przepływu czynnika suszącego [m/s]
Y=0.004; % wilgotność czynnika suszącego [kg/kg]

% ----- warunki początkowe
for i=1:n
    xpX(i)=X1;
    xpT(i)=tp;
end
xp=[xpX xpT];
% ----- wywołanie solvera ODE
[t,x]=ode23s(@uk_row,[czasp czask],xp);
% ----- wizualizacja wyników
[w,k]=size(t);
xi=linspace(0,R,n);
yi=linspace(0,t(w,k),w);
[XI,YI]=meshgrid(xi,yi);

figure(1)
mesh(XI,YI/60,x(:,1:n));
view(150,30)
title('Suszenie kuli - zawartość wilgoci');
xlabel('r [m]')
ylabel('\tau [min]')
zlabel('X [kg/kg]')

figure(2)
mesh(XI,YI/60,x(:,n+1:n+n));
title('Suszenie kuli - temperatura');
xlabel('r [m]')
ylabel('\tau [min]')
zlabel('T [\circ C]')

```

W pliku **uk_row.m** zdefiniowano wyżej podany układ równań różniczkowych:

uk_row.m

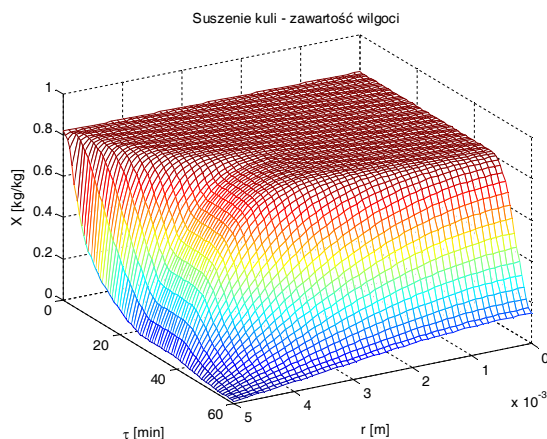
```
function dxdt=uk_row(t,x)
% układ równań różniczkowych
global dr n tg Y
%-----
% równanie ruchu masy dX/dt
pX(1)=Dyf(x(1+n),x(1)).*(2.*(x(2)-x(1))./(dr.^2));
for i=2:n-1
    pX(i)=Dyf(x(i+n),x(i)).*(((x(i+1)-2.*x(i)+x(i-1))./(dr.^2))+...
        (2./(i.*dr)).*((x(i+1)-x(i-1))./(2.*dr)));
end
% warunek brzegowy
wD=ky(x(n+n)).*(Yr(x(n),x(n+n))-Y);
wX1=((2.*dr)./(rom(x(n)).*Dyf(x(n+n),x(n)))).*wD;
pX(n)=Dyf(x(n+n),x(n)).*(((2*(x(n-1)-2.*x(n)-wX1))./(dr.^2))+...
    (2./(n.*dr)).*((-wX1)./(2.*dr)));
% równanie ruchu ciepła dT/dt
pT(1)=(1./(rom(x(1)).*(cpw(x(n+1)).*x(1)+cs(1+n)))).*(lambs(x(n+1)).*((2.*(x(n+2)-x(n+1)))./(dr.^2)) + ...
    Dyf(x(1+n),x(1)).*((2.*(x(2)-x(1)))./(dr.^2)).*cpw(x(n+1)).*x(n+1))+...
    ((cpw(x(n+1)).*x(n+1))./(cpw(x(n+1)).*x(1)+cs(1+n)))).*pX(1);
for i=2:n-1
    pT(i)=(1./(rom(x(i)).*(cpw(x(n+i)).*x(i)+cs(i+n)))).*(lambs(x(n+i)).*((x(n+i+1)-2.*x(n+i)+x(n+i-1))./(dr.^2))+...
        (2./(i.*dr)).*((x(n+i+1)-x(n+i-1))./(2.*dr)))
    +Dyf(x(i+n),x(i)).*(((x(i+1)-2.*x(i)+x(i-1))./(dr.^2))+ ...
        (2./(i.*dr)).*((x(i+1)-x(i-1))./(2.*dr))).*cpw(x(n+i)).*x(n+i))+...
        ((cpw(x(n+i)).*x(n+i))./(cpw(x(n+i)).*x(i)+cs(i+n)))).*pX(i);
end
% warunek brzegowy
wT1=((2.*dr.*alfa(x(n+n)))./lambs(x(n+n)))*(x(n+n)-tg)+2.*dr.*hvw(x(n+n)).*wD./lambs(x(n+n));
pT(n)=(1./(rom(x(n)).*(cpw(x(n+n)).*x(n)+cs(n+n)))).*(lambs(x(n+n)).*((2.*x(n+n-1)-wT1-2.*x(n+n))./(dr.^2)) + ...
    (2./(n.*dr)).*((-wT1)./(2.*dr))) +
    (Dyf(x(n+n),x(n)).*(((2.*x(n-1)-wX1-2.*x(n))./(dr.^2)) + ...
        (2./(n.*dr)).*((-wX1)./(2.*dr)))).*cpw(x(n+n)).*x(n+n))+...
    ((cpw(x(n+n)).*x(n+n))./(cpw(x(n+n)).*x(n)+cs(n+n)))).*pX(n);
% wektor pochodnych do całkowania
```

$$dxdt=[pX \ pT]';$$

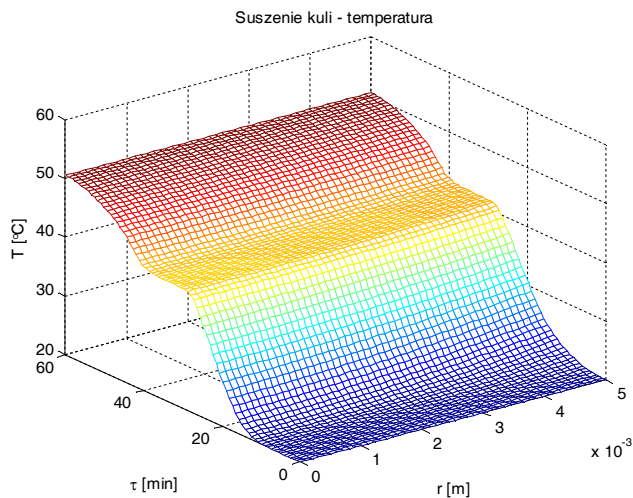
Pozostałe funkcje użyte w projekcie zawierają następujące zmienne:

alfa	współczynnik wnikania ciepła powietrze-cząstka [W/(m ² K)]
btoX	zamiana wilgotności w % na [kg/kg]
cpg	ciepło właściwe suchego powietrze [J/(kg K)]
cpv	ciepło właściwe pary wodnej [J/(kg K)]
cpw	ciepło właściwe wody ciekłej [J/(kg K)]
cs	ciepło właściwe ciała stałego [J/(kg K)]
Dyf	dyfuzyjność wilgoci w materiale [m ² /s]
fis	równowagowa wilgotność względna [-]
hvw	ciepło parowania wody [J/kg]
iwp	entalpia pary [J/kg]
ky	współczynnik wnikania masy [kg/(m ² s)]
lambg	przewodność cieplna powietrza [W/(m K)]
lambs	przewodność cieplna ciała stałego [W/(m K)]
nig	lepkość dynamiczna powietrza [Pa s]
rog	gęstość powietrza [kg/m ³]
yr	równowagowa wilgotność bezwzględna powietrza [kg/kg] (zawiera izotermę sorpcji materiału).

Wyniki 8.14



Rys. 8.27. Rozkład zawartości wilgoci w cząstce



Rys. 8.28. Rozkład temperatury w cząstce

PROJEKT 8.15

Temat: Wymiana ciepła i masy w procesie suszenia przepływem powietrza przez warstwę sferycznych cząstek z oporem dyfuzyjnym wewnątrz cząstki

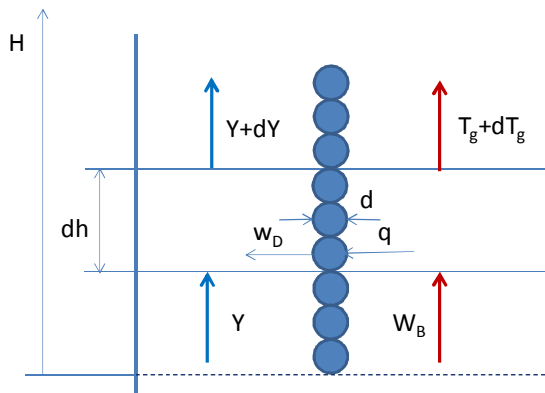
Toolbox: -

W pewnej chwili rozpoczęto przedmuchiwanie powietrzem złoża o wysokości H , wilgotności X_0 i temperaturze t_0 . Powietrze ma wilgotność Y_0 i temperaturę t_{g0} . Należy obliczyć zmianę ww., parametrów wzdłuż wysokości złoża, przyjmując, że w cząstkach występuje dyfuzyjny transport ciepła i masy.

Wysokość złoża 3 m, średnica cząstki 8 mm, początkowa zawartość wilgoci 0.7 kg/kg, natężenie przepływu powietrza 0.1 kg/s, temperatura powietrza 50°C. Pozostałe dane zawarte są w pliku głównym projektu.

Zadanie to ilustruje zmienność warunków brzegowych w czasie wywołaną wymianą ciepła i masy od cząstek do skończonej ilości gazu. Wymaga to uwzględnienia bilansu ciepła i masy fazy gazowej.

Rozważmy element różniczkowej objętości złoża, pokazany na Rys. 8.29.



Rys. 8.29. Schemat różniczkowego elementu objętości złoża – po lewej bilans masy wilgoci, po prawej – bilans ciepła fazy gazowej

Bilans masy fazy gazowej (identyczny jak równ. 8.15):

$$W_B Y - W_B \left(Y + \frac{dY}{dh} dh \right) + S dh a_v w_D = \varepsilon S dh \rho_g \frac{dY}{d\tau} \quad (8.115)$$

Akumulację w fazie gazowej zaniedbujemy jako bardzo małą i po przekształceniu otrzymujemy:

$$\frac{dY}{dh} = \frac{S}{W_B} w_D a_v. \quad (8.116)$$

Bilans entalpii fazy gazowej:

$$W_B i_g - W_B \left(i_g + \frac{di_g}{dh} dh \right) - S dh a_v (q - w_D h_A) = \varepsilon S dh \rho_g c_g \frac{dt_g}{d\tau}. \quad (8.117)$$

Akumulację w fazie gazowej zaniedbujemy jako bardzo małą i otrzymujemy:

$$\frac{di_g}{dh} = \frac{S}{W_B} a_v (q - w_D h_A) = 0, \quad (8.118)$$

gdzie:

$$i_g = (c_L Y + c_g) T_g + \Delta h_{v0} Y. \quad (8.119)$$

Strumienie międzyfazowe obliczymy jako:

$$q = \alpha (T_g - T_p) \text{ i } w_D = k_Y (Y_p^* - Y), \quad (8.120)$$

gdzie indeks p – oznacza warunki na powierzchni cząstki. Do wyznaczenia Y_p^* użyjemy izotermy sorpcji materiału.

Entalpię pary obliczamy jako

$$h_A = c_L T_p + \Delta h_{v0}. \quad (8.121)$$

Równania (8.116) i (8.118) będą całkowane metodą Eulera po podzieleniu wysokości złoża na warstwy. Każdą warstwę reprezentować będzie jedna cząstka, której bilanse różniczkowe pozwolą na obliczenie wartości T_p i Y_p^* . Zaprezentowane są one poniżej. W ten sposób dla każdej z warstw, na które podzieliliśmy złoże, uzyskamy w danej chwili czasu wartość lokalnej temperatury i wilgotności złoża jak również wewnętrzne rozkłady zawartości wilgoci i temperatury złoża w cząstce należącej do danej warstwy. Opisane są one następującymi równaniami:

Zmianę wilgotności we wnętrzu cząstki kulistej opisuje równanie:

$$\frac{\partial X}{\partial \tau} = D_m \left(\frac{\partial^2 X}{\partial r^2} + \frac{2}{r} \frac{\partial X}{\partial r} \right). \quad (8.122)$$

Warunki brzegowe dla tego równania są następujące:

– w osi cząstki:

$$\left. \frac{\partial X}{\partial r} \right|_{r=0} = 0, \quad (8.123)$$

– na powierzchni cząstki:

$$-\rho_m D_m \left. \frac{\partial X}{\partial r} \right|_{r=R} = k_Y (Y - Y_p^*). \quad (8.124)$$

Zmianę temperatury fazy stałej opisuje równanie:

$$\frac{\partial T}{\partial \tau} = \frac{1}{\rho_m (c_L X + c_S)} \left[\lambda \left(\frac{\partial^2 T}{\partial r^2} + \frac{2}{r} \frac{\partial T}{\partial r} \right) - \frac{\partial X}{\partial \tau} c_L T \right] - \frac{c_L T}{c_L X + c_S} \frac{\partial X}{\partial \tau}. \quad (8.125)$$

Warunki brzegowe dla tego równania:

– w osi cząstki:

$$\left. \frac{\partial T}{\partial r} \right|_{r=0} = 0, \quad (8.126)$$

– na powierzchni cząstki:

$$-\lambda \left. \frac{\partial T}{\partial r} \right|_{r=R} = \alpha (t_g - T_p). \quad (8.127)$$

Na tej podstawie wyprowadzono równania robocze na ich postaci zdyskretyzowane (zob. Projekt 8-14).

P8_15.m

%PROJEKT 8.15

%Suszenie głębokiego złoża cząstek z oporem wewnętrznym

clear all; close all; clc; memory

praca1 = cputime;

global Dyf dr n Y rom wx kx x S W a j d tg u R sf ig u

%-----dane fizyczne

H=3; % wysokość złoża [m]

S=1; % powierzchnia przekroju złoża [m^2]

W=0.1; % natężenie przepływu powietrza [kg/s]

hr=15*24; % czas suszenia [godz]

d=0.008; % średnica cząstki [m]

rom=1550; % gęstość materiału [kg/m^3]

sf=1; % sferyczność [-]

eps=0.45; % porowatość złoża [-]

X0=0.7; % pocz. zawartość wilgoci w cząstkach [kg/kg]

T0=20; % temperatura początkowa materiału [C]

Y0=0.01; % wlotowa wilgotność powietrza [kg/kg]

Tg0=50; % wlotowa temperatura powietrza [C]

a=6*(1-eps)/(d*sf); % powierzchnia właściwa złoża [m^2/m^3]

u=W/(rog(Tg0)*S); % prędkość powietrza [m/s]

R=d/2; % promień cząstki [m]

Dyf=15*10^(-6); % dyfuzyjność pary w powietrzu [m^2/s]

%-----
% inne właściwości zależne od temperatury
% zawarte są w odpowiednich funkcjach

```

%-----dyskretyzacja
n=10;                % liczba komórek dyskretyzacji w cząstce [szt]
ntime=60;            % liczba kroków czasowych [szt]
nw=200;              % liczba warstw złoża [szt]
dr=R/n;              % wielkość komórki dyskretyzacji promienia
DH=H/nw;             % wysokość warstwy [m]
                    % wektor czasu
time=linspace(0,hr*3600,ntime);

%-----warunki początkowe
% w pierwszej cząstce
for i=1:n
    xpX(1,i)=X0;
    xpT(1,i)=T0;
end
% we wszystkich cząstkach identycznie jak w pierwszej
for z=1:nw %
warpocz(z,1:2*n,1)=[xpX xpT];
end

%-----rozwiązywanie
for j=2:length(time) % kolejny krok czasowy
    krok_czasu=j;
    t=time(j);
    Y=Y0;
    tg=Tg0;
    ig=(cpg(Tg0)+cpw(Tg0).*Y0)*Tg0+hvw(0)*Y0;
    for i=1:nw %rozkłady X i T w cząstkach w każdej warstwie
        xp=warpocz(i,:,j-1);
        [t,x]=ode15s(@uk_row,[time(j-1) time(j)],xp);
        kulki{i,j}=[t x];
        [wx,kx]=size(x);
        tp=x(wx,kx);
        % obliczenie bilansów w fazie gazowej
        % strumień ciepła
        q=alfa(x(wx,kx)).*(tg-tp);
        % strumień masy
        wd=wDs(x(wx,n),tp);
        % wilgotność gazu
        zY(i,j)=Y+(S/W)*wd*a*DH;
        Y=zY(i,j);
        % temperatura gazu
        zig(i,j)=ig-(S/W)*(q-wd*hvw(tp))*a*DH;
    end
end

```

```

        ig=zig(i,j);
        ztg(i,j)=fzero(@ign,tg);
        tg=ztg(i,j);
    end

    for k=1:nw
        xpom=kulki{k,j};
        [xpomw,xpomk]=size(xpom);
        warpocz(k,:,j)=xpom(xpomw,2:xpomk);
        clear xpom xpomw xpomk
    end
end

%-----prezentacja wyników
Y(1,1:length(time))=Y0;
YY=[Y; zY];
T(1,1:length(time))=Tg0;
TG=[T; ztg];

figure(1)
axis([0 105 0.0 0.02]);
for i=1:ntime
    if mod(i,5)==0 % rysuje profile co 5*hr/ntime godzin
        plot(YY(:,i),'g')
        hold on
    end
end
title('Rozkład Y gazu po wysokości złoża');
xlabel('wysokość [nr węzła]');
ylabel('wilgotność [kg/kg]');

figure(2)
axis([0 105 10 60]);
for i=1:ntime
    if mod(i,5)==0 % rysuje profile co 5*hr/ntime godzin
        plot(TG(:,i),'m')
        hold on
    end
end
title('Rozkład T gazu po wysokości złoża');
xlabel('wysokość [nr węzła]');
ylabel('temperatura [\text{C}]');

```



```

clear Xob Tob
% średnie wartości wilgotności i temperatury cząstek
for jt=2:ntime
for jw=1:nw
clear x zz Xz s1 s2 wX kX
Xz=kulki{jw,jt};
[iw,ik]=size(Xz);
x=Xz(1:iw,2:ik);
[wX,kX]=size(x);
zz=[linspace(0,n,n)]';
% całkowanie profili po objętości dla początkowego czasu
% każdego przedziału
dr=R/n;
rr=zz.*dr;
s1=0; s2=0;
for i=2:n
    X1=x(1,i);
    t1=x(1,i+n);
    im1=(cs(t1)+cpw(t1)*X1)*t1;
    X2=x(1,i-1);
    t2=x(1,i-1+n);
    im2=(cs(t2)+cpw(t2)*X2)*t2;
    s1=s1+(1/R.^3).*((rr(i).^3-rr(i-1).^3).*(X1+X2)./2);
    s2=s2+(1/R.^3).*((rr(i).^3-rr(i-1).^3).*(im1+im2)./2);
end
Xob(jw,jt)=s1;
Tob(jw,jt)=s2/(cs(t2)+cpw(t2)*s1);
end %jw
end %jt

figure(3)
axis([0 100 0 0.8]);
for i=2:ntime
    if mod(i,5)==0 % rysuje profile co 5*hr/ntime godzin
        plot(Xob(:,i),'b')
        hold on
    end
end
title('Rozkład X materiału po wysokości złoża');
xlabel('wysokość [nr węzła]');
ylabel('X [kg/kg]');
hold on

```

```

figure(4)
axis([0 100 10 60]);
for i=2:ntime
    if mod(i,5)==0 % rysuje profile co 5*hr/ntime godzin
        plot(Tob(:,i),'r')
        hold on
    end
end

title('Rozkład T materiału po wysokości złoża');
xlabel('wysokość [nr węzła]');
ylabel('T [\circ C]');
hold on

pX(1,1:n)=X0;
pT(1,1:n)=T0;
for jt=2:ntime
    for jw=10 %wybrany numer warstwy
        clear x zz Xz wX kX
        Xz=kulki{jw,jt};
        [iw,ik]=size(Xz);
        x=Xz(1:iw,2:ik);
        [wX,kX]=size(x);
        ww=x(1,1:n);
        vv=x(1,n+1:2*n);
        pX=[pX;ww];
        pT=[pT;vv];
    end %jw
end %jt

figure(5);
for i=2:ntime
    plot(pX(i,:), '-bo');
    hold on
end;
title('Profile X w cząstce nr 10');
xlabel('promień [nr węzła]');
ylabel('X [kg/kg]');

figure(6);
for i=2:ntime
    plot(pT(i,:), '-ro');
    hold on

```

```

end;
title('Profile T w cząstce nr 10');
xlabel('promień [nr węzła]');
ylabel('T [C]');

praca2=cputime;
disp('Czas wykonania [min]'); disp((praca2-praca1)/60);

```

Pośród funkcji, które zdefiniowano w odrębnych plikach pokażemy plik **uk_row.m**, w którym podano wyrażenia obliczające pochodne po czasie, korzystające z dyskretyzacji pochodnych przestrzennych we wszystkich węzłach siatki, na którą podzielono promień cząstki.

uk_row.m

```

function dxdt=uk_row(t,x)
% układ równań różniczkowych modelu

global Dyf dr n Y rom S W a tg
% dXdT
pX(1)=DyfM(x(1),x(n+1)).*((2.*x(2)-2.*x(1))./(dr.^2));
for i=2:n-1
    pX(i)=DyfM(x(i),x(n+i)).*((((x(i+1)-2*x(i)+x(i-1)))./(dr.^2))+...
        (2/(i.*dr)).*(x(i+1)-x(i-1))./(2.*dr)));
end

% warunek brzegowy dla dXdT
wX1=((2.*dr)./(rom.*DyfM(x(n),x(n+n)))).*(wDs(x(n),x(n+n)));
pX(n)=DyfM(x(n),x(n+n)).*((2.*x(n-1)-2.*x(n)-wX1)./(dr.^2)+(2/(n.*dr)).*(-wX1)./(2.*dr)));
% dTdt
pT(1)=(1./(rom.*(cpw(x(n+1)).*x(1)+cs(x(1+n))))).*...
    (lambs(x(n+1)).*((2.*(x(n+2)-x(n+1)))./(dr.^2)) + ...
    pX(1).*cpw(x(n+1)).*x(n+1))+...

((cpw(x(n+1)).*x(n+1))./(cpw(x(n+1)).*x(1)+cs(x(1+n)))).*pX(1);
for i=2:n-1
pT(i)=(1./(rom.*(cpw(x(n+i)).*x(i)+cs(x(i+n))))).*...
    (lambs(x(n+i)).*((((x(n+i+1)-2*x(n+i)+x(n+i-1)))./(dr.^2))+...
    (2/(i.*dr)).*(x(n+i+1)-x(n+i-1))./(2.*dr)) + ...
    pX(i).*cpw(x(n+i)).*x(n+i))+...

```

```

((cpw(x(n+i)).*x(n+i))./(cpw(x(n+i)).*x(i)+cs(x(i+n)))).*pX(i);
end

% warunek brzegowy dla dTdt
wT1=-(2*dr*(alfa(x(n+n)).*(tg-x(n+n))-
wDs(x(n),x(n+n))*hvw(x(n+n)))./(lambd(x(n+n))));
pT(n)=(1./(rom.*(cpw(x(n+n)).*x(n)+cs(x(n+n)))).*...
(lambd(x(n+n)).*((2.*x(n+n-1)-2.*x(n+n)-
wT1)./(dr.^2)+(2/(n.*dr)).*(-wX1)./(2.*dr)) +...
pX(n).*cpw(x(n+n)).*x(n+n)))+...

((cpw(x(n+n)).*x(n+n))./(cpw(x(n+n)).*x(n)+cs(x(n+n)))).*pX(n);

dxdt=[pX pT]';

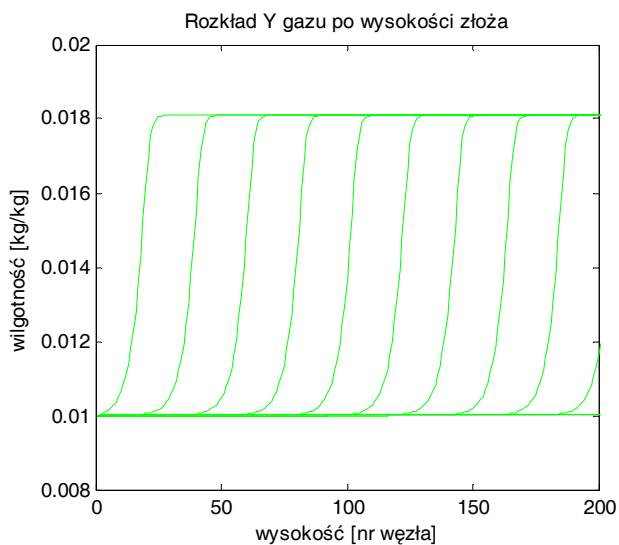
```

Pozostałe funkcje pełnią rolę bazy danych fizykochemicznych lub wykonują obliczenia pomocnicze. Są to następujące funkcje (alfabetycznie):

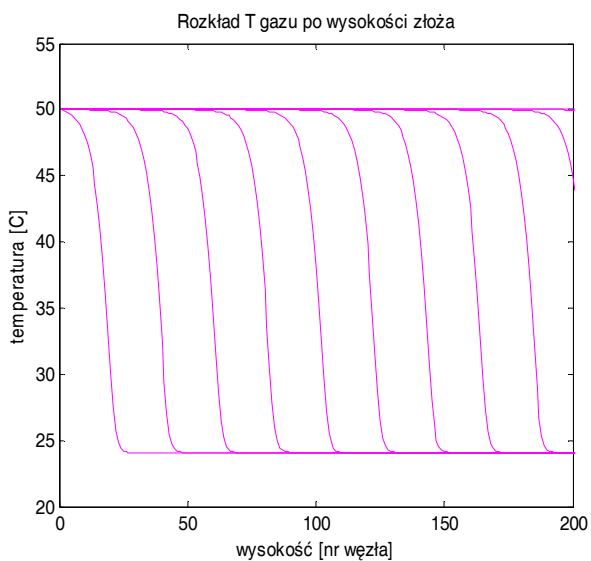
alfa	współczynnik wnikania ciepła powietrze-cząstka [W/(m ² K)]
cpv	ciepło właściwe suchego powietrza [J/(kg K)]
cpw	ciepło właściwe pary wodnej [J/(kg K)]
cs	ciepło właściwe wody ciekłej [J/(kg K)]
DyfM	dyfuzyjność wilgoci w materiale [m ² /s]
hvw	ciepło parowania wody [J/kg]
ign	entalpia wilgotnego powietrza [J/kg]
ky	współczynnik wnikania masy [kg/(m ² s)]
lambg	przewodność cieplna powietrza [W/(m K)]
lambd	przewodność cieplna ciała stałego [W/(m K)]
nig	lepkość dynamiczna powietrza [Pa s]
rog	gęstość powietrza [kg/m ³]
wDs	szybkość suszenia [kg/(m ² s)]
yr	równowagowa wilgotność bezwzględna powietrza [kg/kg] (zawiera izotermę sorpcji materiału).

Wyniki 8.15

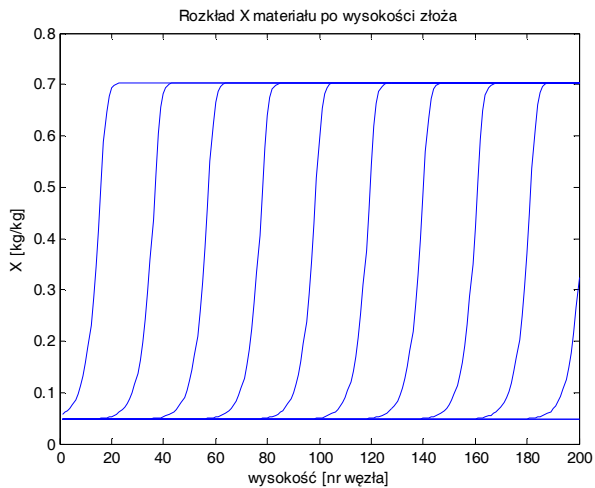
Wyniki uzyskano w formie graficznej w postaci 6 rysunków.



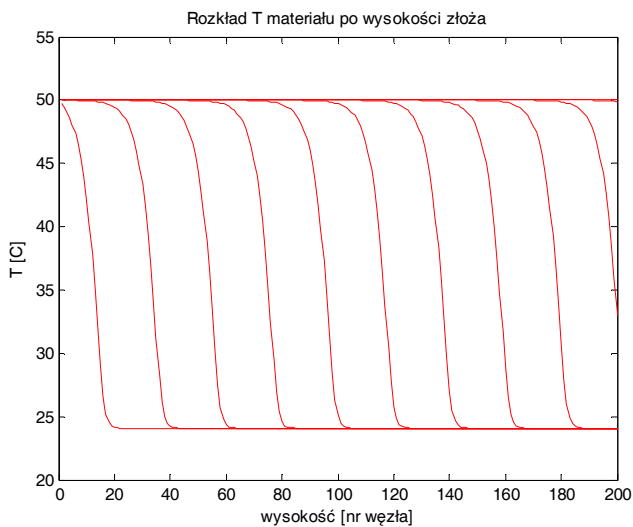
Rys. 8.30. Rozkład wilgotności powietrza po wysokości złoża w miarę upływu czasu.
Przyrost czasu między liniami 40 h



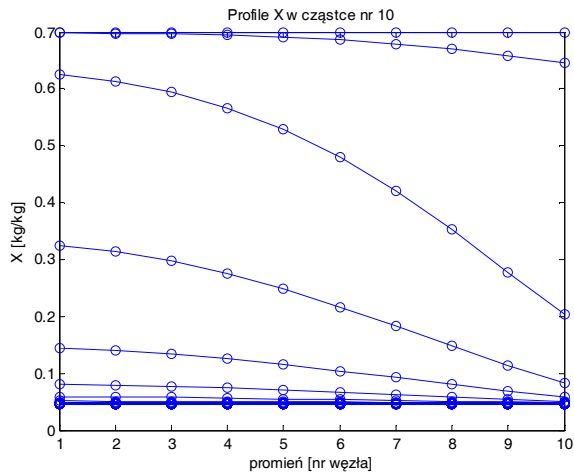
Rys. 8.31. Rozkład temperatury powietrza po wysokości złoża w miarę upływu czasu.
Przyrost czasu między liniami 40 h



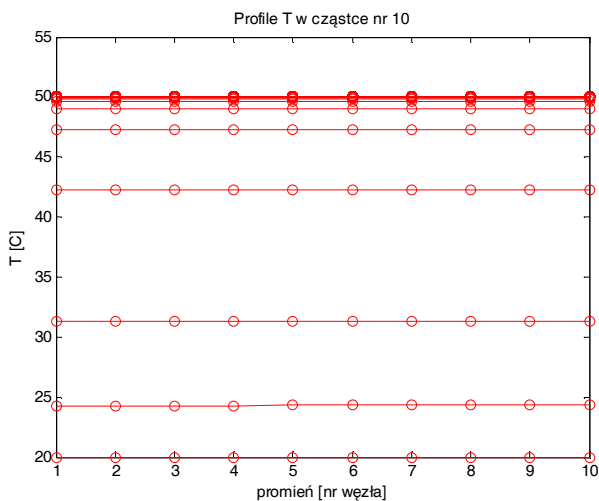
Rys. 8.32. Zmiana średniej zawartości wilgoci cząstek po wysokości złoża w miarę upływu czasu. Przyrost czasu między liniami 40 h



Rys. 8.33. Zmiana średniej temperatury cząstek po wysokości złoża w miarę upływu czasu. Przyrost czasu między liniami 40 h



Rys. 8.34. Radialny rozkład zawartości wilgoci X w cząstce dla warstwy nr 10.
Przyrost czasu między liniami 8 h



Rys. 8.35. Radialny rozkład temperatury w cząstce dla warstwy nr 10.
Przyrost czasu między liniami 8 h

Jak można zauważyć, opór wewnętrzny transportu ciepła jest nieznaczny i można było go pominąć. Opór transportu masy jest znaczny i wpływa na czas suszenia.

Dla odparowania masy wody równej:

$$m_w = HS\rho_m(1 - \varepsilon)X_0 \cong 1790 \text{ kg} \quad (8.128)$$

w tych warunkach potrzeba ok 400 godz., tj. około 17 dni.

Zagadnienie powyższe należy do tzw. modelowania wieloskalowego (ang. *multiscale modeling*) na dwu poziomach. Całkując krok po kroku bilanse fazy gazowej (duża skala) w każdym kroku rozwiązujemy równanie PDE transportu wilgoci w cząstce (mała skala). Powoduje to zarówno znaczne zużycie pamięci jak i wydłuża czas obliczeń, który jest tu kilkunastominutowy i zależy od wydajności procesora.

Oznaczenia:

D_m	dyfuzyjność wody w materiale [m^2/s]
T	temperatura materiału [$^{\circ}C$]
S	przekrój poprzeczny złoża [m^2]
Y	wilgotność gazu [kg/kg]
W_B	natężenie przepływu gazu [kg/s]
a_v	powierzchnia właściwa złoża [m^2/m^3]
c_L	ciepło właściwe wody [$J/(kg \text{ K})$]
c_g	ciepło właściwe gazu [$J/(kg \text{ K})$]
c_s	ciepło właściwe materiału [$J/(kg \text{ K})$]
i_g	entalpia gazu [J/kg]
k_y	współczynnik wnikania masy [kg/m^2s]
t_g	temperatura gazu [$^{\circ}C$]
α	współczynnik wnikania ciepła [$W/(m^2 \text{ K})$]
λ	przewodność ciepła materiału [$W/(m \text{ K})$]
ρ_g	gęstość gazu [kg/m^3]

PROJEKT 8.16

Temat:	Nieustalony ruch ciepła i masy w procesie suszenia parą parą przegrzaną materiału o kształcie walca
Toolbox:	-

Rozpatrzmy nieustalony ruch ciepła i masy w procesie suszenia parą przegrzaną ciała o geometrii cylindrycznej. Długość walca jest skończona, ale jego podstawy są izolowane, aby zapobiec wymianie ciała i masy. Dany jest promień walca, temperatura pary i współczynnik wnikania ciepła od pary do materiału. Wśród danych fizykochemicznych znana jest izobara sorpcji materiału oraz wartość

dyfuzyjności efektywnej wilgoci w materiale i jego przewodność cieplna. Dane są również temperatura i zawartość wilgoci w materiale na początku procesu oraz temperatura i ciśnienie pary. Wszystkie dane są zawarte w listingu programu. W sformułowaniu równań transportu ciepła i masy w geometrii cylindrycznej wykorzystamy teorię zawartą w Projekcie 8.4 i 8.14.

Specyfika suszenia parą przegrzaną powoduje, że szybkość suszenia nie może być zdefiniowana w oparciu o współczynnik wnikania masy (brak siły napędowej wnikania masy), lecz definiujemy ją w oparciu o współczynnik wnikania ciepła

$$w_D = \frac{\alpha \cdot (T - T^*)}{\Delta h_v(T)}, \quad (8.129)$$

gdzie α – współczynnik wnikania ciepła [$\text{W}/\text{m}^2\text{K}$], Δh_v – ciepło parowania [J/kg], T – temperatura materiału [$^{\circ}\text{C}$], T^* – temperatura równowagowa odczytana z izobary sorpcji lub równania równowagi opisującego równowagę w szerokim zakresie temperatur i wilgotności [$^{\circ}\text{C}$]. Koncepcja ta została zaproponowana w pracy *Pakowski, Adamski, 2011*.

P8_16.m

```
%%PROJEKT 8.16
```

```
%Suszenie walca parą przegrzaną
```

```
%Rozwiązanie PDE metodą linii
```

```
clear all; close all; clc;
```

```
global n R dr Xcr Xeq tg P0 up mip lambp
```

```
n=50; % liczba punktów dyskretyzacji [szt]
```

```
R=0.01/2; % średnica walca [m]
```

```
dr=R/n; % szerokość komórki [m]
```

```
P0=101.325; % ciśnienie atmosferyczne [kPa]
```

```
b1=0.45; % wilgotność początkowa [%]
```

```
b2=0.05; % wilgotność końcowa [%]
```

```
tm0=15; % początkowa temperatura ciała stałego [C]
```

```
X1=btoX(b1); % początkowa zaw. wilgoci [kg/kg]
```

```
Xeq=btoX(b2); % końcowa zaw. wilgoci [kg/kg]
```

```
Xcr=0.4; % krytyczna zaw. wilgoci [kg/kg]
```

```
up=5; % prędkość liniowa pary [m/s]
```

```
mip=14e-6; % lepkość pary wodnej [Pas]
```

```
lambp=3e-2; % przewodność cieplna pary wodnej [W/(m*K)]
```

```
tg=150; % temperatura pary przegrzanej [C]
```

```
tp=15; % temperatura początkowa materiału [C]
```

```

czasp=0;
czask=50*3600; % czas suszenia walca [s]
% warunek początkowy
for i=1:n
    xpX(i)=X1;
    xpT(i)=tp;
end
xp=[xpX xpT];
% wywołanie solvera ODE
[t,x]=ode15s(@uk_row,[czasp czask],xp);
[w,k]=size(t);
xi=linspace(0,R,n);
yi=linspace(0,t(w,k),w);
[XI,YI]=meshgrid(xi,yi);
%średnie X
for j=1:w
    s(j)=0;
    for i=2:n
        s(j)=s(j)+(xi(i).^2-xi(i-1).^2)*(x(j,i)+x(j,i-1))/2;
    end
    s(j)=s(j)./R^2;
end
% wizualizacja wyników
% średnia wilgotność w czasie
figure(1)
plot(t/3600,s)
grid on;
xlabel('Czas [h]');
ylabel('Średnia zawartość wilgoci [kg/kg]');
% rozkład zawartości wilgoci
figure(2)
mesh(XI*1000,YI/3600,x(:,1:n));
xlabel('Promień [mm]');
ylabel('Czas [h]');
zlabel('Zawartość wilgoci [kg/kg]');
view(150,30)
% rozkład temperatury
figure(3)
mesh(XI*1000,YI/3600,x(:,n+1:n+n));
xlabel('Promień [mm]');
ylabel('Czas [h]');
zlabel('Temperatura [\circ C]');

```

W pliku **uk_row.m** zdefiniowano równania obliczające rozkład wilgoci i temperatury wewnątrz materiału

uk_row.m

```
function dxdt=uk_row(t,x)
% układ równań różniczkowych
global dr n tg XX
%-----
t0=273.15;
XX=x(n);
% obliczenia temperatury równowagowej
tr=fzero(@tmr,150);
pX(1)=Dyf.*(2.*(x(2)-x(1))./(dr.^2));
for i=2:n-1
    pX(i)=Dyf.*(((x(i+1)-2.*x(i)+x(i-1))./(dr.^2))+...
        (1./(i.*dr)).*((x(i+1)-x(i-1))./(2.*dr)));
end
% szybkość suszenia
wDs=((x(n+n)-tr).*alfa)./hvw(x(n+n));
wX1=((2.*dr)./(rom(x(n)).*Dyf)).*(wDs);
pX(n)=Dyf.*(((2*(x(n-1)-2.*x(n)-wX1))./(dr.^2))+...
    (1./(n.*dr)).*((-wX1)./(2.*dr)));
pT(1)=(1./(rom(x(1)).*(cpw(x(n+1)).*x(1)+cs(1+n)))).*...
    (lambs(x(n+1)).*((2.*(x(n+2)-x(n+1)))./(dr.^2)) + ...
    Dyf.*((2.*(x(2)-
x(1)))./(dr.^2)).*cpw(x(n+1)).*x(n+1))+...
    ((cpw(x(n+1)).*x(n+1))./(cpw(x(n+1)).*x(1)+cs(1+n))).*pX(1);
for i=2:n-1
    pT(i)=(1./(rom(x(i)).*(cpw(x(n+i)).*x(i)+cs(i+n)))).*...
        (lambs(x(n+i)).*((x(n+i+1)-2.*x(n+i)+x(n+i-
1))./(dr.^2))+...
        (2./(i.*dr)).*((x(n+i+1)-x(n+i-1))./(2.*dr)))
    +Dyf.*(((x(i+1)-2.*x(i)+x(i-1))./(dr.^2))+ ...
        (2./(i.*dr)).*((x(i+1)-x(i-
1))./(2.*dr))).*cpw(x(n+i)).*x(n+i))+...
    ((cpw(x(n+i)).*x(n+i))./(cpw(x(n+i)).*x(i)+cs(i+n))).*pX(i);
end
wT1=((2.*dr.*alfa(x(n+n)))./lambs(x(n+n)))*...
    (x(n+n)-tg)+2.*dr.*hvw(x(n+n))*wDs./lambs(x(n+n));
pT(n)=(1./(rom(x(n)).*(cpw(x(n+n)).*x(n)+cs(n+n)))).*...
    (lambs(x(n+n)).*((2.*x(n+n-1)-wT1-
2.*x(n+n))./(dr.^2)) + ...
```

```

(2./(n.*dr)).*((-wT1)./(2.*dr))) + (Dyf.*(((2.*x(n-1)-
wX1-2.*x(n))./(dr.^2)) + ...
(2./(n.*dr)).*((-
wX1)./(2.*dr))).*cpw(x(n+n)).*x(n+n))+...

((cpw(x(n+n)).*x(n+n))./(cpw(x(n+n)).*x(n)+cs(n+n))).*pX(n);

dxdt=[pX pT]';

```

W pliku **tmr.m** zdefiniowano równanie równowagi. Funkcja ta jest lewą stroną równania, które rozwiązane w funkcji **uk_row.m** oblicza temperaturę równowagową dla danej zawartości wilgoci X i ciśnienia ogólnego $P0$ (przekazywane przez zmienne globalne).

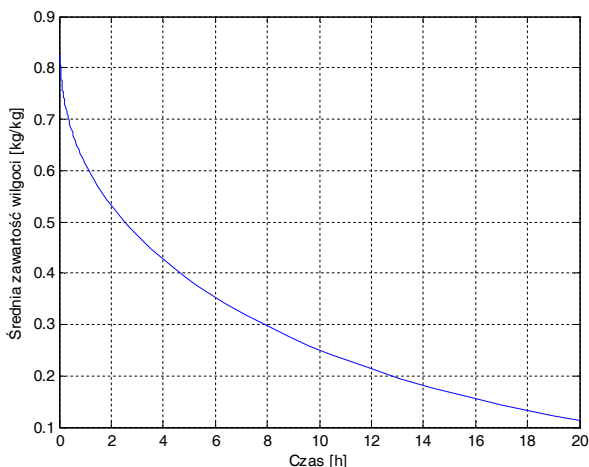
 **tmr.m**

```

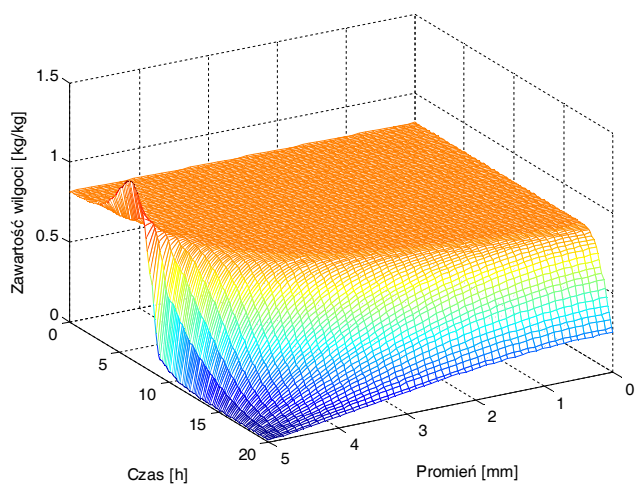
function y=tmr(Tr)
% temperatura równowagowa materiału
global P0 XX
%--prężność pary wodnej
a11=16.376953; a22=3878.8223; a33=229.862;
%--współczynniki w równaniu Hendersona
a1=-0.016834; a2=213.9; a3=0.68475;
y=(1-exp(a1.*(a2+Tr).*XX.^a3)) - P0./(exp(a11-a22./(a33+Tr)));

```

Wyniki 8.16

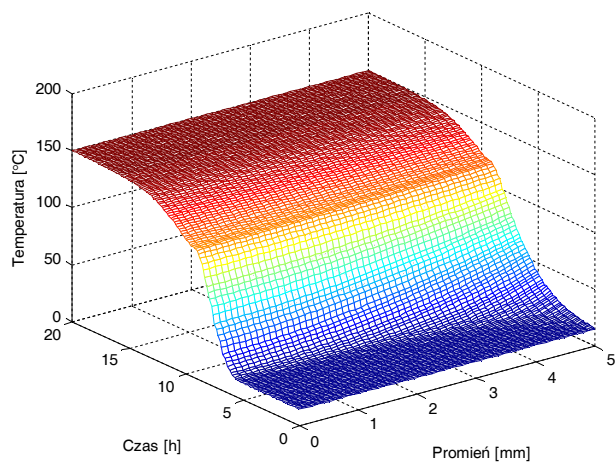


Rys. 8.36. Zmiana w czasie średniej zawartości wilgoci



Rys. 8.37. Rozkład zawartości wilgoci w materiale

Zauważmy, że na początku procesu materiał nawilża się powierzchniowo. Jest to spowodowane kondensacją pary na chłodnej powierzchni materiału.



Rys. 8.38. Rozkład temperatury w materiale

LITERATURA

- Crank J., „*The Mathematics of Diffusion*”, Clarendon Press, Oxford 1956.
- Incropera F.P., Hewitt D.P., Bergman T.L., Lavine A.S., „*Fundamentals of Heat and Mass Transfer*”, J.Wiley, 2007, ss. 212-218.
- Kudra T. (red.), „*Zbiór zadań z podstaw teoretycznych inżynierii chemicznej i procesowej*”, WNT, Warszawa 1985.
- Osthuizen P., Naylor D., „*Introduction to Convective Heat Transfer Analysis*”, WCB/McGrawHill, 1999, ss. 179-183.
- Pakowski Z., Głębowski M., „*Symulacja procesów inżynierii chemicznej*”, Wyd. PŁ, Łódź 2001.
- Pakowski Z., Adamski R., *On prediction of the drying rate in superheated steam drying process*, *Drying Technology*, 2011, 29, 1492-1498.
- Pakowski Z., Adamski R., Kwapisz S., *Effective diffusivity of moisture in low rank coal during superheated steam drying at atmospheric pressure*, *Chem. Eng. & Proc.* 2011, 32(4/2), 43-51.
- Pakowski Z., „*dryPAK v.3.6 Program for psychrometric and drying calculations*”, OMNIKON, Łódź 2000.
- Zaleski T., „*Metody algebry liniowej w inżynierii procesowej*”, PWN, Warszawa 1987.

9. IDENTYFIKACJA PARAMETRÓW RÓWNAŃ RÓŻNICZKOWYCH – ROZWIĄZYWANIE ZAGADNIEŃ ODWROTNYCH

9.1. Sformułowanie problemu identyfikacji parametrów

W inżynierii chemicznej i procesowej buduje się modele procesów, które są wykorzystywane do symulacji, optymalizacji, projektowania i sterowania procesów. W zależności od natury procesu modele te składają się z różnych typów równań. W ogólności można je podzielić na cztery grupy, jak pokazuje Tablica 9.1.

Tablica 9.1. Elementarna klasyfikacja modeli procesów

	Stan ustalony	Stan nieustalony
Obiekty o parametrach skupionych	Równania algebraiczne liniowe lub nieliniowe	Równania różniczkowe zwyczajne I rzędu z pochodną po czasie
Obiekty o parametrach rozłożonych	Równania różniczkowe zwyczajne I rzędu z pochodną po wymiarze	Równania różniczkowe cząstkowe II rzędu

Wymienione są tu najczęściej występujące typy równań, lecz możliwe są przypadki szczególne. I tak np. uwzględnienie dyspersji przepływu w równaniu bilansowym obiektu o parametrach rozłożonych w stanie ustalonym (np. reaktor rurowy) prowadzi do równania II rzędu, a uwzględnienie drugiego czy trzeciego wymiaru przestrzennego, w tym samym przypadku, prowadzi do równania różniczkowego cząstkowego I rzędu. Współczynniki tych równań są często empiryczne lub półempiryczne i wymagają identyfikacji na podstawie wyników eksperymentu. Ponieważ pomiary są obciążone błędami, do identyfikacji używamy wielu zestawów wyników eksperymentalnych. W przypadku identyfikacji stałych pojedynczego równania prowadzi to do aproksymacji metodą najmniejszych kwadratów (MNK), omówionej w Rozdziale 5.3.1.

9.2. Sformułowanie zagadnienia odwrotnego

Zagadnienia odwrotne dotyczą układów określonych równaniami różniczkowymi cząstkowymi drugiego rzędu, a więc najczęściej problemów przewodzenia ciepła i dyfuzji molekularnej. Ścisłe zagadnieniem odwrotnym nazywamy problem identyfikacji warunków brzegowych na brzegu obszaru na podstawie temperatur bądź stężeń zmierzonych wewnątrz obszaru. W szerszym znaczeniu obejmuje ono również identyfikację warunków brzegowych lub parametrów

modelu w przypadku zmierzonej średniej wartości rozpatrywanej wielkości fizycznej. Przykładowo problem dotyczy znajdowania zależności od temperatury i zawartości wilgoci współczynnika dyfuzji efektywnej wilgoci w suszonym cieple stałym, kiedy w eksperymencie mierzona jest temperatura ciała w jednym jego punkcie i średnia zawartość wilgoci, określana na podstawie ubytku masy. Dla prostych przypadków przewodzenia ciepła istnieją odpowiednie metody analityczne (*J. Taler, P. Duda, 2003*), tutaj jednak posługiwać się będziemy metodami numerycznymi.

9.3. Algorytmy optymalizacyjne

W każdym z przypadków identyfikacji parametrów bądź zagadnienia odwrotnego niezbędne będą odpowiednie algorytmy optymalizacyjne. Zawarte są one w funkcjach dostępnych w głównym programie i skrzynce narzędziowej **Optimization Toolbox**. Poszukiwać będziemy wektora współczynników funkcji, który minimalizuje sumę kwadratów odchyłeń funkcji i punktów pomiarowych. Jest to funkcja celu optymalizacji.

Tablica 9.2. Główne algorytmy optymalizacyjne MATLABa

Nazwa	Typowe wywołanie	Opis
fminbnd	fminbnd(f,x1,x2)	Znajduje minimum funkcji jednej zmiennej w przedziale od x1 do x2.
fminsearch	fminsearch(f,x0)	Znajduje minimum funkcji skalarnej wielu zmiennych startując w punkcie x0.

W powyższej tablicy f to uchwyt funkcji celu. Do wywołań obu tych funkcji dopisać można, jako ostatni parametr, opcje określone jako

```
options=optimset('prametr','wartość,...')
```

gdzie parametrami mogą być np.

'Display'	'off' – wyłącza druk przebiegu iteracji, 'iter' – drukuje wszystkie iteracje, 'final' (domyślna) drukuje tylko ostatnią iterację, 'notify' – drukuje tylko powiadomienie o niepowodzeniu.
'FunValCheck'	'on', 'off'(domyślna) – zawiadamia o wystąpieniu urojonej lub NaN wartości funkcji.
'MaxIter'	dopuszczalna liczba iteracji
'TolFun'	tolerancja wartości funkcji
'TolX'	tolerancja wartości zmiennej niezależnej

Więcej narzędzi optymalizacyjnych znaleźć można w skrzynce **Optimization Toolbox**.

Tablica 9.3. Główne algorytmy optymalizacyjne MNK ze skrzynki
Optimization Toolbox

Nazwa	Typowe wywołanie	Opis
lsqcurvefit	lsqcurvefit(f,x0,xdata,ydata)	Znajduje wektor x minimalizujący funkcję celu w postaci SSE (równ. 5.1) startując w punkcie x0 dla dostarczonych danych
lsqnonlin	lsqnonlin(f,x0) lsqnonlin(f,x0,lb,ub)	Znajduje wektor x minimalizujący funkcję celu w postaci SSE startując w punkcie x0, lb, ub – wektory ograniczeń dolnych i górnych

W zestawie opcji funkcji **optimset** są dostępne dodatkowe opcje np.

‘Algorithm’ umożliwia wybór algorytmu optymalizacji.

‘Diagnostics’ ‘on’, ‘off’ (domyślna) – wyświetlanie diagnostyki funkcji celu.

‘Jacobian’ ‘on’, ‘off’ (domyślna) – dostarczanie przez użytkownika Jakobianu funkcji celu (w celu przyspieszenia obliczeń), inaczej jest on obliczany numerycznie.

Można też posługiwać się innym wiarygodnym algorytmem optymalizacyjnym np. użytą tutaj w Projekcie 9.1 i 9.2 metodą gradientową Hooke’a-Jeevsa. Jej implementację napisano w oparciu o dostępny algorytm tej metody (*Quateroni i in., 2007*).

Wartość funkcji celu w przypadku identyfikacji parametrów funkcji jednej zmiennej jest oparta o jeden lub wiele zestawów danych (serii pomiarowych) otrzymanych przy tych samych lub różnych parametrach eksperymentu. Identyfikacji parametrów funkcji dokonuje się wtedy najlepiej na wszystkich seriach jednocześnie, używając odpowiedniego zestawu parametrów procesu dla każdej serii.

W przypadku funkcji wielu zmiennych można sumować SSE dla każdej zmiennej, poprzedzając je odpowiednią wagą. Wagi będą większe dla zmiennych i pomiarów o dużej wiarygodności, a mniejsze dla tych o mniejszej wiarygodności przy czym suma wag wynosi 1. Końcowy efekt wpływu wag zwykle ocenia się wizualnie oraz obliczając typowe statystyczne testy dopasowania jak R^2 .

Podczas optymalizacji optymalizatorami MATLABa w kolejnych iteracjach należy obserwować wartość „First-order nonlinearity”. Jest to maksymalna aktualna wartość bezwzględna gradientu funkcji. W czasie iteracji powinna ona maleć i osiągnąć wartość możliwie bliską zero, jeśli minimum zostało osiągnięte.

9.4. Identyfikacja współczynników modelu określonego równaniem różniczkowym zwyczajnym

Przypadki tego typu dotyczą najczęściej problemów identyfikacji współczynnika wnikania lub przewodzenia ciepła, dyfuzyjności lub stałych kinetycznych reakcji chemicznych. Ilustrują to poniższe projekty.

PROJEKT 9.1	
Temat:	Identyfikacja współczynników kinetycznych reakcji chemicznej
Toolbox:	-

Podczas badań metanowej kofermentacji osadu ściekowego (*Sosnowski i in., 2008*) dokonano pomiarów kinetyki zmian zawartości węgla w poszczególnych frakcjach: suchej masie (SM), frakcji lotnych kwasów tłuszczowych (LKT), metanie (CH₄) i dwutlenku węgla (CO₂) w fazie gazowej. Dane te są zawarte w pliku **dane_eksp.txt**. Kolejne kolumny zawierają: czas [doba], C_{SM}, C_{LKT}, C_{CH₄}, C_{CO₂} [g/l]. Kinetykę reakcji kofermentacji opisano układem równań prowadzących się do postaci:

$$\begin{aligned}
 \frac{dy_1}{d\tau} &= -k_1 y_1 \\
 \frac{dy_2}{d\tau} &= k_2 y_1 - \frac{k_3}{F} y_2 \\
 \frac{dy_3}{d\tau} &= \frac{k_5 y_2}{F} \\
 \frac{dy_4}{d\tau} &= (k_1 - k_2) y_1 + (k_3 - k_5) \frac{y_2}{F},
 \end{aligned}
 \tag{9.1}$$

gdzie

$$F = k_4 + y_2 \quad (9.2)$$

oraz wektor $y = [C_{SM}, C_{LKT}, C_{CH_4}, C_{CO_2}]$ [g/l].

Należy dopasować współczynniki kinetyczne tego układu równań (od k_1 do k_5).

P9_1.m

```
%PROJEKT 9.1
%Identyfikacja współczynników kinetycznych
%układu reakcji biochemicznych
clc; clf
%zmienne globalne
global wartpocz przedzialy liczba dane czas t Y
%-----
format compact %Pisanie linijka pod linijką
%-----
%wczytywanie danych eksperymentalnych
plik=load('-ascii', 'dane_eksp.txt');
czas=plik(:,1);
dane=plik(:,2:size(plik,2));
clear plik;
%-----
%wartości początkowe dla układu równań
wartpocz=dane(1,:);
disp('Warunek początkowy - zawartości węgla [g/l]:');
disp(wartpocz);
%wartości początkowe poszukiwanych k dla optymalizacji
k= [1.83E-01; %K1 -> k(1)
    1.62E-01; %K3 -> k(2)
    3.20E+01; %K4 -> k(3)
    9.14E+01; %K5 -> k(4)
    1.90E+01]; %K9 -> k(5)
disp('Wartości początkowe poszukiwanych współczynników k :');
disp(k);
%ograniczenia dla poszukiwanych k w optymalizacji
przedzialy=[1.83E-021.83E+01; %K1 -> k(1)
            1.62E-022.00E-01; %K3 -> k(2)
            1.00E+004.20E+01; %K4 -> k(3)
            1.00E+011.23E+02; %K5 -> k(4)
            1.00E+002.50E+01]; %K9 -> k(5)
disp('Ograniczenia dla poszukiwanych współczynników :');
```

```

disp(przedzialy);
%liczba kolumn w danych
%to także liczba równań
liczba=size(dane,2);
disp(['Liczba równań : ',num2str(liczba)]);
%-----
%poszukiwanie wyniku za pomocą algorytmu optymalizacji
ileiteracji=100;           %maksymalna liczba iteracji
krokpoczątkowy=20;        %krok początkowy
tolerancja=0.001;         %minimalna odchyłka funkcji celu
disp('Optymalizacja:');
disp('Metoda Hooke-Jeevsa:');
disp(['  Maks. liczba iteracji   = ',num2str(ileiteracji)]);
disp(['  Krok początkowy        = ',num2str(krokpoczątkowy)]);
disp(['  Tolerancja funkcji celu = ',num2str(tolerancja)]);
disp(' ')
disp('Obliczenia....');
format long e %długie liczby, e-oznacza format inżynierski
wy-
nik=OptymHJ(k,przedzialy,krokpoczątkowy,ileiteracji,tolerancja);
%-----
%wyświetlenie wyniku
disp(' ')
disp('Otrzymane współczynniki:')
disp(wynik)
%-----
%rozwiązanie ukł. równań dla otrzymanych współczynników
podzial=40; %liczba rysowanych punktów
maksczas=czas(length(czas));
punkty=0:maksczas/podzial:maksczas;
k=wynik;
[t,Y] = ode15s(@uk_row,punkty,wartpocz);
%rysowanie kolejnych wykresów
figure(1)
hold on, grid on
plot(t,Y(:,1),'-b','LineWidth',2)
plot(t,Y(:,2),'--r','LineWidth',2)
plot(t,Y(:,3),'-.g','LineWidth',2)
plot(t,Y(:,4),'-m','LineWidth',2)
plot(czas,dane(:,1),'ob');
plot(czas,dane(:,2),'xr');
plot(czas,dane(:,3),'*g');
plot(czas,dane(:,4),'^m');

```

```

legend('SM','LKT','CH4','CO2');
xlabel('czas [doby]'); ylabel('Stężenie C, [g/l]');

%-----
warning on all % Włączenie komunikatów o uwagach
format loose %Wolna linijka między poleceniami
format short %'krótkie' wyświetlanie liczb
clear ileiteracji k tolerancja przedziały wartpocz krokpoczątko-
wy
clear podział makszczas punkty
%-----
%Zapisanie otrzymanych wyników
plik=[t,Y];
save ('wyniki.txt','plik', '-ascii')
clear plik
%-----
%KONIEC
disp(' ')
disp('Obliczenia zakończone')
disp('Użyj polecenia -wynik- aby obejrzeć obliczone współczynniki.')
disp('Wyniki symulacji zapisano w pliku wyniki.txt')
%-----

```

W pliku **f_celu.m** zapisana jest funkcja celu

f_celu.m

```

% Funkcja celu
% 1. Rozwiązuje, dla danych współczynników k,
%   układ równań różniczkowych z pliku uk_row
% 2. Zwraca sumę kwadratów różnic wartości obliczonych
%   z modelu i danych eksperymentalnych z pliku
% -----
function FC = f_celu(x)
global wartpocz liczba dane czas t Y k
k=x;
wynik=0;
% -----
% rozwiązanie ukł.równ.różn.
[t,Y] = ode15s(@uk_row,czas,wartpocz);
% -----
% obliczanie współczynnika jakości
for i=1:liczba

```

```

        for nr=1:length(t)
            wynik=wynik+(Y(nr,i)-dane(nr,i))^2;
        end
    end
% -----
FC=wynik/liczba;
end

```

W pliku **uk_row.m** zdefiniowany jest układ równań różniczkowych zwyczajnych tworzących model.

uk_row.m

```

% Układ równań różniczkowych:
% t          - czas
% y()        - funkcje
% dydt()     - pochodne
% k          - poszukiwane współczynniki

function dydt = uk_row(t,y)
global k
    F=k(4)+y(2);          %zmienna pomocnicza
    dydt = zeros(4,1);    %pusty wektor pochodnych
    dydt(1) = -k(1)*y(1);
    dydt(2) = k(2)*y(1)-k(3)*y(2)/F;
    dydt(3) = k(5)*y(2)/F;
    dydt(4) = (k(1) - k(2))*y(1) + (k(3) - k(5))*y(2)/F;
end

% y(1) -> S sucha masa
% y(2) -> A LKT
% y(3) -> M metan
% y(4) -> G dwutlenek węgla

```

Algorytm optymalizacyjny Hooke’a-Jeevsa jest w pliku **optymHJ.m**.

optymHJ.m

```

% Metoda optymalizacji Hooke-Jeevsa z ograniczeniami na zmienne
function OPHJ = OptymHJ(k,kogr,tau,maxit,eps)
% k      - poszukiwana macierz wartości optymalnych
% kogr   - ograniczenia związane z macierzą k
% tau    - wstępny krok iteracji
% maxit  - maksymalna liczba iteracji

```

```

% eps - tolerancja
% krok - funkcja przesuwająca o jeden krok w przód lub w tył
% sprawdz - funkcja sprawdzająca

global X1 WekX F1 ni
global Xogr

WekX=k;
Xogr=kogr;
N=size(WekX,1);
X0=WekX;
F2 = f_celu(WekX);
F0 = F2;    m=0;    ni=tau;    Czy=false;

tau1=ni;
%-----
while ((tau1>=eps) && (m<=maxit))
    %-----
    m=m+1; tau1=ni;
    for i=1:N
        krok(1,i);
        if ((F1<F2)&&(sprawdz(i,X1)))
            WekX(i)=X1(i);    F2=F1;
        else
            krok(-1,i);
            if ((F1<F2)&&(sprawdz(i,X1)))
                WekX(i)=X1(i);    F2=F1;
            end
        end
    end
end; % koniec pętli for
%-----
if (F2<F0)&&(sprawdz(i,X1))
    for i=1:N
        X1(i)=X0(i);
        X0(i)=WekX(i);
        WekX(i)=X0(i)+1.4*(WekX(i)-X1(i));
    end;
    F0=F2;
    F2=f_celu(WekX);
    ni=1.2*ni;    Czy=true;
elseif Czy
    WekX=X0;
    F2=F0;

```



```

        Czy=false;
    elseif tau1>eps
        ni=0.2*ni;
    end;
    disp(F2); %Wyświetlenie aktualnej wartości celu
end % koniec pętli while
%OPHJ=FunCelu(WekX);
OPHJ=WekX;
if m>maxit
    disp(['Przekroczono maksymalną liczbę iteracji = ',num2str(maxit)]);
else
    disp(['Nie przekroczono maksymalnej liczby iteracji = ',num2str(maxit)]);
    disp(['Liczba iteracji : ',num2str(m)]);
end
%end % koniec funkcji OPHJ
%=====

% Funkcja krok zmieniająca wartość
function krok(j,i)
global X1 WekX F1 ni
% j = +-1
    X1=WekX;
    X1(i)=WekX(i)+j*ni;
    F1=f_celu(X1);
%end; % koniec funkcji krok

%=====
function log=sprawdz(ind,~)
global Xogr WekX
    if (WekX(ind)<Xogr(ind,1)) || (WekX(ind)>Xogr(ind,2))
        log=false; % wartość nie mieści się między min i max
    else
        log=true; % wartość mieści się między min i max
    end
%end; % koniec funkcji sprawdz

```

Wyniki 9.1

```

Warunek początkowy - zawartości węgla [g/l]:
    5.0094
    0.3216

```

```

0
0
Wartości początkowe poszukiwanych współczynników k :
0.1830
0.1620
32.0000
91.4000
19.0000

```

```

Ograniczenia dla poszukiwanych współczynników :
0.0183    18.3000
0.0162    0.2000
1.0000    42.0000
10.0000   123.0000
1.0000    25.0000

```

Liczba równań : 4

Optymalizacja:

Metoda Hooke-Jeevsa:

```

Maks. liczba iteracji   = 100
Krok początkowy        = 20
Tolerancja funkcji celu = 0.001

```

Obliczenia....

```

1.317258799324742e+000
1.261888326703322e+000
1.261888326703322e+000
. . . . .
1.028600479824042e+000
1.031361944073055e+000

```

Nie przekroczono maksymalnej liczby iteracji = 100

Liczba iteracji : 58

Otrzymane współczynniki:

```

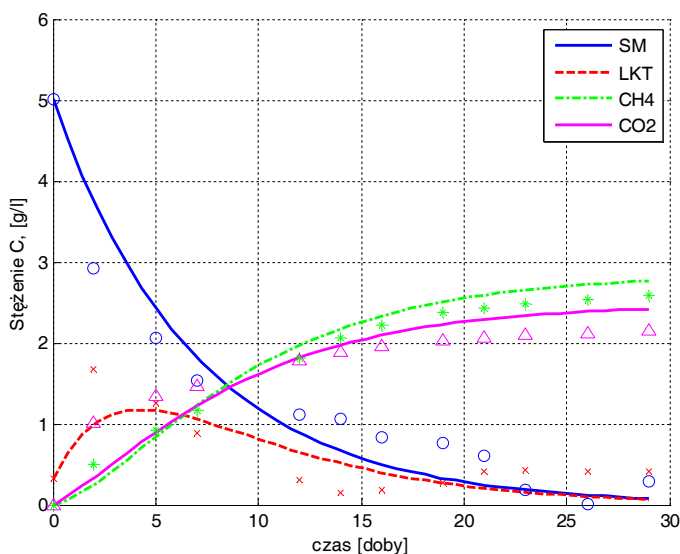
1.459173959601979e-001
1.229012484046848e-001
3.293295290989411e+001
1.120254913871319e+002
2.037053324959302e+001

```

Obliczenia zakończone

Użyj polecenia -wynik- aby obejrzeć obliczone współczynniki.

Wyniki symulacji zapisano w pliku wyniki.txt



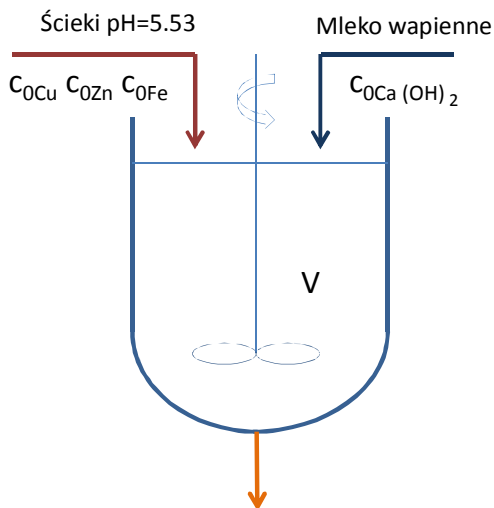
Rys. 9.1. Kinetyka zmian zawartości węgla w poszczególnych frakcjach

PROJEKT 9.2

Temat: Identyfikacja parametrów równań kinetycznych procesu oczyszczania ścieków metalurgii mokrej

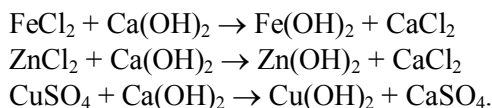
Toolbox: -

W warunkach przemysłowych wykonano następujący eksperyment (*niepublikowane dane*, 2008). Do zbiornika reakcyjnego (Rys. 9.2) o objętości czynnej $V = 15 \text{ m}^3$, wyposażonego w mieszadło szybkoobrotowe, przepompowano 10 m^3 ścieków przemysłowych pochodzących z procesów metalurgicznych, takich jak: trawienie walcówki stalowej w HCl , cynkowanie drutów przy użyciu HCl jako dotrawiacza i ZnCl_2 jako aktywatora powierzchni, oraz miedziowanie drutów w roztworze CuSO_4 . Ścieki te zawierają związki żelaza, cynku i miedzi w postaci przede wszystkim chlorków i siarczanów, a ich skład początkowy jest następujący: $c_{\text{Fe}} = 358 \text{ mg/dm}^3$, $c_{\text{Zn}} = 203 \text{ mg/dm}^3$, $c_{\text{Cu}} = 24 \text{ mg/dm}^3$ oraz $\text{pH} = 5.53$.



Rys. 9.2. Schemat reaktora okresowego w którym przeprowadzano reakcje

Następnie, przy włączonym mieszaniu, dodawano zawiesinę mleka wapiennego, zawierającą $258 \text{ mol/m}^3 \text{ Ca(OH)}_2$, tak aby uzyskać w zbiorniku $\text{pH} = 8.5$. Dodano 0.512 m^3 mleka wapiennego, co spowodowało zajście następujących reakcji chemicznych:



Wydzielone nierozpuszczalne wodorotlenki opadły na dno zbiornika w postaci osadu, natomiast ciecz nad osadem była poddawana analizie na obecność jonów ww. metali w odstępach 1, 2, 5, 10, 20 minut po zaprzestaniu mieszania.

Ponieważ rządowość powyższych reakcji jest identyczna, wystarczy napisać ogólne równanie bilansowe dla jednego ze składników np. Fe:

$$(m_{\text{Fe}})_{\text{wl}} - (m_{\text{Fe}})_{\text{wyl}} + Z_{\text{Fe}} = \frac{dM_{\text{Fe}}}{d\tau} \quad (9.3)$$

Reaktor okresowy charakteryzuje się tym, że w czasie przebiegu reakcji nie doprowadza się do niego ani nie odprowadza z niego żadnych strumieni masy, czyli

$$(m_{\text{Fe}})_{\text{wl}} = 0 \text{ oraz } (m_{\text{Fe}})_{\text{wyl}} = 0 \quad (9.4)$$

Masę żelaza w reaktorze, w dowolnym momencie czasu, można wyznaczyć jako iloczyn średniego stężenia molowego c_{Fe} w rozważanej chwili, objętości fazy reakcyjnej V oraz masy molowej N_{Fe}

$$M_{Fe} = N_{Fe} V c_{Fe} . \quad (9.5)$$

Szybkość akumulacji żelaza wynika ze zróżniczkowania powyższej zależności

$$\frac{dM_{Fe}}{d\tau} = N_{Fe} \frac{d(V c_{Fe})}{d\tau} = N_{Fe} \left(V \frac{dc_{Fe}}{d\tau} + c_{Fe} \frac{dV}{d\tau} \right) . \quad (9.6)$$

W tym konkretnym przypadku możemy przyjąć, że objętość fazy reakcyjnej $V = \text{const}$ w czasie procesu, zatem $dV/d\tau = 0$ i mamy

$$\frac{dM_{Fe}}{d\tau} = N_{Fe} V \frac{dc_{Fe}}{d\tau} . \quad (9.7)$$

Stężenie żelaza maleje w wyniku reakcji chemicznej, więc reprezentujący szybkość tego znikania upust masy Z_{Fe} ma znak minus. Masowa szybkość znikania żelaza liczona na całą objętość czynną reaktora jest równa iloczynowi całki z molowej szybkości reakcji po objętości V reaktora i masy molowej N_{Fe} . Iloczyn ten wyznacza natężenie upustu masy:

$$Z_{Fe} = - N_{Fe} \int_V r_{Fe} dV . \quad (9.8)$$

Szybkość reakcji możemy zapisać jako zależność

$$r_{Fe} = k_1 c_{Fe} c_{Ca(OH)_2} , \quad (9.9)$$

stąd natężenie upustu masy

$$Z_{Fe} = - N_{Fe} k_1 \int_V c_{Fe} c_{Ca(OH)_2} dV . \quad (9.10)$$

Przy dobrym mieszaniu zawartości reaktora stężenia c_{Fe} i $c_{Ca(OH)_2}$ są równomier-
nie rozłożone w całej objętości reaktora, a zatem po wyłączeniu ich przed znak
całki otrzymujemy natężenie upustu masy Z_{Fe}

$$Z_{Fe} = -N_{Fe} k_1 V c_{Fe} c_{Ca(OH)_2}. \quad (9.11)$$

Podstawiając uzyskane równania do bilansu ogólnego, otrzymujemy:

$$\frac{dc_{Fe}}{d\tau} = -k_1 c_{Fe} c_{Ca(OH)_2} \quad (9.12)$$

Jest to równanie różniczkowe opisujące zmianę stężenia Fe w reaktorze w czasie
przebiegu reakcji. Z lewej strony mamy szybkość akumulacji, z prawej upust
masy. Analogicznie dla Zn i Cu będziemy mieli równania:

$$\begin{aligned} \frac{dc_{Zn}}{d\tau} &= -k_2 c_{Zn} c_{Ca(OH)_2} \\ \frac{dc_{Cu}}{d\tau} &= -k_3 c_{Cu} c_{Ca(OH)_2} \end{aligned} \quad (9.13)$$

Zmienność stężenia $Ca(OH)_2$ w czasie przedstawia zależność:

$$\frac{dc_{Ca(OH)_2}}{d\tau} = - (k_1 c_{Fe} c_{Ca(OH)_2} + k_2 c_{Zn} c_{Ca(OH)_2} + k_3 c_{Cu} c_{Ca(OH)_2}), \quad (9.14)$$

czyli

$$\frac{dc_{Ca(OH)_2}}{d\tau} = -c_{Ca(OH)_2} (k_1 c_{Fe} + k_2 c_{Zn} + k_3 c_{Cu}). \quad (9.15)$$

Aby wyznaczyć przebiegi stężeń składników w reaktorze należy rozwiązać po-
wyższy układ równań, z warunkami początkowymi dla $\tau = 0$ $c_{Fe} = c_{0Fe}$,
 $c_{Zn} = c_{0Zn}$, $c_{Cu} = c_{0Cu}$ oraz $c_{Ca(OH)_2} = c_{0Ca(OH)_2}$.

Do rozwiązania układu bilansowych równań różniczkowych wykorzystano
solver równań różniczkowych ode15s oraz algorytm optymalizacyjny Hooke'a-
Jeevsa. Optymalizacja ma za zadanie identyfikację współczynników k w tym
modelu.

P9_2.m

```
%PROJEKT 9.2
%Identyfikacja stałych układu równań kinetyki chemicznej
clc; clear all; close all;
% zmienne globalne wykorzystywane w programie
global wartpocz przedzialy ilosc dane czas t Y
% -----
format compact %pisanie linijka pod linijką
warning off all % wyłączenie komunikatów o uwagach
% -----
% wczytywanie danych
plik=load('-ascii','Eksperyment.txt');
czas=plik(:,1);
dane=plik(:,2:size(plik,2));
clear plik;
% -----
% wartości początkowe dla układu równań
wartpocz=dane(1,:);
disp('Wartości początkowe dla rozw. równań różn.:');
disp(wartpocz);
% wartości początkowe poszukiwanych k dla optymalizacji
k=[ 0.0014    % k1
    0.0018    % k2
    0.00204]; % k3
disp('Wartości początkowe poszukiwanych współczynników k');
disp(k);
% ograniczenia dla poszukiwanych k w optymalizacji
przedzialy=[4e-3 4e+3
             6e-3 6e+3
             5e-3 5e+3];
disp('Ograniczenia dla poszukiwanych k w optymalizacji');
disp(przedzialy);
% liczba kolumn w danych
% jest to także liczba równań
ilosc=size(dane,2);
disp(['Liczba równań:',num2str(ilosc)]);
% -----
% poszukiwanie wyniku
ileiteracji=25; %maksymalna liczba iteracji
krokpoczatkowy=20; %krok początkowy
tolerancja=1e-6; % minimalna odchyłka
disp('Optymalizacja:')
```

```

disp('Metoda Hooke'a-Jeevsa:')
disp([' Maks. liczba iteracji=', num2str(ileiteracji)])
disp([' Krok początkowy      =', num2str(krokpoczątkowy)])
disp([' Tolerancja          =', num2str(tolerancja)])
disp('')
disp('Obliczenia.....')
format long e
wy-
nik=OptymHJ(k,przedziały,krokpoczątkowy,ileiteracji,tolerancja);

% -----
% wyświetlenie wyniku
disp('')
disp('Otrzymane współczynniki:');
disp(wynik);
% -----
% Rozwiązanie układu równań dla otrzymanego wyniku
podział=60; % liczba punktów
maksczas=czas(length(czas));
punkty=0:maksczas/podział:maksczas;
k=wynik;
[t,Y]=ode15s(@uk_row,punkty,wartpocz);

% rysowanie wykresów
figure(1)
hold on
grid on
plot(t,Y(:,1),'r')
plot(czas,dane(:,1),'ro')
plot(t,Y(:,2),'b')
plot(czas,dane(:,2),'b+')
plot(t,Y(:,3),'g')
plot(czas,dane(:,3),'g>')
plot(t,Y(:,4),'m')
xlabel('czas [s]')
ylabel('stężenie [mol/m3]')
legend('zmiana stężenia Fe wg modelu','doświadczalne stężenia
Fe','zmiana stężenia Zn wg modelu','doświadczalne stężenia
Zn','zmiana stężenia Cu wg modelu','doświadczalne stężenia
Cu','zmiana stężenia Ca(OH)2 wg modelu')
figure(2)

subplot(4,1,1)

```



```

hold on
grid on
plot(t,Y(:,1),'m')
plot(czas,dane(:,1),'ro')
xlabel('czas [s]')
ylabel('[mol/m3]')
legend('zmiana stężenia Fe wg modelu','doświadczalne stężenie Fe')
subplot(4,1,2)
hold on
grid on
plot(t,Y(:,2),'m')
plot(czas,dane(:,2),'bo')
xlabel('czas [s]')
ylabel('[mol/m3]')
legend('zmiana stężenia Zn wg modelu','doświadczalne stężenie Zn')
subplot(4,1,3)
hold on
grid on
plot(t,Y(:,3),'m')
plot(czas,dane(:,3),'go')
xlabel('czas [s]')
ylabel('[mol/m3]')
legend('zmiana stężenia Cu wg modelu','doświadczalne stężenie Cu')
subplot(4,1,4)
hold on
grid on
plot(t,Y(:,4),'m')
xlabel('czas [s]')
ylabel('[mol/m3]')
legend('zmiana stężenia Ca(OH)2 wg modelu')
%-----
warning on all %Włączenie komunikatów o uwagach
format loose
format short
clear ileiteracji k tolerancja przedzialy wartpocz krokpoczatkowy
clear podzial makszas punkty
% -----
% Zapisanie otrzymanych wyników
plik=[t,Y];

```

```
save('wyniki.txt','plik','-ascii');
clear plik;
% -----
```

Funkcja, w której zdefiniowano układ równań różniczkowych, nosi nazwę **uk_row.m**.

uk_row.m

```
% Równania różniczkowe
% t          - czas
% y()        - kolejne funkcje
% dydt()     - kolejne pochodne
% k          - współczynniki zmienne
% -----
function dydt=Fun(t,y)
global k dane czas wartpocz

dydt=zeros(4,1);
%y(4)=dane(1,4)-dane(1,1)-dane(1,2)-dane(1,3)+y(1)+y(2)+y(3);
dydt(1)=-k(1).*y(1).*y(4);
dydt(2)=-k(2).*y(2).*y(4);
dydt(3)=-k(3).*y(3).*y(4);
dydt(4)=-k(1).*y(1).*dane(1,4)-k(2).*y(2).*dane(1,4)-
k(3).*y(3).*...
    dane(1,4)+k(1).*y(1).*dane(1,1)+k(2).*y(2).*...
    dane(1,1)+k(3).*y(3).*dane(1,1)+k(1).*y(1).*...
    dane(1,2)+k(2).*y(2).*dane(1,2)+k(3).*y(3).*...
    dane(1,2)+k(1).*y(1).*dane(1,3)+k(2).*y(2).*...
    dane(1,3)+k(3).*y(3).*...
    dane(1,3)-k(1).*y(1).^2-k(2).*y(2).*y(1)-
k(3).*y(3).*y(1)-...
    k(1).*y(1).*y(2)-k(2).*y(2).^2-k(3).*y(3).*y(2)-...
    k(1).*y(1).*y(3)-k(2).*y(2).*y(3)-k(3).*y(3).^2;
```

Funkcja celu dla optymalizatora jest zapisana w pliku **f_celu.m**

f_celu.m

```
% Funkcja celu
% 1.Rozwiązuje dla danych współczynników k układ równań różniczkowych
%   umieszczony w pliku uk_row
```

```

% 2.Ocena na ile rozwiązanie różni się od danych pobranych w
pliku
% zgodnie ze wzorem
% 3.Zwraca współczynnik jakości określony jako sumę kwadratów
% różnic wartości obliczonych i danych z pliku

% -----
function FC=f_celu(x)
global wartpocz ilosc dane czas t Y k ;
k=x;
wynik=0;

% -----
% rozwiązanie układu równań różniczkowych
[t,Y]=ode15s(@uk_row,czas,wartpocz);
% -----

% obliczanie współczynnika jakości
for i=1:ilosc
    for nr=1:length(t)
        wynik=wynik+(Y(nr,i)-dane(nr,i)).^2;
    end
end
%-----

FC=wynik/ilosc;

```

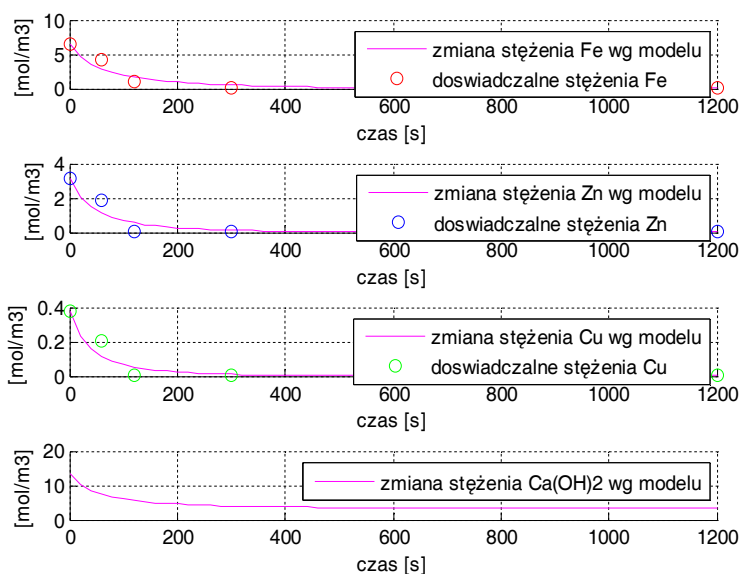
Wyniki 9.2

W wyniku działania programu otrzymano następujące stałe kinetyczne:

```

Otrzymane współczynniki:
1.4000000000000000e-003
1.8000000000000000e-003
2.0400000000000000e-003

```



Rys. 9.3. Zmiany poszczególnych stężeń wg modelu z dobranymi współczynnikami

PROJEKT 9.3

Temat: Identyfikacja współczynnika wnikania ciepła przy opływie ciała
Toolbox: **Optimization Toolbox**

W trakcie pomiaru α -kalorymetrem o kształcie kuli współczynnika wnikania ciepła w ośrodku o stałej temperaturze, mierzono w kolejnych odstępach czasu temperaturę kulki metalowej umieszczonej w tym ośrodku. Zaniedbano opory przewodzenia ciepła w kulce. Dane pomiarowe są zawarte w pliku **dane.txt**.

P9_3.m

```
%PROJEKT 9-3
```

```
%Identyfikacja współczynnika wnikania ciepła
```

```
clear all; close all; clc; memory; pack;
```

```
global tp Tp A m c tc tk t0
```

```
% W pomiarach mierzono temperaturę kulki metalowej
```

```
% w kolejnych odstępach czasu w środowisku o stałej temperaturze.
```

```
% Należy zidentyfikować współczynnik wnikania ciepła
```

```

%-----
% wczytanie danych doświadczalnych
load dane.txt
tp=dane(:,1);      % punkty pomiarowe czasu [s]
Tp=dane(:,2);      % punkty pomiarowe temperatury [C]
A=0.001;           % powierzchnia wymiany ciepła [m^2]
m=0.01;            % masa ciała [kg]
c=1.5;             % ciepło właściwe [kJ/kg*K]
tc=100;            % temperatura otoczenia [C]
t0=20;             % temperatura początkowa kulki [C]
tk=tp(length(tp)); % końcowy czas pomiaru [s]
% punkt startowy optymalizacji
a0=1;              % alfa startowe [kW/m^2*C]
ad=0.5;            % ograniczenie dolne
ah=2;              % ograniczenie górne
%--optymalizator--
options = optimset('Display','iter','Diagnostics','on');
aob = lsqnonlin(@f_celu,a0,ad,ah,options); % optymalizator
disp('alfa obl.= '); disp(aob);

%-----
[t,x]=calkuj(aob);
% wizualizacja wyników
figure(1)
plot(tp,Tp,'r*',t,x,'LineWidth',2)
legend('punkty dosw.','wynik modelu',4)
grid on
ylabel('temperatura [C]')
xlabel('czas [s]')
title('Identyfikacja alfa')

```

Plik **f_celu.m** zawiera zdefiniowaną funkcję celu

f_celu.m

```

function y=f_celu(a)
global tp Tp
% wywołanie integratora ODE dającego wartości Tob - temperatury
obliczonej
[t,x]=calkuj(a);
% tworzenie funkcji sklejanej temperatury obliczonej z modelu
pT = spline(t,x);
% obliczenie wartości temperatury dla czasów odpowiadających
punktom

```

```
% doświadczalnym
Tob = ppval(pT,tp);
% funkcja celu - wektor różnic wartości eksp. i obliczonych z
modelu
y=[Tp-Tob];
```

Plik **calkuj.m** to funkcja rozwiązująca równanie modelu

```
 calkuj.m
function [t,x]=calkuj(a)
global tk aloc t0
% lokalne alfa
aloc=a;
% integrator ODE
[t,x]=ode45(@uk_row,[0 tk],t0);
```

Plik **uk_row.m** zawiera zdefiniowane równanie różniczkowe

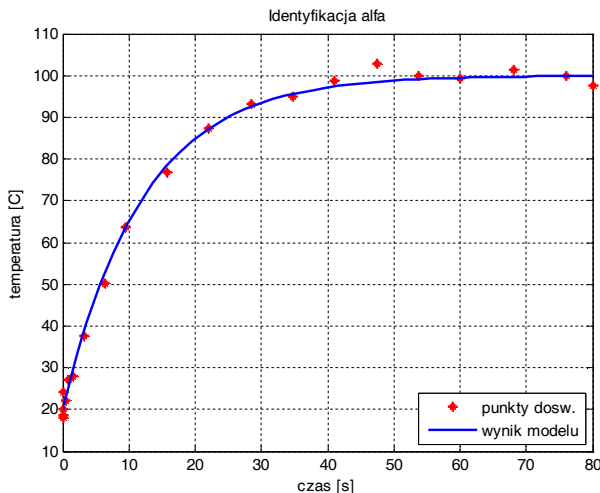
```
 uk_row.m
function dxdt=uk_row(t,x)
global aloc A m tc c
% równanie różniczkowe ODE
pT=[ ((alfa(aloc).*A)./(m.*c).*(tc-x(1)))];
dxdt=[ pT ];
```

Plik **alfa.m** to identyfikowany parametr

```
 alfa.m
function y=alfa(a)
% poszukiwany współczynnik alfa
y=a(1);
```

Wyniki 9.3

```
alfa obl. = 1.2248
```



Rys. 9.4. Wykres dopasowania modelu do punktów doświadczalnych ze zidentyfikowanym parametrem alfa

9.5. Identyfikacja współczynników modelu określonego równaniem różniczkowym cząstkowym

Identyfikacja przewodności cieplnej w procesach cieplnych bądź dyfuzyjności w procesach ruchu masy to główne typy spotykanych zadań tego rodzaju. Ich przykłady podano poniżej.

PROJEKT 9.4

Temat: Identyfikacja przewodności cieplnej ciała stałego
 Toolbox: **Optimization Toolbox**

W walec materiału izolacyjnego o promieniu R wbito, w kierunku równoległym do osi, termoelement, którego ostrze mierzy temperaturę na promieniu R_p . Walec umieszczono w środowisku o stałej temperaturze i znanym współczynniku wnikania ciepła. Zmiany temperatury w czasie podczas nagrzewania, zmierzone tym termoelementem są zapisane w pliku **dane.txt**. Należy na tej podstawie wyznaczyć przewodność cieplną tego materiału.

Posłużymy się, jak poprzednio, optymalizatorem **lsqnonlin**. Jako funkcja celu posłuży wektor różnic między wartością eksperymentalną temperatury a wartością uzyskaną z rozwiązania PDE przewodzenia ciepła w geometrii walcowej.

P9_4.m

```
%PROJEKT 9.4
%Identyfikacja przewodności cieplnej w walcu na podstawie
%punktowego pomiaru temperatury
clear all; close all; clc; memory; pack;
global tp Tk t0 R alfa tg cp ro dR n np

% wczytanie danych doświadczalnych
load dane.txt
tp=dane(:,1); % punkty pomiarowe czasu [s]
Tp=dane(:,2); % punkty pomiarowe temperatury [C]
% pozostałe dane
cp=0.85; % ciepło właściwe [kJ/kg*K]
tg=100; % temperatura otoczenia [C]
n=50; % liczba punktów dyskretyzacji [szt]
T0=Tp(1,1); % temperatura początkowa [C]
R=40e-3; % promień [m]
Rp=28e-3; % promień gdzie umieszczono termoelement [m]
alfa=1.2; % współczynnik wnikania ciepła [kW/kg*K]
ro=400; % gęstość materiału [kg/m^3]
dR=R/n; % szerokość komórki

%Obliczenia-----
np=round(Rp/R*n); % nr węzła pomiaru termopary
Tk=tp(length(tp)); % końcowy czas pomiaru [s]

% punkt startowy i ograniczenia optymalizacji
a0=1e-6; % dyfuzyjność cieplna
ad=1e-8; % ograniczenia od dołu
ah=1e-3; % ograniczenia od góry

% wywołanie optymalizatora
opt = optimset('Display','iter','Diagnostics','on','TolX',1e-9,'TolFun',1e-9);
aob = lsqnonlin(@f_celu,a0,ad,ah,opt);

% obliczenia końcowe-----
% współczynnik przewodzenia ciepła
lamb=aob*cp*ro;
lamb3=lamb*1e3; % [W/(m K)]
disp('Współczynnik przewodzenia ciepła [W/(m K)]')
disp(lamb3)
```



```

% obliczenia krzywej z modelu dla wartości znalezionej
[t,x]=calkuj(aob);
[wX,kX]=size(x);
Tob=x(:,np);

% wizualizacja wyników
figure(1)
plot(tp,Tp,'r*',t,Tob,'LineWidth',2);
legend('punkty dośw.','wynik modelu',4)
grid on
title('Identyfikacja parametru')
ylabel('temperatura [C]')
xlabel('czas [s]')
figure(2)
xi=linspace(0,R,n);
[w,k]=size(x);
yi=linspace(0,tk,w);
[XI,YI]=meshgrid(xi,yi);
surf(XI,YI,x)
title('Nieustalony ruch ciepła w walcu')
xlabel('odległość [m]')
ylabel('czas [s]')
zlabel('temperatura [C]')
hidden on
shading interp
alpha 0.8
figure(3)
[C,h] = contour(XI,x,YI,15);
set(h,'ShowText','on','LineWidth',2,'TextStep',get(h,'LevelStep')*2)
grid on
xlabel('odległość [m]')
ylabel('temperatura [C]')

```

Plik **calkuj.m** zawiera funkcję, która rozwiązuje układ równań ODE wynikający z dyskretyzacji metodą linii równania PDE przewodzenia ciepła (zob. Rozdz. 8).

calkuj.m

```

function [t,x]=calkuj(a)
global tk aloc T0 n
% lokalne a
aloc=a;
% warunek początkowy

```

```

for i=1:n
temp(i)=T0;
end
xp=temp;
% wywołanie solvera ODE
[t,x]=ode23tb(@uk_row,[0 tk],xp);

```

Plik **uk_row.m** to funkcja, która dokonuje dyskretyzacji przestrzennej i definiuje układ równań ODE.

uk_row.m

```

function dxdt=uk_row(t,T)
global aloc alfa tg cp ro dR n
% układ równań różniczkowych
a=ac(aloc);
lamb=a.*cp.*ro;
p(1)=2*a.*((T(2)-T(1))./(dR.^2));
for i=2:n-1
    p(i)=a.*(((T(i+1)-2*T(i)+T(i-1))./(dR.^2))+...
        (1/(i.*dR)).*(T(i+1)-T(i-1))./(2.*dR)));
end
wT=((2.*dR.*alfa)./lamb).*(T(n)-tg);
p(n)=a.*((2.*T(n-1)-2.*T(n)-wT)./(dR.^2)+(1/(n.*dR)).*(-
wT)./(2.*dR));
dxdt=[p'];

```

W pliku **f_celu.m** zawarta jest funkcja, która oblicza wektor różnic temperatur w punktach pomiarowych.

f_celu.m

```

function y=f_celu(a)
global tp Tp np
% wywołanie funkcji temperatury obliczonej
[t,x]=calkuj(a);
[wX,kX]=size(x);
% odczytanie wartości temperatury w miejscu instalacji termopary
Tob=x(:,np);
% dopasowanie funkcji sklejanej do temperatury obliczonej z modelu
pT = spline(t,Tob);
% obliczenie temperatury w punktach doświadczalnych
Tob = ppval(pT,tp);

```

```
% funkcja celu - różnica wartości eksperymentalnych i obliczo-
nych
y=(Tp-Tob);
```

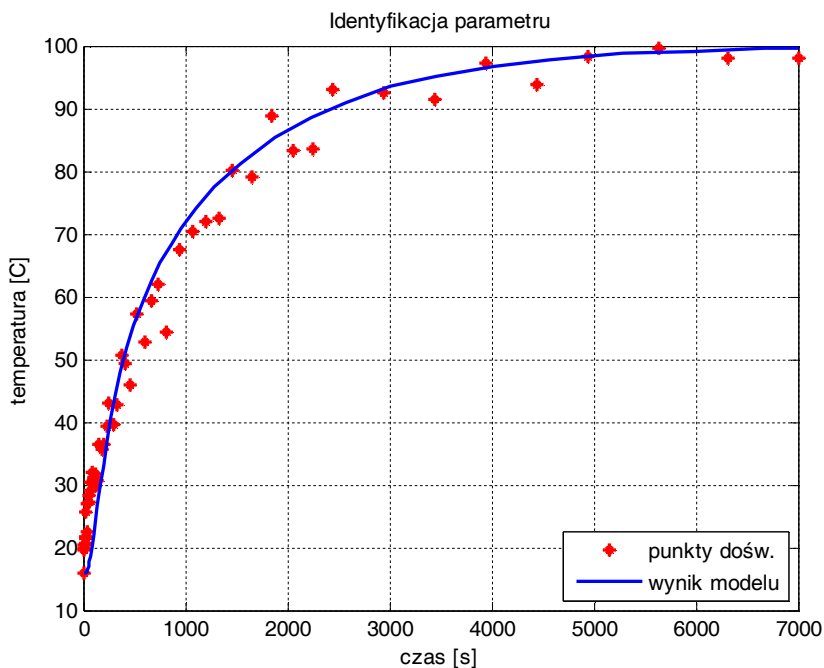
Funkcja dostarczająca aktualnej wartości poszukiwanego parametru to **ac.m**.

 **ac.m**

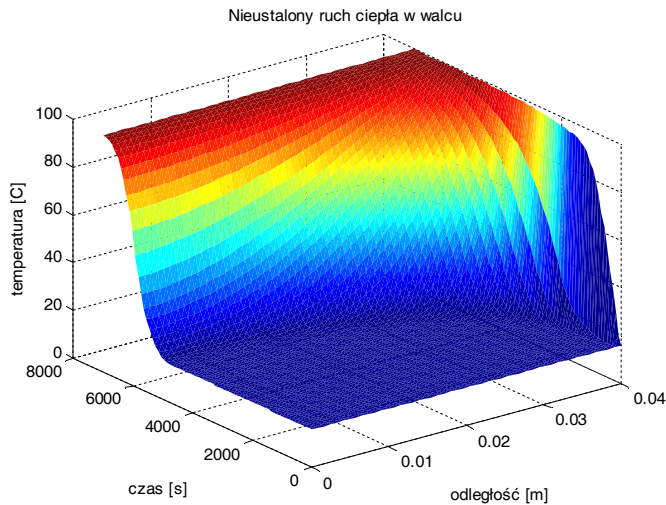
```
function y=ac(a)
% współczynnik przewodności temperaturowej
y=a(1);
```

Wyniki 9.4

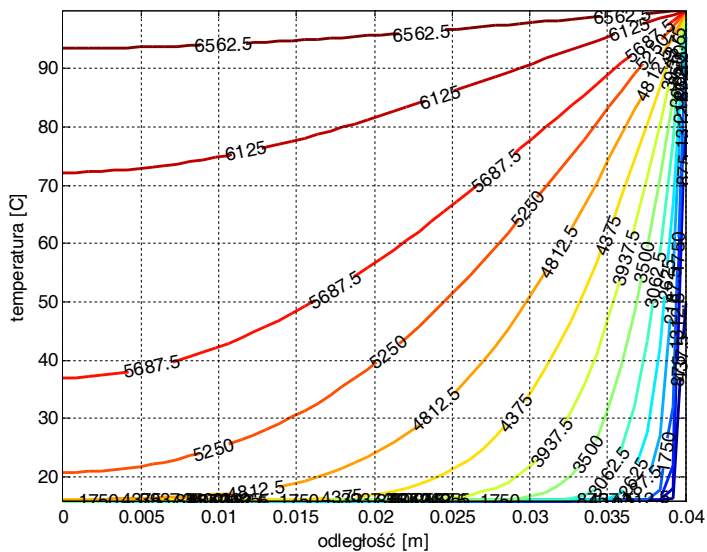
Współczynnik przewodzenia ciepła [W/(m K)]
0.0665



Rys. 9.5. Wykres dopasowania modelu do punktów doświadczalnych ze zidentyfikowanym parametrem



Rys. 9.6. Rozkład temperatur ze zidentyfikowanym parametrem



Rys. 9.7. Rozkład temperatur w czasie obliczony ze zidentyfikowanym parametrem

PROJEKT 9.5

Temat:	Identyfikacja dyfuzyjności efektywnej wilgoci w materiale suszonym o kształcie walca z uwzględnieniem skurczu – model we współrzędnych nieruchomych
Toolbox:	Optimization Toolbox

Wykorzystano model opisujący proces suszenia ciał ze skurczem zaproponowany przez *Pakowskiego, 2000*. Model ten jest zdefiniowany we współrzędnych nieruchomych. Oznacza to, że po każdym kroku czasowym jest obliczany nowy wymiar ciała kurczącego się i cała siatka obliczeniowa jest przeliczana na nowy rozmiar, tzn. obliczenia bilansu masy i ciepła są wykonywane w zmieniającej się geometrii. Przemieszczanie się powierzchni walca wywołuje dodatkowy przepływ konwekcyjny wilgoci w walcu. Model składa się z następujących równań:

Równanie konstytutywne opisujące skurcz suszarniczy:

$$\frac{\partial r}{\partial \tau} = \left(\frac{r}{r_0} \right) \frac{2\varepsilon}{\rho_{s0}} w_D. \quad (9.16)$$

Równanie bilansu masy z uwzględnionym skurczem:

$$\frac{\partial X}{\partial \tau} = D \left(\frac{\partial^2 X}{\partial r^2} + \frac{1}{r} \frac{\partial X}{\partial r} \right) + \frac{X}{r} \frac{\partial r}{\partial \tau}. \quad (9.17)$$

Równanie bilansu ciepła:

$$\frac{\partial T}{\partial \tau} = \frac{1}{\rho_s (c_L X + c_s)} \left[\lambda \left(\frac{\partial^2 T}{\partial r^2} + \frac{1}{r} \frac{\partial T}{\partial r} \right) + c_L T \frac{\partial X}{\partial \tau} \right] - \frac{c_L T}{c_L X + c_s} \frac{\partial X}{\partial \tau}. \quad (9.18)$$

Warunki brzegowe dla bilansu masy:

$$\left. \frac{\partial X}{\partial r} \right|_{r=0} = 0 \quad (9.19)$$

$$-D\rho_s \left. \frac{\partial X}{\partial r} \right|_{r=R} = w_D + (\rho_s X^i + \rho_g Y^i) \frac{\partial r}{\partial \tau}. \quad (9.20)$$

Warunki brzegowe dla bilansu ciepła:

$$\left. \frac{\partial T}{\partial r} \right|_{r=0} = 0 \quad (9.21)$$

$$\lambda \left. \frac{\partial T}{\partial r} \right|_{r=R} = \alpha (T_g - T) - \Delta h_v. \quad (9.22)$$

Zaproponowano następującą zależność dyfuzyjności od zawartości wilgoci i temperatury

$$D = D_1 X^{D_2} \exp\left(-\frac{D_3}{RT}\right). \quad (9.23)$$

Należy zidentyfikować współczynniki tego równania na podstawie wyników eksperymentalnych pomiaru zmian masy całkowitej próbki w kształcie walca i temperatury mierzonej w trzech punktach – w osi, na powierzchni i w połowie promienia walca, w trakcie procesu suszenia.

P9_5.m

%PROJEKT 9.5

%Identyfikacja dyfuzyjności efektywnej w walcu

%kurczącym się podczas suszenia

clear all; close all; clc; pack; memory;

global u Dt h Xmp Xob n t x R czas dr wm tp x0 T1 T2 T3 Tg Y epsilon

% dane doświadczalne-----

load dane.txt; % wczytanie danych doświadczalnych
% kinetyki zmiany masy kuli

Z1=dane;

tp=Z1(:,1); % czas [s]

mp=Z1(:,2); % masa [g]

[wm,km]=size(mp);

ms=2.2; % masa sucha [g]

Xmp=(mp(:,:)-ms)./ms; % wilgotność materiału z pomiarów [kg/kg]

load temp.txt; % wczytanie danych doświadczalnych

```

% kinetyki zmian temperatury
Z2=temp;
T1=Z2(:,1); % termopara na zewn.
T2=Z2(:,3); % termopara w środku promienia
T3=Z2(:,2); % termopara w osi
[wtz,ktz]=size(T3);

% geometria walca i tunelu
R=30e-3/2; % promień suszonego walca [m]
h=45e-3; % wysokość walca [m]
Dt=0.2; % średnica zastępcza tunelu [m]
% siatka dyskretyzacji
n=50; % ilość punktów dyskretyzacji [szt]
dr=R/n; % wymiar komórki [m]
% stan czynnika suszącego
Tg=50; % temperatura powietrza [C]
Y=0.004; % wilgotność powietrza suszącego [kg/kg]
u=1.5; % prędkość przepływu powietrza w tunelu [m/s]
% stan początkowy materiału
x0=Xmp(1,1); % wilgotność początkowa [kg/kg]

czas=tp(wm,1)+5; % czas trwania procesu [s]
epsilon=0.2481/3; % współczynnik skurczu liniowego [-]

% optymalizacja
D0=[1e-08; 0.03; 2500]; % wektor współczynników równania na Deff
Du=[1e-12; 1e-4; 1e3]; % zawartego w pliku Dif.m
Dh=[1e-7; 1e-1; 4e3];
tic
options = optimset('Display','iter','Diagnostics','on');
Dob = lsqnonlin(@f_celu,D0,Du,Dh,options); % optymalizator
Toc

%post-processing-----
[t,x]=calkuj(Dob);
%--średnia wilgotność obliczona z modelu
% całka metodą trapezów
zz=[linspace(0,n,n)']';
[wX,kX]=size(t);
Rob=x(:,n+n+1);
for j=1:wX
    R=Rob(j,1);
    dr=R/n;

```

```

r=zz.*dr;
s1=0; s2=0;
for i=2:n
    s1=s1+(1/R.^2).*((r(i).^2-r(i-1).^2).*(x(j,i)+x(j,i-1))./2) ;
    s2=s2+(1/R.^2).*((r(i).^2-r(i-1).^2).*(x(j,i+n)+x(j,i-1+n))./2) ;
end
Xob(j,1)=s1;
Tob(j,1)=s2;
End

%obliczenie z modelu temperatur w miejscach
%zainstalowania termoelementów
[wT,kT]=size(T1);
tT=linspace(0,czas,wT);
TT1=spline(tT,T1);
T1ob = ppval(TT1,tp);
TT2=spline(tT,T2);
T2ob = ppval(TT2,tp);
TT3=spline(tT,T3);
T3ob = ppval(TT3,tp);

%--wykresy 2D---
figure(1)
plot(t,Xob,tp,Xmp,'*')
title('Krzywa suszenia');
ylabel('X [kg/kg]'); xlabel('\tau [s]');
grid on

figure(2)
plot(t,x(:,n+n),'g',tp,T1ob,'g*',t,x(:,n+25),'b',tp,T2ob,'b+',t,x(:,n+1),'r',tp,T3ob,'ro')
title('Eksperymentalne i obliczone zmiany temperatury');
grid on
ylabel('T [C]'); xlabel('\tau [s]');

figure(3)
plot(t,Rob)
title('Kinetyka zmian promienia walca');
ylabel('R [m]'); xlabel('\tau [s]');
grid on

```



```

%--wykresy 3D-----
figure(4)
[w,k]=size(x);
for i=1:w
xi=linspace(0,x(i,k),n);
yi=linspace(0,t(i,1),1);
[XI,YI]=meshgrid(xi,yi);
ZI=x(i,1:n);
plot3(XI,YI,ZI)
hold on
end
grid on
view(0,0)
xlabel('r [m]'); ylabel('\tau [s]'); zlabel('X [kg/kg]')
title('Rozkłady promieniowe zawartości wilgoci');

figure(5)
for i=1:w
xi=linspace(0,x(i,k),n);
yi=linspace(0,t(i,1),1);
[XI,YI]=meshgrid(xi,yi);
ZI=x(i,n+1:2*n);
plot3(XI,YI,ZI,'r')
hold on
end
xlabel('R [m]'); ylabel('\tau [s]'); zlabel('T [C]')
title('Rozkłady promieniowe temperatury');
grid on
view(60,30)

```

Funkcja celu jest zdefiniowana w pliku **f_celu.m**

 **f_celu.m**

```

function y=f_celu(D)
% funkcja celu: różnica kwadratów
% wartości doświadczalnych i obliczonych
global Xmp Xob n t x R czas dr wm tp T1 T2 T3
[t,x]=calkuj(D);

%--średnia wilgotność obliczona z modelu
% całka metodą trapezów
zz=[linspace(0,n,n)]';
% r=zz.*dr;

```

```

[wX,kX]=size(t);
clear Xob
dr=R/n;
r=zz.*dr;
for j=1:wX
    s1=0;
    for i=2:n
        s1=s1+(1/R.^2).*((r(i).^2-r(i-1).^2).*(x(j,i)+x(j,i-1))./2) ;
    end
    Xob(j,1)=s1;
end

pX = spline(t,Xob);
Xob2 = ppval(pX,tp);

pT = spline(t,x(:,n+1));
Tob2 = ppval(pT,tp);

[wT,kT]=size(T1);
tT=linspace(0,czas,wT);
TT1=spline(tT,T1);
T1ob = ppval(TT1,tp);
TT2=spline(tT,T2);
T2ob = ppval(TT2,tp);
TT3=spline(tT,T3);
T3ob = ppval(TT3,tp);
% aby otrzymać odpowiednik średniej całkowej temperatury
% z danych pomiarowych wprowadzono odpowiednie wagi
w1=(1/4)^2; w2=(3/4)^2; w3=1-w2; w2=w2-w1;
Tpomiar=T1ob*w1+T2ob*w2+T3ob*w3;

y=[ abs(Xmp-Xob2)/Xmp(1)+0.2*abs(Tpomiar-Tob2)/Tpomiar(1)];
% Tak określona funkcja celu ma różne wagi dla różnicy
% zawartości wilgoci i różnicy temperatur gdyż pomiary
% temperatur są mniej pewne niż pomiar masy. Wartości różnic są
% znormalizowane i pozbawione znaku.

```

Plik **calkuj.m** to wywołanie solvera ODE zdyskretyzowanych równań modelu.

 **calkuj.m**

```

function [t,x]=calkuj(D)
global x0 n czas Dloc R

```

```

% warunki początkowe
% dla ruchu masy
Dloc=D;
for i=1:n
xpX(i)=x0;
end
% dla ruchu ciepła
for i=1:n
xpT(i)=20;
end
xp=[xpX xpT R]';
% solver ODE
[t,x]=ode15s(@uk_row,[0 czas],xp);

```

Plik zawierający zdyskretyzowane równania to **uk_row.m**.

 **uk_row.m**

```

function dxdt=uk_row(t,x)
global n R dr Tg Dloc Y epsilon
[wx,kx]=size(x);
RR=x(n+n+1);
dr=RR./n; % szerokość komórki [m]
%-----
% obliczam X średnie w każdym kroku czasowym
zz=[linspace(0,n,n)]';
r=zz.*dr;
s=0;
for i=2:n
    s=s+(1./(RR).^2).*((r(i).^2-r(i-1).^2).*(x(i)+x(i-1))./2) ;
end
Xob=s;
%-----
% skurcz w czasie
dRdt=-
R.*epsilon.*((x(n+n+1)./R).^2).*(2./(rom(Xob,x(n+n)).*R)).*ky((T
g+x(n+n))./2,RR).*(Yr(x(n),x(n+n))-Y);
% masa w czasie
pX(1)=Dif(Dloc,x(1),x(1+n)).*(2*(x(2)-
x(1))./(dr.^2))+Xob./x(n+n+1)).*dRdt;
for i=2:n-1
    pX(i)=Dif(Dloc,x(i),x(i+n)).*(((x(i+1)-2.*x(i)+x(i-
1))./(dr.^2))+1./(i*dr)).*((x(i+1)-x(i-
1))./(2*dr))+Xob./x(n+n+1)).*dRdt;

```

```

end
wD=ky((Tg+x(n+n))./2,RR).*(Yr(x(n),x(n+n))-Y);
S=-(x(n).*rom(x(n),x(n+n))-Y.*rop(Tg)).*dRdt;
wX=(2*dr./(Dif(Dloc,x(n),x(n+n)).*rom(x(n),x(n+n)))).*(wD+S);
pX(n)=Dif(Dloc,x(n),x(n+n)).*(((2.*x(n-1)-2.*x(n)-wX)./(dr.^2))-
((1./(i*dr)).*(wX./(2*dr))))+(Xob./x(n+n+1)).*dRdt;

%temperatura w czasie
pT(1)=(1./(rom(x(1),x(n+1)).*(cpw(x(n+1)).*x(1)+cs(x(1),x(n+1))))
)).*(lambd(x(n+1)).*(2.*(x(n+2)-
x(n+1))./(dr.^2))+pX(1).*cpw(x(n+1)).*x(n+1))+
((cpw(x(n+1)).*x(n+1))./(cpw(x(n+1)).*x(1)+cs(x(1),x(n+1))))).*pX
(1);
for i=2:n-1

pT(i)=(1./(rom(x(i),x(n+i)).*(cpw(x(n+i)).*x(i)+cs(x(i),x(n+i))))
)).*(lambd(x(n+i)).*(((x(i+n+1)-2*x(i+n)+x(i+n-
1))./(dr.^2))+1./(i*dr))*((x(i+n+1)-x(i+n-
1))./(2*dr))+pX(i).*cpw(x(n+i)).*x(n+i))+
((cpw(x(n+i)).*x(n+i))./(cpw(x(n+i)).*x(i)+cs(x(i),x(n+i))))).*pX
(i);
end

wT=(alfa((Tg+x(n+n))./2,RR).*(Tg-x(n+n))-hvw(x(n+n)).*wD-
hsorp(x(n),x(n+n)).*wD ).*2*dr./lambd(x(n+n));
pT(n)=(1./(rom(x(n),x(n+n)).*(cpw(x(n+n)).*x(n)+cs(x(n),x(n+n))))
)).*(lambd(x(n+n)).*(((2*x(n+n-1)-
2*x(n+n)+wT)./(dr.^2))+1./(n*dr))*((wT./(2*dr))+pX(n).*cpw(x(n+
n)).*x(n+n))+
((cpw(x(n+n)).*x(n+n))./(cpw(x(n+n)).*x(n)+cs(x(n),x(n+n))))).*pX
(n);

% -----
dxdt=[pX';pT';dRdt];

```

W pliku **dif.m** zawarto równanie na dyfuzyjność efektywną z poszukiwanymi współczynnikami.

Dif.m

```

function y=Dif(D,X,T)
%równanie na Deff
y=D(1).*X.^D(2).*exp(-D(3)./(T+273));

```

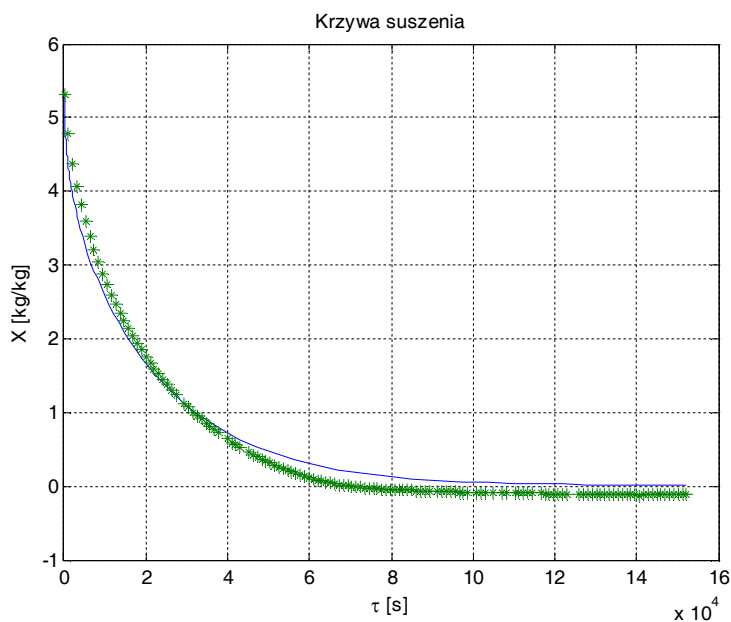
Pozostałe funkcje opisujące własności materiału, wilgoci oraz czynnika suszącego znajdują się w katalogu projektu.

Wyniki 9.5

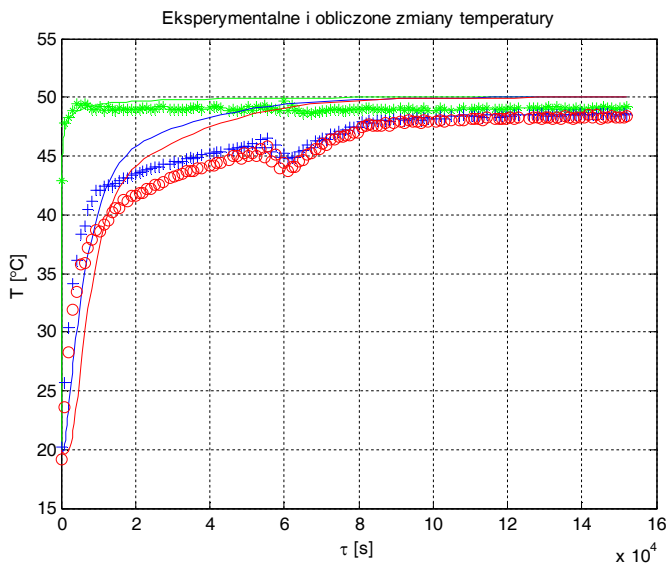
Umieszczenie zmiennych *tic* i *toc* wokół optymalizatora pozwoliło wyświetlić czas obliczeń:

Elapsed time is 267.331592 seconds.

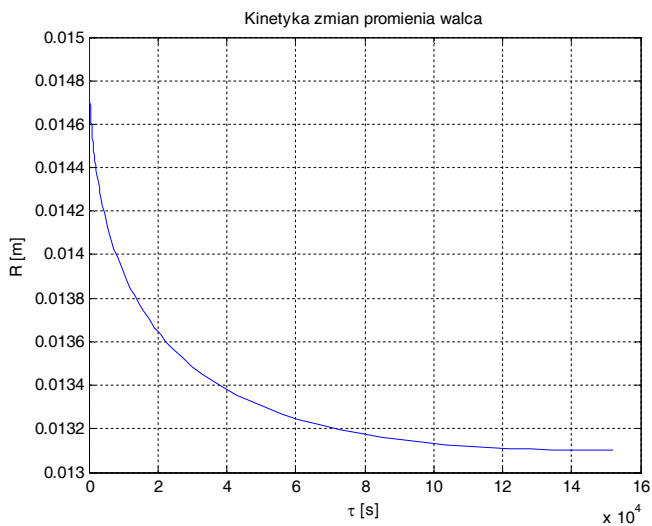
Współczynniki równania są zawarte w zmiennej *Dob* i wynoszą $8.5670\text{e-}8$, $2.1537\text{e-}4$ i $1.3393\text{e}3$. Jak widać z małej wartości wykładnika *Dob*(2) wpływ zawartości wilgoci na *Deff* jest pomijalnie mały, co dobrze świadczy o modelu. Pozostałe wyniki podano graficznie poniżej.



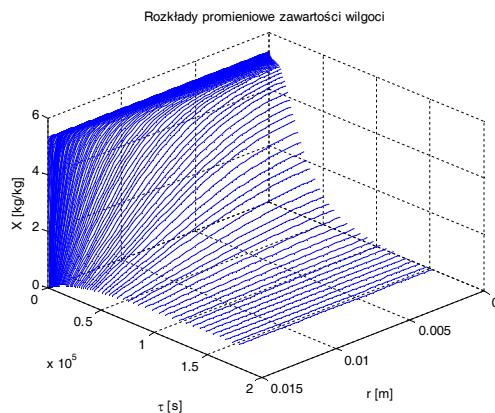
Rys. 9.7. Wykres zmian średniej wilgotności materiału w czasie



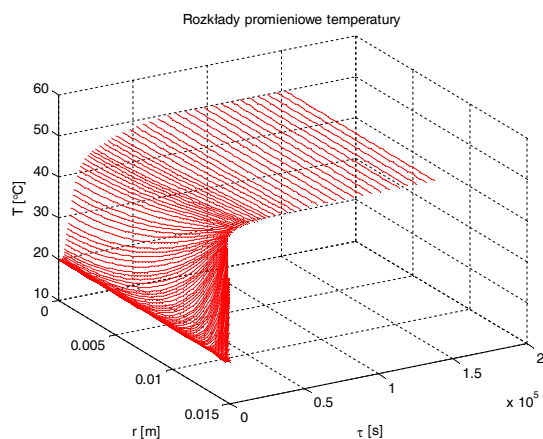
Rys. 9.8. Wyniki doświadczalne i doświadczalne modelu temperatury w punktach pomiarowych



Rys. 9.9. Wykres zmian promienia walca



Rys. 9.10. Wykres profili zawartości wilgoci



Rys. 9.11. Wykres rozkładu profili temperatury

Oznaczenia:

c	ciepło właściwe [J/(kg K)]
D	współczynnik dyfuzji [m^2/s]
Δh_v	ciepło parowania [J/(kg K)]
k_Y	współczynnik wnikanie ciepła [$\text{kg}/(\text{m}^2\text{s})$]
l	wymiar liniowy [m]
L	długość [m]
m	masa [kg]
r	promień [m]
τ	czas [s]

T	temperatura [$^{\circ}\text{C}$]
V	objętość [m^3]
w_D	szybkość suszenia [$\text{kg}/(\text{m}^2\text{s})$]
X	wilgotność [kg/kg]
Y	wilgotność powietrza [kg/kg]
z	współrzędna nieruchoma [m]

Greckie litery

α	współczynnik wnikania ciepła [$\text{W}/(\text{m}^2\text{K})$]
λ	przewodność cieplna [$\text{W}/(\text{m K})$]
ρ	gęstość [kg/m^3]
ε_L	współczynnik skurczu liniowego

Indeksy

g	gaz
L	ciecz
V	para
0	początkowy
*	równowaga

PROJEKT 9.6

Temat:	Identyfikacja dyfuzyjności efektywnej wilgoci w materiale suszonym w kształcie walca z uwzględnieniem skurczu – model we współrzędnych ruchomych
Toolbox:	Optimization Toolbox

Zastosowano tu model opisujący proces suszenia ciał ze skurczem zaproponowany przez *Kechaou i Roquesa, 1990*. Model ten zdefiniowany jest we współrzędnych ruchomych. Oznacza to, że w każdym kroku czasowym współrzędne śledzą rozmiar walca. Układ równań zdefiniowany w tych współrzędnych pozostaje taki jak we współrzędnych nieruchomych bez skurczu. Dopiero po zakończeniu obliczeń wyniki przelicza się na współrzędne nieruchome.

Z uwagi na niewielki opór cieplny, co wykazały obliczenia w projekcie 9.5, równanie ruchu ciepła jest napisane jak dla obiektu o parametrach skupionych. W tym modelu skurcz suszarniczy jest opisywany za pomocą współczynnika skurczu objętościowego. Szczegóły zawarte są w pracy *Pakowski, Adamski 2007*.

Poniżej zapisano równania modelu.

- Równanie bilansu masy:

Transformacja współrzędnych z nieruchomego układu współrzędnych do układu współrzędnych ruchomych związanych ze szkieletem ciała.

$$\frac{\rho_s}{\rho_{s0}} \left(\frac{\zeta}{z} \right) \frac{d\zeta}{dz} \quad (9.24)$$

Przekształcając równanie (22), otrzymujemy:

$$2\pi z dz \rho_s 1 = 2\pi \zeta d\zeta \rho_{s0} 1. \quad (9.25)$$

Wprowadzając przekształcenie współczynnika dyfuzji w postaci (24) w geometrii walcowej

$$D_\zeta = D_z \left(\frac{\rho_s}{\rho_{s0}} \right)^2 \frac{z}{\zeta}, \quad (9.26)$$

i podstawiając do równania liniowego skurczu objętościowego

$$\bar{V} = \bar{V}_0 (1 + \varepsilon_V X), \quad (9.27)$$

otrzymuje się

$$\left(\frac{\partial X}{\partial \tau} \right)_\zeta = D_\zeta \left(\frac{\partial^2 X}{\partial \zeta^2} + \frac{1}{\zeta} \frac{\partial X}{\partial \zeta} \right), \quad (9.28)$$

gdzie współczynnik dyfuzji dany jest wzorem

$$D_\zeta = \frac{D_z}{(1 + \varepsilon_V X)} \left(\frac{z}{\zeta} \right)^2. \quad (9.29)$$

Zakładając izometryczny skurcz, można zapisać równanie

$$\frac{\rho_s}{\rho_{s0}} = \frac{1}{1 + \varepsilon_V X}. \quad (9.30)$$

- Równanie bilansu ciepła:

$$\frac{dT}{d\tau} = \frac{1}{c_L X + c_S} \frac{A}{m_S} \left(\alpha (T_g - T) + w_D [(c_L - c_V) T - \Delta h_{v0}] \right). \quad (9.31)$$

Warunki brzegowe dla bilansu masy:

$$\left. \frac{\partial X}{\partial \zeta} \right|_{\zeta=0} = 0 \quad (9.32)$$

$$-D_\zeta \rho_{s0} \left. \frac{\partial X}{\partial \zeta} \right|_{\zeta=R_\zeta} = w_D. \quad (9.33)$$

Ponieważ wcześniej nie stwierdzono zależności dyfuzyjności od zawartości wilgoci, równanie na D_z przyjmie teraz postać

$$D_z = D_1 \exp\left(-\frac{D_2}{RT}\right). \quad (9.34)$$

P9_6.m

```
%PROJEKT 9.6
%Identyfikacja dyfuzyjności w walcu ze skurczem
%model w układzie z ruchomymi współrzędnymi
clear all; close all; clc; pack; memory;
global ms Rk A u Dt h Xmp Xob n t x R czas dr wm tp x0 T1 T2 T3
Tg Y epsilon

% dane doświadczalne kinetyki suszenia
load dane.txt;
Z1=dane;
tp=Z1(:,1); % czas [s]
mp=Z1(:,2); % masa [g]
[wm,km]=size(mp); % rozmiar macierzy mp
ms=2.2; % masa sucha
Xmp=(mp(:,:)-ms)./ms; % wilgotność materiału z pomiarów

% dane doświadczalne kinetyki zmian temperatury
load temp.txt;
```

```

Z2=temp;
T1=Z2(:,1); % termopara zewn.
T2=Z2(:,3); % termopara w środku promienia
T3=Z2(:,2); % termopara w osi
[wtz,ktz]=size(T3); % rozmiar macierzy T3

% geometria walca i tunelu
Rp=30e-3/2; % promień suszonego walca [m]
epsilon=0.2481; % współczynnik skurczu objętościowego
R=Rp./((1+epsilon.*Xmp(1,1)).^0.5);% promień suchy [m]
Rk=R;
h=45e-3; % wysokość walca [m]
Dt=0.2; % średnica zastępcza tunelu
A=h/R; % aspect ratio

% siatka dyskretyzacji
n=50; % liczba punktów dyskretyzacji [szt]
dr=R/n; % wymiar komórki [m]
% stan czynnika suszącego
Tg=50; % temperatura powietrza
Y=0.004; % wilgotność powietrza suszącego
u=1.5; % prędkość przepływu powietrza w tu-
nelu [m/s]
% stan początkowy materiału
x0=Xmp(1,1); % wilgotność początkowa [kgH2o/kgsm]
czas=tp(wm,1)+5; % czas trwania procesu [s]

% optymalizacja
D0=[1e-010; 3000];
Du=[1e-13; 1000];
Dh=[1e-3; 5000];
tic
options = optimset('Display','iter','Diagnostics','on');
Dob = lsqnonlin(@f_celu,D0,Du,Dh,options); % optymalizator
toc

% obróbka wyników
[t,x]=calkuj(Dob);
%--średnia wilgotność obliczona z modelu
% całka metodą trapezów
zz=[linspace(0,n,n)]';
r=zz.*dr;
[wX,kX]=size(t);

```

```

for j=1:wX
    s1=0;
    for i=2:n
        s1=s1+(1/R.^2).*((r(i).^2-r(i-1).^2).*(x(j,i)+x(j,i-1))./2);
    end
    Xob(j,1)=s1;
end
zzz=((1+epsilon.*Xob).^0.5).*R;
x=[x zzz];

%--wykres kinetyki suszenia
figure(1)
plot(tp,Xmp,'r*',t,Xob)
title('Krzywa kinetyki suszenia');
legend('pomiar','z modelu',1)
xlabel('t [s]'); ylabel('X [kg/kg]')

%--wykres przebiegu temperatury
figure(2)
[wT,kT]=size(T1);
tT=linspace(0,czas,wT);
TT1=spline(tT,T1); T1ob = ppval(TT1,tp);
TT2=spline(tT,T2); T2ob = ppval(TT2,tp);
TT3=spline(tT,T3); T3ob = ppval(TT3,tp);
% aby otrzymać odpowiednik średniej całkowitej temperatury
% z danych pomiarowych, wprowadzono odpowiednie wagi
w1=(1/4)^2; w2=(3/4)^2; w3=1-w2; w2=w2-w1;
Tsr=T1ob*w1+T2ob*w2+T3ob*w3;
plot(tp,Tsr,'r*',t,x(:,n+1))
title('Kinetyka zmian temperatury');
legend('pomiar temp.','z modelu',4)
xlabel('t [s]'); ylabel('T [C]')

%--wykres zmian promienia
figure(3)
plot(t,zzz)
title('Kinetyka skurczu')
xlabel('t [s]'); ylabel('r [m]')

%--wykres 3D rozkładów profili wilgotności
figure(4)
[w,k]=size(x);
for i=1:w

```

```

xi=linspace(0,x(i,k),n);
yi=linspace(0,t(i,1),1);
[XI,YI]=meshgrid(xi,yi);
ZI=x(i,1:n);
plot3(XI,YI,ZI)
hold on
end
grid on
view(0,0)
title('Profile zawartości wilgoci')
xlabel('R [m]'); ylabel('t [s]'); zlabel('X [kg/kg]')

```

Zasadniczej zmianie w porównaniu z projektem 9.5 uległ plik **uk_row.m**, w którym zdefiniowano równania obliczające rozkład wilgoci wewnątrz materiału i jego temperaturę

uk_row.m

```

function dxdt=uk_row(t,x)
% Zdyskretyzowane równanie PDE
global n R dr Tg Dloc Y ms Rk A

[wx,kx]=size(x);
% RR=x(n+n+1);
RR=Rk;
dr=RR./n; % szerokość komórki [m]
romp=rom(ms,x(n+1)).*((R.^3)./(RR.^3));

%-----
%--obliczam X średnie w każdym kroku czasowym
zz=[linspace(0,n,n)']';
r=zz.*dr;
s=0;
for i=2:n
    s=s+(1./(RR).^2).*((r(i).^2-r(i-1).^2).*(x(i)+x(i-1))./2) ;
end
Xob=s;

%-----
pX(1)=Dif(Dloc,x(1),x(1+n)).*(2*(x(2)-x(1))./(dr.^2));
for i=2:n-1
    pX(i)=Dif(Dloc,x(i),x(1+n)).*(((x(i+1)-2.*x(i)+x(i-1))./(dr.^2))+(1./(i*dr))*...
        ((x(i+1)-x(i-1))./(2*dr)));

```

```

end
wD=ky((Tg+x(n+1))./2,RR).*(Yr(x(n),x(n+1))-Y);
wX=(2*dr./(Dif(Dloc,x(n),x(n+1)).*rcomp)).*(wD);
pX(n)=Dif(Dloc,x(n),x(1+n)).*(((2.*x(n-1)-2.*x(n)-wX)./(dr.^2))-
((1./(i*dr)).*...
(wX./(2*dr))));

%pochodną temperatury liczono jak dla obiektu o parametrach sku-
pionych
pT=(1./((cpw(x(n+1)).*Xob+cs(ms,x(n+1)))).*
((2*pi*A*RR.^2)./ms).*(alfa((Tg+x(n+1))./2,RR).*...
(Tg-x(n+1))-wD.*((cpw(x(n+1))-
cpp(x(n+1))).*x(n+1)+hvw(0))+hsorp(ms,x(n+1))));

dxdt=[pX';pT'];

```

Plik **Dif.m** został zmodyfikowany. Zawiera on równanie na współczynnik dyfuzji zależny tylko od temperatury oraz szukane parametry tego równania.

Dif.m

```

function y=Dif(D,X,T)
y=D(1).*exp(-D(2)./(T+273));

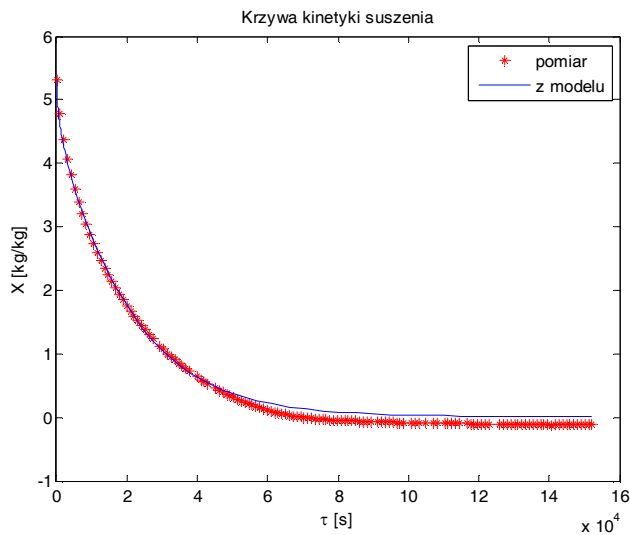
```

Wyniki 9.6

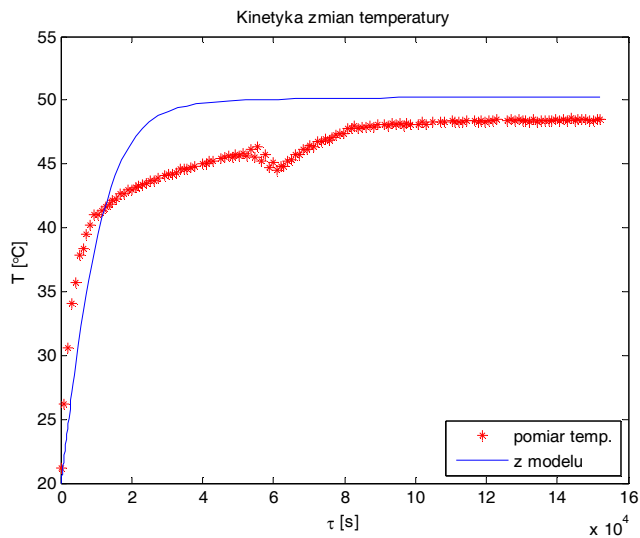
Rezygnacja z rozpatrywania przewodzenia ciepła w materiale poskutkowała znacznym zmniejszeniem czasu obliczeń:

```
Elapsed time is 15.403837 seconds.
```

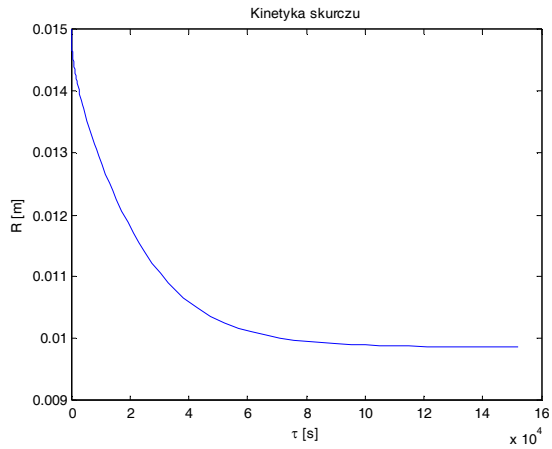
Uzyskano następujące współczynniki równania $Dob(1) = 7.1718e-6$ i $Dob(2) = 2.9074e3$. Pozostałe wyniki podano graficznie poniżej.



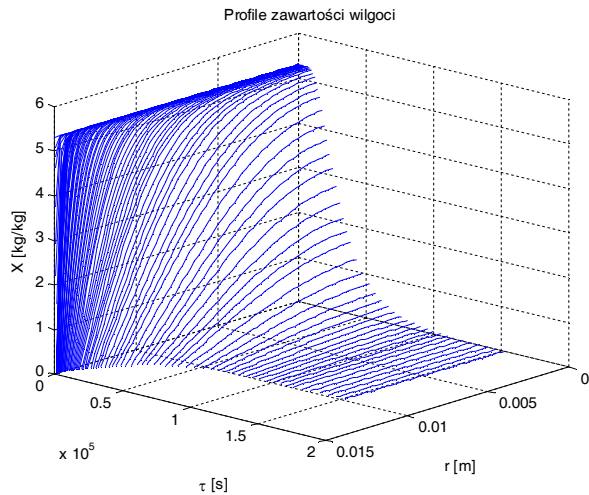
Rys. 9.2. Wykres zmian średniej wilgotności materiału w czasie



Rys. 9.12. Wykres zmian średniej temperatury w czasie



Rys. 9.13. Wykres zmian promienia walca



Rys. 9.14. Wykres profili zawartości wilgoci

Oznaczenia

c	ciepło właściwe [J/kg K]
D	współczynnik dyfuzji [m^2/s]
Δh_{V0}	ciepło parowania [J/kg]
k_Y	współczynnik wnikania masy [$\text{kg}/\text{m}^2\text{s}$]
l	wymiar liniowy [m]
L	długość [m]

m	masa [kg]
r	promień [m]
τ	czas [s]
T	temperatura [$^{\circ}\text{C}$]
V	objętość [m^3]
w_D	szybkość suszenia [$\text{kg}/\text{m}^2\text{s}$]
X	wilgotność [kg/kg]
Y	wilgotność powietrza [kg/kg]
z	współrzędna nieruchoma [m]

Indeksy

g	gaz
L	ciecz
V	para
0	początkowy
*	równowaga

LITERATURA

Kechaou N. Roques M., 1990, A model for convective drying on nonporous shrinking spheres, IDS'90, Vol. 1, 369-373.

Niepublikowane dane przemysłowe, 2008.

Quateroni A., Sacco R., Saleri F., „Numerical Mathematics”, ser. Texts in Apl. Math., No. 37, Springer, 2007.

Pakowski Z., „DryPAK – software for psychrometric and drying computations V. 3.6”, Omnicon, Łódź 2000.

Pakowski Z., Adamski R., The Comparison of Two Models of Convective Dryig of Shrinking Materials using Apple Tissue as an Example, Drying Techn., 2007, 25, 2-7, 1139-1147.

Sosnowski P., Klepacz-Smółka A., Kaczorek K., Ledakowicz S., 2008, Kinetic investigations of methane co-fermentation of sewage sludge and organic fraction of municipal solid wastes, Bioresource Technol., 99, 5731-5737.

Taler J., Duda P., „Rozwiązywanie prostych i odwrotnych zagadnień przewodzenia ciepła”, WNT, Warszawa 2003.

10. SPIS LITERATURY

- Balcerzak J., 2005, „Zmienność sorpcji wody przez folie chitozanowe wraz z temperaturą”, praca magisterska, Wydział Inżynierii Procesowej i Ochrony Środowiska PŁ.
- Brzózka J., Dobczyński L., Programowanie w Matlab, Wyd. Mikom, Warszawa 1998.
- Cady M.P., Trapp C.A., „A Mathcad Primer for Physical Chemistry”, Oxford Univ. Press, 1999, s. 172-173.
- Chen N.H., 1979, An Explicit Equation for Friction Factor in Pipe, Ind.&Eng. Chem. Fundam., 18, s. 216-231.
- Crank J., „The Mathematics of Diffusion”, Clarendon Press, Oxford, 1956.
- Finlayson B., Nonlinear Analysis in Chemical Engineering, McGraw-Hill 1980.
- Incropera F.P., Hewitt D.P., Bergman T.L., Lavine A.S., Fundamentals of Heat and Mass Transfer, J.Wiley, 2007, s. 212-218.
- Kechaou N. Roques M., 1990, A model for convective drying on nonporous shrinking spheres, IDS'90, Vol. 1, 369-373
- Krishna R., Taylor R., “Multicomponent Mass Transfer”, J. Wiley & Sons, 1993.
- Krzystek L., „Stechiometria i kinetyka bioprocessów”, Wyd. PŁ, Łódź 2010.
- Kudra T. (red.), „Zbiór zadań z podstaw teoretycznych inżynierii chemicznej i procesowej”, WNT, Warszawa 1985.
- Moler C., Numerical Computations with MATLAB, SIAM, 2004, rozdz. 7, s.19
- Myers A.L., Seider W.D., „Obliczenia komputerowe w inżynierii chemicznej”, Seria: Inżynieria chemiczna, WNT, 1979, s. 225.
- Niepublikowane dane przemysłowe, 2008.
- Orlik M., “Reakcje oscylacyjne porządek i chaos” WNT, Warszawa 1996.
- Ott E., „Chaos w układach dynamicznych”, WNT, Warszawa 1997, s. 20.
- Pakowski Z., 1986, Computer drawing of enthalpy-humidity charts, Hungarian Journal of Industrial Chemistry, Veszprem, Vol. 14, pp. 225-236.
- Pakowski Z., „Stany stacjonarne w procesie odparowywania kropli wieloskładnikowych do strumienia inerty”, PŁ, 1994.

- Pakowski Z., „dryPAK v.3.6 Program for psychrometric and drying calculations”, OMNIKON, Łódź 2000.
- Pakowski Z., „Projektowanie suszarek do suszenia parą przegrzaną”, Wyd. PŁ, Łódź 2012.
- Pakowski Z., *Procesy dyfuzyjne i aparaty (materiały do wykładów)*, PŁ, 2008-2014.
- Pakowski Z., Adamski R., *The Comparison of Two Models of Convective Drying of Shrinking Materials using Apple Tissue as an Example*, *Drying Techn.*, 2007, 25, 2-7, 1139-1147.
- Pakowski Z., Adamski R., *On prediction of the drying rate in superheated steam drying process*, *Drying Technology*, 2011, 29, 1492-1498.
- Pakowski, Z., B. Krupińska and R. Adamski, 2005, *Water activity of energetic variety of wood in a wide temperature range applicable to both superheated steam and air drying*, 7th World Congress of Chemical Engineering, Glasgow.
- Pakowski Z., Adamski R., Kwapisz S., *Effective diffusivity of moisture in low rank coal during superheated steam drying at atmospheric pressure*, *Chem. Eng.&Proc.* 2011, 32(4/2), 43-51.
- Pakowski Z. Druzdziel A. Drwiega J., 2002. *Validation of an expanding superheated steam flash dryer for cut tobacco based on processing data*. IDS 2002, Beijing, Vol. B, 828-836.
- Pakowski Z., Głębowski M., „Symulacja procesów inżynierii chemicznej”, Wyd. PŁ, 2001.
- Pakowski Z., Głębowski M., Adamski R., 2003, *Mathcad vs. MATLAB in the curriculum of chemical engineering at the Technical University of Łódź*, *Process Integration and Modeling of Chromatography Processes Conf.*, Rzeszów
- Pratap R., *MATLAB 7 dla naukowców i inżynierów*, PWN, Warszawa 2007.
- Quateroni A., Sacco R., Saleri F., „Numerical Mathematics”, ser. *Texts in Apl. Math.*, No. 37, Springer, 2007.
- Ramirez W.F., *Computational Methods for Process Simulation*, 2nd ed., Butterworth-Heinemann, 1997.
- Serwiński M., *Zasady inżynierii chemicznej*, WNT, Warszawa 1971, s. 509.

- Sosnowski P., Klepacz-Smółka. A., Kaczorek K., Ledakowicz S., 2008, *Kinetic investigations of methane co-fermentation of sewage sludge and organic fraction of municipal solid wastes*, *Bioresource Technol.*, 99, 5731-5737.
- Stachurski M., *Metody numeryczne w programie MATLAB*, Wyd. Mikom, Warszawa 2003.
- Szustalewicz A., „*Matematyka obliczeniowa*”, *Mat. wykł.*, *Uniw. Wrocł.*, 2002/2003, w. 11.
- Taler J., Duda P., „*Rozwiązywanie prostych i odwrotnych zagadnień przewodzenia ciepła*”, *WNT, Warszawa* 2003.
- Tabiś B., „*Zasady inżynierii reaktorów chemicznych*”, *WNT, Warszawa* 2000, s. 136-137.
- Walas S.M., „*Modeling with Differential Equations in Chemical Engineering*”, *Butterworth-Heinemann*, 1991.
- Wesselingh J.A., R. Krishna, „*Mass Transfer in Multicomponent Mixtures*”, *VSSD, Delft*, 2006.
- Zaleski T., *Metody algebry liniowej w inżynierii procesowej*, *PWN, Warszawa*, 1987.
- Zalewski A., Cegiela R., *Matlab – obliczenia numeryczne i ich zastosowania*, *Wyd. Nakom, Poznań* 2003.

W książce zaprezentowano możliwości szerokiego wykorzystania uniwersalnego programu do obliczeń numerycznych MATLAB w dziedzinie inżynierii chemicznej i procesowej.

Choć obecnie wiele obliczeń inżynierskich, takich jak np. symulacja instalacji chemicznych, wykonuje się rutynowo przy użyciu komercyjnego oprogramowania, pozostaje obszar badań i rozwoju, gdzie konieczne jest indywidualne podejście. W takich przypadkach MATLAB jest niezastąpiony.

Przedstawiono podstawowe właściwości MATLABa, ilustrując je przykładami z zakresu inżynierii chemicznej i procesowej. Książka zawiera 43 projekty będące działającymi programami wraz z systemem plików pomocniczych, ilustrującymi takie zagadnienia, jak: prezentacja graficzna wyników, aproksymacja danych eksperymentalnych, rozwiązywanie układów równań, rozwiązywanie równań różniczkowych zwyczajnych i cząstkowych czy identyfikacja parametrów.