Instytut Sterowania i Systemów Informatycznych
Wydział Informatyki, Elektrotechniki i Automatyki
Uniwersytet Zielonogórski

Laboratorium Metod Numerycznych

Informatyka/Elektrotechnika[1]

Spis treści

[1] Zbiór zadań dotyczy kierunku informatyka i elektrotechnika w trybie stacjonarnym (studia dzienne) oraz w trybie niestacjonarnym (studia zaoczne).

1 Wstęp do Matlaba

Motto:
Matlab przyjacielem każdego studenta.

1.1 Materiał omawiany na laboratorium

Materiał laboratorium Metod Numerycznych obejmuje następujące zagadnienia:

1. Wstęp, plan zajęć na laboratorium podstawowe informacje o Matlabie
2. Obliczenia, operacje elementarne na macierzach
3. Grafika 2D i 3D w Matlabie
4. Programowanie w Matlabie, m-pliki, m-funkcje.
5. COLLOQUIUM 1
6. Rozwiązywanie równań nieliniowych
7. Zagadnienia algebry liniowej
8. COLLOQUIUM 2
9. Zagadnienia interpolacji
10. Zagadnienia aproksymacji
11. COLLOQUIUM 3
12. Całkowanie numeryczne
13. COLLOQUIUM 4 – podsumowanie, wpisy do indeksów

Końcowa ocena z zaliczenia laboratorium to średnia ocen z poszczególnych colloquium. W przypadku braku oceny z colloquium student jest zobowiązany do zaliczenia sprawdzianu na następnych zajęciach.

1.2 Literatura

Literatura dotyczącą metod numerycznych jest bogata. Oto kilka pozycji:

1. Metody numeryczne, Fortuna Zenon, Macukow Bohdan, Wąsowski Janusz, WNT, Warszawa, 1995
2. Algorytmy numeryczne, Kazimierz Wanat, Gliwice, Helion, 1994
3. Metody numeryczne, Ake Bjärck, Germund Dahlquist, Warszawa, PWN, 1987
4. Metody numeryczne, Jerzy Klamka i in., Gliwice : Politechnika Śląska, 1998

O programie Matlab także napisano kilka książek. Kilka pozycji oferuje wydawnictwo Mikom (łącznie sześć pozycji) i Helion:

1. MATLAB i Simulink. Poradnik użytkownika. Wydanie II, Bogumiła Mrozek, Zbigniew Mrozek, 2004
2. Programowanie w MATLAB, Jerzy Brzózka, Lech Dorobczyński, Warszawa 1998, wydanie I
3. Matlab. Ćwiczenia z ..., Przykłady i zadania, Anna Kamińska, Beata Pańczyk, Warszawa 2002, wydanie I
4. Wykresy i obiekty graficzne w MATLAB, Wiesława Regel, Warszawa 2003, wydanie I
5. Metody numeryczne w programie Matlab, Marcin Stachurski, Warszawa 2003, wydanie I
6. Obliczenia symboliczne i numeryczne w Matlab, Wieslawa Regel, Warszawa 2003, wydanie I

Jednak wszystkie informacje o Matlab'ie można odszukać w samym programie. Posiada on bardzo dobrą dokumentację z której trzeba korzystać!

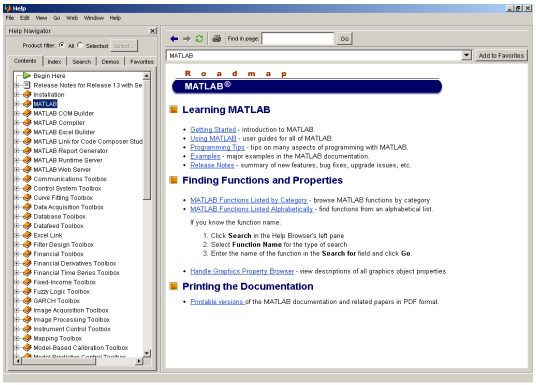
1.3 Kilka uwag o programie Matlab

Poniższe uwagi mają charakter informacyjny nie wyczerpują dokładnie całości zagadnień. Dlatego warto sięgnąć do dokumentacji wbudowanej do programu Matlab oraz eksperymentowania[2].

Program Matlab to jeden z lepszych pakietów zorientowanych przede wszystkim na obliczenia numeryczne. Zaletą Matlaba jest możliwość interaktywnej pracy z konsoli, gdzie można wydawać pojedyncze polecenia. Istnieje też możliwość tworzenie skryptów oraz dodawanie nowych funkcji. Zaletą Matlab'a jest także duży pakiet gotowych do wykorzystania funkcji np.: do tworzenia sieci neuronowych, czy pakiet Simulink. Tego rodzaju pakiety nazywa się też toolbox'ami. Ważnym aspektem jest także wydajność numeryczna. Wcześniejsze wersje Matlab'a nie oferowały pod tym względem dużych możliwości. Jednak od wersji 6.5 Matlab dzięki bibliotekom Atlas[3] może zaoferować wysoką wydajność.

1.4 Pomoc programu Matlab

Program Matlab posiada bardzo bogatą dokumentację z licznymi przykładami. Z pomocy możemy korzystać na dwa sposoby: za pomocą konsoli i polecenia **help**, bądź z interfejsu graficznego programu. Dokumentacja została zgromadzone w opcji *Help/Full Product Family Help* i jest to główny zbiór gdzie znajdują się informacje o toolbox'ach i funkcjach w nich zgromadzonych. Po wybraniu tej opcji pokaże się okno podobne do okna z rysunku 1 Korzystanie z tej formy pomocy jest charakterystyczne dla aplikacji okienkowych. Cała struktura pakietu



Rysunek 1: Główne okno pomocy programu Matlab

Matlab jak widać na rysunku 1 została przedstawiona w formie drzewa. Upraszcza to dostęp do informacji o poszczególnych toolbox'ach. Podczas wyszukiwania informacji przydają się zakładki *Index* – zostały tam zgromadzone wszystkie słowa kluczowe. Wpisując słowa system zawęża listę słów rozpoczynających się od podanej przez nas frazy. Okno pomocu posiada też zakładkę *Favorites* możemy w niej umieszczać odniesienia do najbardziej interesujących nas zagadnień. Wystarczy w drzewie Contents wybrać nas potrzebne zagadnienia i poprzez menu kontekstowe (prawy przycisk myszy) wybrać opcję *Add to Favorites*.

System pomocy Matlab'a pozwala także na przeszukiwanie dokumentacji elektroniczej. Służy do tego zakładka *Search* w oknie pomocy. Możemy przeszukiwać dokumentację na kilka sposób:

- opcja Full Text – przeszukuje treść dostępnych dokumentów pomocy
- opcja Document Titles – umożliwia przeszukiwanie wśród tytułów zgromadzonych dokumentów
- opcja Function Name – szuka podanej frazy w bazie nazw funkcji dostępnych w Matlabie
- opcja Online Knownledge Base – przeszukiwanie zasobów sieciowych

[2] Choć przy funkcjach obsługi systemów plików np.: **delete** oraz **rmdir** zalecana jest daleko idąca powściągliwość.

[3] Pakiet Atlas – darmowy zestaw procedur numerycznych dostępnych pod adresem: http://math-atlas.sourceforge.net/. Nie jest to jedyna darmowa biblioteka z jakiej korzysta Matlab innym jest zbiór metod numerycznych napisanych w Fortranie o nazwie LAPACK czy BLAS.

Oprócz wygodnego graficznego systemu pomocy Matlab oferuje także specjalne polecenie **help**. Wszelkie informacje jakie uzyskamy są wyświetlane bezpośrednio w konsoli programu. Uzyskanie informacji nt. jakiegoś polecenia jest następujące np.:

```
>> help det
```

Po wydaniu tego polecenie otrzymamy na ekranie konsoli informacje na temat podanej funkcji. Gdy pomoc dla danego polecenia nie będzie dostępna to Matlab wyświetli stosowny komunikat jak w poniższym przykładzie:

```
>> help fid2

No help comments found in fid2.m.
```

Jednak gdy dana funkcja rzeczywiście nie istnieje to otrzymamy inny komunikat:

```
>> help fff

fff.m not found.
```

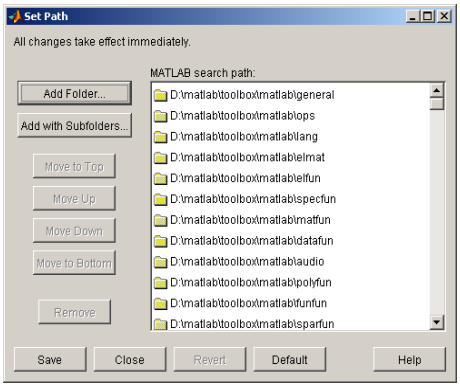
Polecenie help jest w rzeczywistości funkcją którą należy utożsamiać np.: z pojęciem funkcji w języku Pascal. Przyjmuje ona argumenty i zawraca wartość np.: **msg=help('sin');** oznacza, że funkcja help znajdzie informacje na temat **sin** a tekst pomocy zapisze do zmiennej **msg**.

Wersja 7.0 posiada dodatkowe polecenie o nazwie **doc** związane z systemem pomocy. Sposób użycia tego polecenia jest podobny do help np.: `doc vpa` spowoduje wyświetlenie informacji za pośrednictwem okna *Help*, czyli z głównego systemu pomocy.

Wiele cennych informacji znajduje się także na stronie producenta programu Matlab[4]. Natomiast podstawowym forum wymiany informacji o tym programie jest strona MatLabCentral[5]. Znajduje się tam wiele odsyłaczy do innych miejsc w sieci oraz wiele przykładów stosowania różnych toolbox'ów.

1.5 Ścieżki przeszukiwań

Istotną czynnością jaką warto wykonać przed rozpoczęciem pracy jest utworzenie nowego katalogu. W katalogu tym powinny znajdować się pliki z jakimi będziemy pracować. Aby zwiększyć wygodę pracy warto dołączyć wspomniany katalog do ścieżki przeszukiwań. Tą czynność wykonujemy w oknie *Set Path* (Rysunek 2). Aby to zrobić należy wybrać opcję *Set Path* z menu *File* a następnie dodać potrzebną ścieżkę za pomocą przycisku *Add Folder...* poczym zapisać zmiany przyciskiem *Save*. Pozwoli to korzystać z utworzonych przez nas m-plików identycznie jak z pozostałych funkcji programu Matlab. Ścieżkę można także dodać poprzez polecenie



Rysunek 2: Ścieżki przeszukiwań, czyli okno *Set Path*

addpath. Wystarczy np.: wydać następujące polecenie: `addpath c:\mojepliki` aby uzupełnić ścieżkę przeszukiwań o nowe katalog. W Matlabie wiele rzeczy można na różne sposoby. Nową ścieżkę można dopisać także w

[4]http://www.mathworks.com/
[5]http://www.mathworks.com/matlabcentral/

następujący sposób: `path(path, 'c:\\mojepliki')`. Czasami przydaje się informacja o tym w jakim katalogu został zainstalowany Matlab. Tą informację uzyskamy za pomocą polecenia: **matlabroot**.

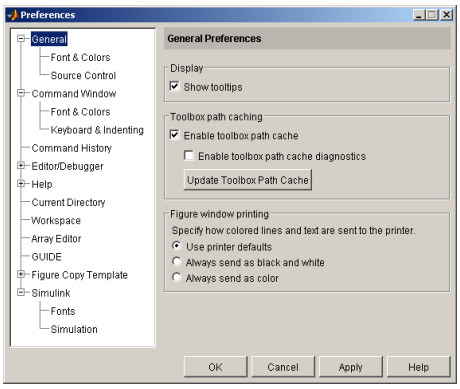
1.6 Uruchamianie i wyłączanie Matlab'a

Proces uruchamiania Matlab'a sprowadza się do kliknięcie na odpowiednią ikonę na pulpicie bądź wybrania z menu Start jeśli Matlab został zainstalowany w środowisku Windows stosownej opcji. Matlab można także uruchomić za pomocą pliku bat. Znajduje się on w katalogu `c:\matlab\bin`, jeśli naturalnie program został zainstalowany w katalogu `c:\matlab`.

Podczas pierwszego uruchomienia Matlaba po instalacji dokonywana jest indeksacja toolbox'ów. Operację tę warto powtórzyć jeśli instalowany były nowe toolbox'y. Wykonujemy ją np.: wydając polecenie **rehash** z odpowiednim parametrem:

```
>> rehash toolboxcache
```

Oczywiście operację uaktualnienia wykonać z poziomu menu. Należy z menu *File* wybrać opcję *Preferences...*. Ukaże się okno podobne do okna z rysunku 3 a następnie w sekcji *General* przyciskiem *Update Toolbox Path Cache* rozpoczyna operację uaktualnienia.



Rysunek 3: Okno preferencji

Zakończenie pracy z Matlabem można wykonać na dwa sposoby. Wybierając z menu *File* opcję *Exit*, bądź klikając na oknie przycisk zamknięcia okna lub kombinacją klawiszy *ALT-F4*. Oprócz tych podstawowych możliwości zakończenie pracy można wymusić poleceniem **quit**. Przyjmuje ono dwa argumenty: **quit cancel** odwołuje proces zamknięcia natomiast **quit force** definitywnie zamknie program. Otóż polecenie quit domyślnie szuka m-pliku o nazwie *finish.m*. Jeśli ten plik zostanie odszukany to wykonywana jest jego treść. W owym pliku można nakazać odwołanie operacji zamknięcia podaną opcją **cancel**. Doskonałym przykładem może być poniższy kod który umieszczony w pliku *finish.m* w momencie zamykania programu zada pytanie czy z istotnie chce zamknąć program Matlab:

```
button = questdlg('Zakończyć działanie programu?', 'Pytanie','Tak','Nie','Nie');
switch button
  case 'Tak',
    disp('Zamykanie programu');
    save
  case 'Nie',
    quit cancel;
end
```

W powyższym skrypcie występuje jedno bardzo istotne polecenie: **save**. Za jego pomocą można zapisać całą przestrzeń roboczą do pliku. Polecenie to przyjmuje cały szereg parametrów np.: **save roboczy.mat** zapisze workspace do pliku *roboczy* o rozszerzeniu *mat*. Za wczytanie zapisanych danych jest odpowiedzialne polecenie **load**.

Podczas pracy z Matlabem nader często przydaje się historia wszystkich poleceń jakie są wydawane z poziomu konsoli. Sam Matlab w oknie *History* przechowuje historie wydanych poleceń. Można też nakazać programowy umieszczanie wszystkich danych wyświetlanych na konsoli (nie tylko historię wydanych poleceń ale także ich wyników) poleceniem **diary**. Polecenie to przyjmuje za argument nazwę pliku. Wydanie polecenie `diary moja_sesja.txt` z powoduje, że w pliku *moja_sesja.txt* będa zapisywane wszystkie informacje jakie ukazują na ekranie konsoli. Wyłączenie zapisu następuje po wydaniu polecenia: `diary off`.

1.7 Zmienne oraz funkcje specjalne

Jeśli wynik działania jakiejkolwiek operacji nie zostanie umieszczony w zmiennej to Matlab samodzielnie tworzy zmienną o nazwie **ans**. W tej zmiennej umieszczany jest wynik działania.

Matlab posiada też predefiniowane wartości do oznaczania wartości specjalnych np.: nieskończoności. Za pomocą litery **j** oznaczany jest czynnik liczby zespolony. Za pomocą skrótu **NaN** (ang. *not a number*) oznaczono sytuację, że dana wartość nie jest wartością w sensie liczbowym. Podobnie przez **Inf** oznaczono nieskończoność.

Wartości NaN, Inf może stosować tak jak inne liczby czy ciągi znaków i porównywać choćby przy pomocy instrukcji **if**. Matlab oferuje także specjalne funkcje sprawdzające czy podana zmienna czy wyrażenie jest określonego typu. Oto kilka przykładów:

- isinf – sprawdza czy wartość jest nieskończona
- isfinite – przeciwieństwo poprzedniej funkcji bo sprawdzamy czy wartość jest skończona
- isnan – czy wartość jest typu NaN
- isreal – czy wartość jest typu rzeczywistego
- isfloat – czy wartość jest typu zmiennoprzecinkowego
- isinteger – czy wartość jest całkowita

Matlab jako pakiet numeryczny definiuje kilkanaście typów. Oprócz podziału na macierze, wektory, skalary. W ramach tych ostatnich wyróżnia się kilkanaście różnych typów. Tabela 1 prezentuje podstawowe typu dostępne w Matlabie. Dokładniejsze informacje o typach można przeczytać po wydaniu następującego polecenia: **help datatypes**

W przypadku różnych typów całkowitych czy zmiennoprzecinkowych ważne są wartości minimalne i maksymalne. W kontekście metod numerycznych do bardzo istotna informacja. Poznajemy je za pomocą następujących funkcji:

- intmin
- realmin
- intmax
- realmax

Uzupełniającą te informacje jest funkcja **eps**. Jej wartością jest dokładność z jaką przeprowadzane są obliczenia a dokładniej różnica pomiędzy podaną liczbą na następną większą od niej liczbą zmiennoprzecinkową. Poniżej kilka przykładów zastosowania funkcji **eps**:

```
double precision
   eps(1/2) = 2^(-53)
   eps(1) = 2^(-52)
   eps(2) = 2^(-51)
single precision
   eps(single(1/2)) = 2^(-24)
   eps(single(1)) = 2^(-23)
   eps(single(2)) = 2^(-22)
```

Typ	Krótki opis
logical	Logical array of true and false values
char	Characters array
numeric	Integer or floating-point array
integer	Signed or unsigned integer array
int8	8-bit signed integer array
uint8	8-bit unsigned integer array
int16	16-bit signed integer array
uint16	16-bit unsigned integer array
int32	32-bit signed integer array
uint32	32-bit unsigned integer array
int64	64-bit signed integer array
uint64	64-bit unsigned integer array
float	Single- or double-precision floating-point array
single	Single-precision floating-point array
double	Double-precision floating-point array
cell	Cell array
struct	Structure array
function_handle	Function handle
'class_name'	Custom MATLAB object class or Java class

Tabela 1: Podstawowe typy danych w Matlab'ie

1.8 Format wyjścia oraz funkcje wyjścia

Standardowo wyniki obliczenia w Matlabie są wyświetlane do pięciu miejsc po przecinku. Odpowiada to poleceniu `format short`. Jest to tzw. format stałoprzecinkowy. Zwiększenie dokładności do piętnastu miejsc po przecinku wymaga podania opcji **long** dla polecenia **format**. Polecenie to pozwala na wyświetlanie liczba w postaci szesnastkowej (`format hex`) oraz w postaci ułamków `format rat` jednak tylko dla małych liczb.

Większa dokładność obliczeń można wymusić poprzez polecenie **vpa** np.: `vpa(pi, 1000)` oblicza wartość liczby π do tysięcznego miejsca po przecinku.

Choć po wpisaniu samej nazwy zmiennej Matlab wyświetli wartość jaką jest w niej przechowywana, w skryptach taki sposób wyświetlania wartości nie jest poprawny. Do wyświetlania własnych komunikatów można zastosować dwa polecenia **disp** oraz **sprintf**.

Polecenie **disp** przyjmuje za argument wyrażenie lub zmienną przeznaczoną do wyświetlenia np.: `disp(a)`. Dopuszczalnym argumentem jest ciąg znaków objęty apostrofami: np.: `disp(' jakis tekst ')`. Gdy istnieje potrzeba wyświetlenia kilku zmiennych to należy stosować operator konkatencji ciągów znaków reprezentowany przez dwa nawiasy kwadratowe. Jednak należy w przypadku każdej wykonać konwersję z wartości liczbowej na tekst za pomocą funkcji **num2str** (bądź **int2str**):

```
x=1; y=2; z=3;
disp([num2str(x) ',' num2str(y) ',' num2str(z)])
```

Więcej możliwości formatowania reprezentacji liczb daje funkcja przeniesiona z języka C o nazwie **sprintf**. Jednak przygotowuje ona tylko i wyłącznie ciąg znaków zgodnie z parametrami. Do ostatecznego wyświetlenia trzeba stosować polecenie **disp**. Przykład formatowania z wykorzystaniem polecenia **sprintf** jest następujący:

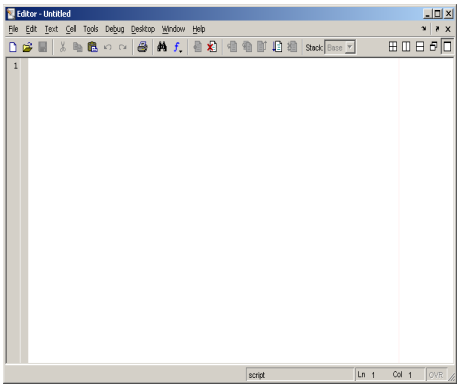
```
x=1; y=2; z=3;
disp(sprintf('%d %d %d',x,y,z))
```

1.9 Tworzenie m-plików

Matlab jest środowiskiem interaktywnym tzn. z poziomu konsoli wydajemy polecenia. Jednak wiele z tych poleceń to nazwy skryptów, czyli plików zawierających polecenia do wykonania przez Matlaba. Takie pliki nazywa się m-plikami. Istnieją dwa rodzaje tych plików tzw. skrypty gdzie instrukcje wykonywane są kolejno

linia po linii począwszy od początku pliku. Drugi rodzaj to m-funkcje. W pliku tworzona jest specjalna funkcja która przyjmuje argumenty. W ten sposób zostały opracowane niemal wszystkie dodatkowe pakiety dostępne w systemie.

Tworzenie m-funkcji jest tematem jednego z następnych laboratoriów i zostanie tam dokładniej omówione. Jednakże tworzenie skryptów jest bardzo łatwe. Z menu *File* należy wybrać opcję *New* a z menu jakie się ukaże opcję *M-File*. Ukaże się okno podobne do okna z rysunku 4. Po wpisaniu treści m-pliki można go natychmiast



Rysunek 4: Okno edytora m-plików

wykonać naciskając klawisz F5. Plik taki po zapisaniu na dysku pod określoną nazwą można wykonać wpisując w oknie konsoli nazwę pliku (katalog zawierający ten plik musi naturalnie być wpisany do listę ścieżek przeszukiwań).

1.10 Octave – darmowy odpowiednik

Program Octave[6] to darmowy odpowiednik programu Matlab. Choć nie jest on tak bardzo rozbudowany jak Matlab to ma pewne zalety. Po pierwsze jest programem znacznie mniejszym przez co nie wymaga dużej ilości pamięci RAM czy szybkiego procesora. Pracuje bardzo sprawnie na starszych komputerach. Posiada pełny kod źródłowy co pozwala na pełna modyfikację tego pakietu i dopasowanie jego do własnych potrzeb.

Octave nie oferuje po instalacji tak dużego zbioru funkcji jak Matlab ale z punktu widzenia metod numerycznych obydwa pakiety oferują podobne możliwości. Przy czym w rzeczywistości społeczność użytkowników tego programu opracowała wiele dodatkowych toolbox'ów. Niestety dodatkowe funkcje wymagają więcej wysiłku w ich instalacje.

Pakiet Octave najlepiej funkcjonuje w systemie Unix w tym oczywiście także pod Linux'em. W nim też najłatwiej instalować dodatkowe pakiety. Wydajność pakietu wbrew pozorom nie jest niższa niż Matlab a to z tego względu, że wykorzystywane są te same zbiory metod numerycznych zgromadzone w pakietach LAPACK, BLAS oraz Atlas. Co więcej mogą one działać nawet sprawniej ponieważ Octave jest pozbawiony wielu dodatków co niewątpliwie wpływa bardzo dobrze na jego szeroko pojętą wydajność.

Sposób pracy z Octave jest identyczny jak z Matlab'em. Dostępny jest tryb interaktywny, można też tworzyć m-pliki. Na stronie domowej tego projektu dostępna jest obszerna dokumentacja.

1.11 Zadania

1. Odszukać informacje o funkcji **det** za pomocą graficznego interfejsu (poszukać za pomocą zakładek Index oraz Search). Co wyznacza funkcja **det**?

2. Poszukać informacji o funkcji **fsolve** za pomocą okna *Help*.

3. Spróbować wykonać przykłady związane z funkcją **fsolve**.

4. Wyszukać informacje o funkcji **plot** za pomocą *Search* we wszystkich czterech kategoriach.

[6]Program można odszukać pod adresem http://www.octave.org. Jest on też dostępny w wielu dystrybucjach Linux'a np.: RedHat, Debian.

5. Jakie informacje otrzymujemy po wydaniu polecenia **help**.
6. Przeczytać informacje uzyskane z polecenia: **help help**.
7. Uzyskać informacje o poleceniach: **lookfor**, **what**, **which**, **more**, **who**. Podać przykłady ich zastosowań.
8. Polecenie help (jak i kilka innych) występuje w dwóch formach. Jakie są to formy i jakie mają przeznaczenie.
9. Sprawdzić jak działają klawisze kursora w konsoli programu Matlab.
10. Odszukać informacje na temat poleceń systemu plików: **dir**, **cd**, **ls**, **pwd**, **copyfile**, **delete**, **fileattrib**, **movefile**, **mkdir**, **rmdir**, **exist**.
11. Jakie jest przeznaczenie poleceń: **clear**, **clc**, **home**
12. Sprawdzić działanie funkcji **demo**.
13. Wykonać następujące polecenia w Matlabie:

```
[x,y] = meshgrid(-3:1:3);
z = peaks(x,y);
surf(x,y,z)
```

    Zapoznać się z funkcjami okna wykresów jakie się pojawiło.
14. Utworzyć plik *finish.m* z treścią podaną w poprzednich punktach i sprawdzić czy istotnie blokuje on proces wyłączanie programu?
15. Sprawdzić jak działają polecenia **save**, **load** oraz **diary**.?
16. Polecenie **save** posiada możliwość zapisu wybranych zmiennych podać przykład jego zastosowania.
17. Sprawdzić na czym polegają różnice pomiędzy poleceniami **path** oraz **addpath**.
18. Po co stosujemy polecenie **rehash**? Jakie inne argumenty można je zastosować?
19. Napisać krótki skrypt (m-plik) który pokaże wartości minimalne i maksymalne dla wszystkich podstawowych typów liczbowych.
20. Funkcje num2str oraz int2str wykonują tą samą operację dokonują konwersji liczby na ciąg znaków. Różnią się jednak istotnym detalem. Jakim?

2 Operacje na macierzach oraz obliczenia symboliczne

2.1 Zadania

1. Obliczyć numerycznie następujące wyrażenia:
   (a) $4 \operatorname{arc\,tg} 1$, π^2, $\sqrt{10}$
   (b) rozwiązać równania: $x^2 + 1 = 0$, $x^2 - x + 3 = 0$, $x^2 - 6x + 13 = 0$
   (c) wykonać działania: $(3 + 7i)(-2 + i) + (-5 - 2i)(-1 - 7i)$, $\frac{43+7i}{2-3i}$, $\left(\frac{1}{2} - i\frac{\sqrt{3}}{2}\right)^3$
2. Dla następującej macierzy A:
$$\begin{pmatrix} 11 & 12 & 13 & 14 \\ 21 & 22 & 23 & 24 \\ 31 & 32 & 33 & 34 \\ 41 & 42 & 43 & 44 \end{pmatrix}$$
   wykonać z poziomu konsoli następujące polecenia:

(a) zdefiniować zmienną A (np.: `A=[11 12 13 14 ; 21 i etc. ]`)

(b) wykonać operacje: A, A(:,3), A(2,:), A(1,2), A(1:3,3:4), A(:), A(:,:)

(c) wykonać operacje: [A A], [A ; A], [A(:, 1) A(:,2)], [A(:, 1) ; A(1, :)]

(d) wykonać operacje: A(:,1)=[ ], A(2, :)=[ ], A(1,1)=[], x(1,:)=[1 4 9]

(e) przeczytać do czego wykorzystuje się polecenia: **zeros**, **ones**, **eye**, **diag**, **rand**

(f) wykonać operacje: ones(3,3), eye(3), diag(A,-1), diag(A), diag(A,1), [eye(2) ; rand(2,2)]

3. Narysować trójkąt Sierpińskiego wykorzystując fakt, że reszta z dzielenia modulo dwa poszczególnych elementów trójkąta Pascala daje przybliżony obraz trójkąta Sierpińskiego (wykorzystać polecenie `format +` aby poprawić czytelność).

4. Za pomocą funkcji diag, ones oraz operatora dodawania i zakresu podać polecenia potrzebne do zbudowania następującej macierzy:

$$\begin{pmatrix} -3 & 3 & 0 & 0 & 0 & 0 & 0 \\ 2 & -2 & 3 & 0 & 0 & 0 & 0 \\ 0 & 2 & -1 & 3 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 & 3 & 0 \\ 0 & 0 & 0 & 0 & 2 & 2 & 3 \\ 0 & 0 & 0 & 0 & 0 & 2 & 3 \end{pmatrix}$$

.

5. Wykonać następujące operacje na macierzach A i B:

(a) $|A|$, $|A||B|$, $|AB|$, A^T, AB, $A+B$, $2A+4B^T$, AA, $\det A$, $\det AB$, $\det A \det B$, $2+B$

6. Rozwiązać następujące układy równań (korzystamy tylko z funkcji **det**):

(a) $\begin{cases} 3x+4y-z=6 \\ 6x-5y+2z=8 \\ 9x-4y+z=10 \end{cases}$

(b) $\begin{cases} 5x+2y=-1 \\ 3x+3z=9 \\ 2y-2z=-4 \end{cases}$

(c) $\begin{cases} 2x+3y=6 \\ 3x+2y=9 \end{cases}$

2.2 Obliczenia symboliczne w Matlabie

Choć Matlab jest zorientowany przede wszystkim na obliczenia numeryczne to został wyposażony w odpowiedni toolbox do przeprowadzania obliczeń symbolicznych. Toolbox ten nazywa się *Symbolic Math*, co warto podkreślić pakiet ten umożliwia korzystanie z Maple zwiększając radykalnie możliwości Matlab'a w dziedzinie przetwarzania symbolicznego.

Korzystanie z tego pakietu wymaga stosowania specjalnego słowa kluczowego o nazwie **sym**. Komplikuje to nieco zapis jednak w praktyce okazuje się łatwe do opanowania np.: gdy podamy wyrażenie `1/2` to naturalnie Matlab nam odpowie, że jest to wartość: `0.5000`. Gdy jednak podamy polecenie sym(1/2) to otrzymamy zapis tego ułamka. Pozwala to nam łatwe prowadzenie operacji na ułamkach zwykłych np.: `sym(1/2) + sym(1/4)` otrzymamy wartość `3/4`.

Następny przykład jest nieco bardziej skomplikowany. Dysponujemy następującym wyrażeniem ρ opisujący złoty podział:

$$\rho = \frac{1+\sqrt{5}}{2}$$

Za pomocą polecenia **sym** definiujemy powyższe wyrażenie:

```
rho = sym('(1 + sqrt(5))/2')
```

Na obiektach symbolicznych możemy wykonywać operacje tak jak na innych obiektach w Matlabie:

```
f = rho^2 - rho - 1 ;
```

Jeśli po wydaniu powyższego polecenia wpiszemy „f" i naciśniemy enter to zobaczymy wynik operacji jaką nakazaliśmy wykonać (pewną wadą jest niestety brak matematycznej notacji, do poprawienia czytelności pomocne jest polecenie **pretty**):

```
(1/2+1/2*5^(1/2))^2-3/2-1/2*5^(1/2)
```

Pakiet *Symbolic Math* pozwoli nam na uproszczenie tego wyrażenia. Stanie się to wydaniu polecenia: `simplify(f)`.

Podczas stosowania wyrażeń należy też zadbać o odpowiednie zdefiniowanie zmiennych. Jest to szczególnie istotne w momencie liczenia pochodnych czy całkowania. Wszystkie zamienne w danych wyrażeniu muszą być zmiennymi symbolicznymi inaczej wyrażenie nie będzie możliwe do zdefiniowania. W poniższym przykładzie definiujemy ogólną postać równania kwadratowego. Jak widać poszczególnie zmienne a,b,c,x tego wyrażenia zostały zdefiniowana z użyciem polecenia **sym**:

```
a = sym('a')
b = sym('b')
c = sym('c')
x = sym('x')

f = sym('a*x^2 + b*x + c')
```

Taka definicja obliczyć pochodną (funkcją **diff**) i uzyskać wynik zgodny z oczekiwaniami:

```
>> diff(f)

ans =

2*a*x+b
```

Dość często w przypadku dużej liczby zmiennych stosuje się skrót w postaci polecenia **syms** np.: `syms a b c d`.

2.3 Polecenie subs

Zadaniem tego polecenie są podstawienia danych dla zmiennych symbolicznych. Dysponując wyrażeniem $cos(a)+sin(b)$ chcemy podstawić w miejsce zmiennych a i b wartości odpowiednio: α oraz 2. Polecenia są następujące:

```
subs(cos(a)+sin(b),{a,b},{sym('alpha'),2})
```

otrzymamy

```
cos(alpha)+sin(2)
```

Polecenie to warto stosować podczas tworzenia wykresów gdzie funkcja znajduje się w postaci symbolicznej. Załóżmy że g reprezentuje dowolną funkcję (ze zmienna symboliczną x) natomiast xd zawiera elementy dziedziny. Uzyskanie wartości funkcji sprowadza się do następującego polecenia: `yd = subs(g,x,xd);`

2.4 Szereg Taylora

Prócz całkowania symbolicznego czy liczenia pochodnych jedną z ciekawszych funkcji w kontekście metod numerycznych jest generowania szeregu Taylora dla zadanych funkcji. Matlab 7.0 posiada specjalne narzędzie **taylortool**. W oknie jakie się nam ukaże można w wizualny sposób badać rozwinięcia różnych funkcji i porównywać wykresy funkcji rzeczywistej oraz szeregu.

Na poziomie konsoli dostępne jest polecenie **taylor** którym można wygenerować potrzebny szereg. Korzystamy naturalnie z możliwości pakietu *Symbolic Math*.

Ogólna postać polecenia do generowania szeregu jest następująca:

```
r = taylor(f,n,v,a)
```

Wynik czyli szereg w postaci obiektu symbolicznie jest zapisywany do zmiennej r. W argumentach funkcja ta przyjmuje następujące wyrażenie:

- f – oznacza funkcję jaką chcemy przybliżać
- n – ilość elementów a dokładnie najwyższą potęga jak zostanie zastosowana w rozwinięciu
- v – oznacza niezależną zmienną
- a – wartość wokół której wyliczany będzie szereg[7]

Nie trzeba podawać wszystkich argumentów najważniejszym jest naturalnie funkcja. Poniższy przykład pokazuje jak wyznaczyć szereg dla funkcji sin:

```
syms x;
f=sym('sin(x)');
t=taylor(f);
pretty(t)
```

Aby uzyskać dłuższe rozwinięcie np.: do dziesiątej potęgi podajemy następujące polecenie:

```
t=taylor(f, 10)
```

Ponieważ polecenie **taylor** zwraca funkcją to po przypisaniu wyniku do zmiennej symbolicznej można narysować wykres szeregu. W tym celu najlepiej skorzystać z polecenia **ezplot** np.: wydając polecenia: `ezplot(f) ; hold on ; ezpl` zobaczymy na wykresie jakość przybliżenia szeregu Taylora odpowiada rzeczywistej funkcji zdefiniowanej w zmiennej f.

2.5 Liczenie pochodnych i granic

Do wyznaczania pochodnych służy polecenie **diff**. Podobnie jak inne funkcje pakietu **Symbolic Math** wymaga ono zdefiniowania obiektów symbolicznych. Wyznaczenie pochodnej funkcji sin można zrealizować w następujący sposób:

```
>> syms x;
>> f=sym('sin(x)');
>> diff(f)
ans =

cos(x)
```

Wyznaczenie n-tej pochodnej jest równie trywialne wystarczy podać numer pochodnej jaka jest nam potrzebna np.: `diff(f, 3)` i otrzymamy trzecią pochodną funkcji sin.

Jeśli mamy zdefiniowaną zmienną symboliczną to nie istnieje potrzeba definiowania dodatkowego obiektu dla funkcji. Jej treść możemy podawać bezpośrednio do polecenia **diff** np.:

```
>> diff(x^3)

ans =

3*x^2
```

Podobnie jest zrealizowane liczenie granic przy pomocy polecenie **limit**. Ogólna postać wywołania przedstawia się następująco:

```
limit(F,x,a,'right')
```

Argumenty są następujące:

[7] W dalszej części tego dokumentu punkt ten będzie oznaczany jako x_0

- F – postać funkcji
- x – zmienna
- a – punkt graniczny (za pomocą **inf** oznaczamy nieskończoność)
- 'right' bądź 'left' – kierunek z którego zbliżamy się do punktu granicznego

Jeśli chcemy policzyć następującą granicę: $lim_{x\to\infty}x^3$ to polecenie limit przyjmuje następującą postać:

```
>> limit(x^3, x, inf)

ans =

Inf
```

W przypadku granic lewo bądź prawo stronnych należy podać w ostatnim argumencie wartość 'left' bądź 'right'. Dla przykładu granicę $\lim_{x\to 0^-}\frac{1}{x}$ obliczymy po wydaniu następującego polecenia:

```
limit(1/x, x, 0, 'left')
```

Istotna uwaga gdy podamy tylko jeden argument czyli funkcję dla której liczmy granicę wtedy domyślnym punktem granicznym jest zero.

2.6 Symbolicznie rozwiązywanie równań

Pakiet do obliczeń symbolicznych oferuje rozwiązywanie równań algebraicznych oraz ich układów oraz zwyczajnych równań różniczkowych oraz ich układów. Pierwszy typ równań rozwiązujemy funkcją **solve** natomiast drugi **dsolve**.

Rozwiązanie równania wymaga zdefiniowana za pomocą **sym** lub **syms** wszystkich zmiennych oraz parametrów jakie wchodzą w skład równania. Załóżmy, że chcemy rozwiązać symbolicznie równanie kwadratowe: $ax^2 + bx + c = 0$. W pierwszej kolejności definiujemy zmienne i parametry następnie tworzymy zmienną zawierającą postać naszego równania i za pomocą solve uzyskujemy pierwiastki.

```
>> syms a b c x ;
>> S = a*x^2 + b*x + c;
>> solve(S)

ans =

 1/2/a*(-b+(b^2-4*a*c)^(1/2))
 1/2/a*(-b-(b^2-4*a*c)^(1/2))
```

Polecenie solve umożliwia także rozwiązywanie według dowolnej zmiennej np.: rozwiązanie równania kwadratowego względem **b** to polecenie: **solve(S,b)**. Domyślnie polecenie solve rozwiązuje równania w postaci: $f(x) = 0$, jeśli chcemy rozwiązywać równania w postaci $f(x) = q(x)$ to należy postać równania ująć w apostrofy np.: `solve('cos(x)=1')`.

Polecenie **solve** pozwala także na rozwiązywanie układów równań. Jeśli poszczególne równania są w postaci:

$$\begin{aligned} f_1(x) &= 0 \\ f_2(x) &= 0 \\ &... \\ f_n(x) &= 0 \end{aligned}$$

To polecenie solve przyjmuje następującą postać: `solve(row1 , row2, row3)`, inaczej mówiąc poszczególne równania rozdzielamy przecinkiem. Podobnie jak pojedynczym równaniu jeśli postać układu jest następująca:

$$\begin{aligned} f_1(x) &= q_1(x) \\ f_2(x) &= q_2(x) \\ &\dots \\ f_n(x) &= q_3(x) \end{aligned}$$

To poszczególne równania rozdzielone przecinkami dodatkowo obejmujemy cudzysłowami.

2.7 Zadania

1. Wyznaczyć za pomocą Matlaba (nie liczyć na kartce!) wartości następujących granic:
   (a) $\lim_{x\to 0} \frac{\sin(x)}{x}$, $\lim_{x\to 0^-} \frac{1}{x}$, $\lim_{x\to 0^+} \frac{1}{x}$, $\lim_{h\to 0} \frac{sin(x+h)-sin(x)}{h}$
   (b) $\lim_{x\to 1} \frac{x^2+2x+1}{x^3+1}$, $\lim_{x\to\infty} \sqrt{x^2+5x+5}+x$
   (c) $\lim_{x\to\infty} \frac{x^2+3x+2}{x^3+3x+1}$, $\lim_{x\to\infty} \frac{\log(2x+7)}{x^2+3x+1}$
   (d) $\lim_{x\to 0} \operatorname{arc}\cos(\frac{e^x-1}{2x})$, $\lim_{x\to 0} \sqrt[3]{\frac{\sin(2x)}{x}}$
2. Obliczyć pochodne poniższych funkcji:
   (a) $f(x) = 2x^4 + x^2 - 5$, $f(x) = ax^6 + 3x^3 - c$
   (b) $f(x) = \sin(x)$, $f(x) = \log_a x$, $f(x) = e^x$
   (c) $f(x) = \frac{1}{x^2}$, $f(x) = \frac{x}{1+x}$, $f(x) = \frac{1}{\sqrt{x}}$
3. Rozwiązać następujące równania bądź układy:
   (a) $ax - b = 0$, $x^2 - 2x + 5 = 0$, $x^2 - x + 1 = 0$, $ax^3 + bx^2 + cx + d$
   (b) $\cos(2x) + sin(x) = 1$, $p\cos(x) = r$, $\sin(2x) = r$, $\sin(2x) = 1$
   (c) $\begin{cases} ax + by = c \\ dx + ey = f \end{cases}$, $\begin{cases} x^2y^2 = 0 \\ x - \frac{y}{2} = \alpha \end{cases}$
4. W jakim przypadku funkcja Matlab'a **taylor** istotnie generuje szereg Taylor'a, a w jakim MacLaurina?
5. Wyznaczyć szeregi Taylora (5 i 10 elementów szeregu) następujących funkcji:
   (a) $f(x) = \sin(x)$, $f(x) = \cos(x)$, $f(x) = \ln x$, $f(x) = e^x$, $f(x) = \frac{x}{\sqrt{1+x}}$
6. Wyznaczyć szeregi Taylora, przyjmując że $x_0 = 1$ i $n = 3$, dla następujących funkcji:
   (a) $f(x) = e^x$, $f(x) = e^{-x}$, $f(x) = \ln x$
7. Za pomocą polecenia **taylortool** zbadać jak rząd rozwinięcia szeregu wpływa na jakość odwzorowania następujących funkcji:
   (a) $f(x) = x\cos(x)$
   (b) $f(x) = e^x$, dla jakiej wartości n otrzymujemy na przedziale $-2\pi \leqslant x \leqslant 2\pi$, „rozsądne" przybliżenie
8. Przygotować wykres bez użycia narzędzia **taylortool** funkcję $f(x) = e^{x\sin(x)}$ oraz odpowiadający jej szereg Taylora dla parametrów:
   (a) $n = 5, x_0 = 2$
   (b) $n = 10, x_0 = 2$
   (c) $n = 15, x_0 = 2$

3 Wykresy 2D oraz 3D

3.1 Wstęp

Matlab oferuje bardzo szerokie możliwości wizualizacji danych. Oprócz tworzenie klasycznych wykresów w układach kartezjańskich, biegunowym dostępne także są wykresy słupkowe, konturowe. Dostępne są również wykresy trójwymiarowe. W dość prosty sposób można tworzyć animację wykresów.

3.2 Wykresy 2D

Za tworzenie wykresów w dwóch wymiarach odpowiedzialne jest polecenie **plot**. Polecenie to przyjmuje szereg parametrów. Opis tych parametrów uzyskamy po wykonaniu skorzystaniu z pomocy: **help plot**.

Najmniej skomplikowana droga do uzyskania przebiegu dowolnej funkcji jest następująca. Przygotowujemy wartość dziedziny funkcji np.: `x=-2:0.01:2;`. Generujemy wartości funkcji np.: `y=x.*x.*x;`. Ostatnim elementem jest uzyskanie wykresu: `plot(x,y)`.

Do tworzenia wykresów w układzie biegunowym wykorzystuje się polecenie **polar**. Sposób korzystania z tego polecenia jest taki sam jak dla polecenie **plot**. Nader często korzystać trzeba wykresów o logarytmicznej skali. Wykorzystuje się polecenie **loglog** np.:

```
x = logspace(-1,2);
loglog(x,exp(x),'-s')
grid on
```

3.3 Wykresy 3D

Otrzymywanie wykresów trójwymiarowych jest dość podobne jak w przypadku poleceń dla grafiki dwywymiarowej. Przygotowujemy wartości x oraz y jednak wykorzystujemy funkcje **meshgrid**:

```
[x,y] = meshgrid(-2:.2:2, -2:.2:2);
```

Następnie wyznaczamy wartość dla osi z: `z = x .* exp(-x.^2 - y.^2);`. Ostatnim elementem jest narysowanie wykresu: `mesh(z)`. Użyte polecenie **mesh** tworzy wykres siatkowy jeśli chcemy uzyskać wykres bardziej kolory należy zastosować polecenie **surf**. Standardowym poleceniem do rysowania w przestrzeni trójwymiarowej jest polecenie **plot3**.

3.4 Zarządzanie wykresami

Choć funkcja **plot** samoczynnie tworzy okno z wykresem nader często istnieje potrzeba utworzenia pustego okna. Do tego celu przeznaczona jest funkcja **figure**. Aby narysować elementarne wykresy funkcji x^2 oraz $-x^2$ wystarczy przygotować dane: `x=-3:0.1:3; y=x.*x;`. A następnie za pomocą **figure** utworzyć nowe okno i wydać polecenie **plot**. Dla następnego wykresu ponownie wydajemy polecenie **figure** tworząc tym samym następne okno. Poszczególne polecenia prezentują się następująco:

```
plot(x,y)
figure
plot(x,-y)
```

Rysowanie nowego wykresu na istniejącym jest możliwe po wydaniu polecenia **hold**. Następnie polecenie **plot** nie powoduje tworzenia nowego okna. Narysowanie obydwu parabol z poprzedniego przykładu na jednym wykresie jest następujące:

```
x=-3:0.1:3;
y=x.*x;
figure
plot(x,y)
hold
plot(x,-y)
```

Polecenie **plot** oprócz podania danych, pozwala opisać w jaki sposób ma być rysowania linia samego wykresu. Opis wszystkich oznaczeń uzyskamy po wydaniu polecenie **help plot**. Do narysowania wykresu np.: za pomocą czerwonych kropek polecenie jest następujące: `plot(x,y,'.r')`. Znak „.” oznacza naturalnie kropkę, natomiast litera „r” to nazwa koloru czerwonego. Zastosowane w jednym z poprzednich przykładów opis „-s” oznacza wykres narysowany ciągłą linią ale w punktach węzłowych zostaną narysowane niewielkie kwadraty.

Po narysowaniu samego wykresu wykres można dalej przetwarzać np.: polecenie **grid on** spowoduje włączenie siatki. Poleceniem **title** określa się tytuł wykresu. Co ważne można stosować notację z Tex'a np.: `title('f(x)=x^2');`. Inny przykład z greckimi literami:

```
title('\ite^{\omega\tau} = cos(\omega\tau) + isin(\omega\tau)')
```

Sam wykres także może być opatrywany komentarzami za pomocą polecenia **text**. Dwa pierwsze argumenty to współrzędne a następnie podawany jest tekst komunikatu. Po nim może wystąpić dalszy opis np.: wielkości fontu.

```
plot(0:pi/20:2*pi,sin(0:pi/20:2*pi))
text(pi,0,' \leftarrow sin(\pi)','FontSize',18)
```

Poszczególne osie także można opisać za pomocą poleceń **xlabel**, **ylabel** oraz dla wykresów trójwymiarowych **zlabel**. Krótki przykład:

```
xlabel('t = 0 to 2\pi','FontSize',16)
ylabel('sin(t)','FontSize',16)
title('\it{Wartości funkcji sin od zera do Pi}','FontSize',16)
```

Wydruk wykresów czy też kopiowanie poprzez schowek można wykonać z poziomu menu. Matlab oferuje także polecenie **print** z przeznaczeniem do wydruku bądź zapisu postaci wykresu do pliku. Zapis do pliku w formacie *PostScipt 2* przy zachowaniu kolorów jest następujący:

```
figure
% tworzenie wykresu
print -dpsc2 sin.ps
```

Wśród wielu dostępnych funkcji przydatne jest polecenie **subplot**. Polecenie dzieli okno wykresu ma macierz o podanych wymiarach. Poszczególne pod-wykresy są numerowane począwszy od jedności. Pierwszy wykres znajduje się w lewym górnym rogu. Poniżej znajduje się przykład z poleceniem **subplot**:

```
subplot(2,2,1)
plot(x,y1)
subplot(2,2,2)
plot(x,y2)
subplot(2,2,3)
plot(x,y3)
subplot(2,2,4)
plot(x,y4)
```

3.5 Animacja

Matlab umożliwia bardzo łatwe tworzenie animacji wykresów. Każde polecenie plot jak wiadomo generuje nowe okno, zawartość tego okna można przy pomocy polecenie **getframe** zapisać np.: w tablicy. Utworzoną tablicę odtworzamy wykorzystując polecenie **movie**. Poniższy krótki skrypt ukazuje zasadę działania tych dwóch poleceń:

```
for k = 1:16
    plot(fft(eye(k+16)))
    axis equal
    M(k) = getframe;
end

movie(M,30)
```

3.6 Zadania

1. Narysować standardowe wykresy dla następujących funkcji:
   - $y = x^2 - 4$, $-5 \leqslant x \leqslant 5$
   - $y = \cos(x)$, $-2\pi \leqslant x \leqslant 2\pi$
   - $y = \sin(x)$, $-2\pi \leqslant x \leqslant 2\pi$
   - $y = \mathrm{tg}(x)$, $-2\pi \leqslant x \leqslant 2\pi$
2. Na jednym wykresie umieścić wykres funkcji sin oraz cos. Pierwszy z wykresów niech będzie narysowany linią czerwoną, drugi niebieską. Ponadto umieścić na wykresie wszystkie przydatne informacje jak legenda, tytuł wykresu, zakresy liczbowe na osi wykresów i etc.
3. W jednym oknie narysować wykresy funkcji (sam wykres funkcji przedstawić za pomocą różnych kolorów, wykorzystać polecenie **hold on**) w przedziale $x \in (0, 10)$:
   - $f(x) = x^2 - 2x$
   - $f(x) = 5\sin(x)$
4. W jednym oknie narysować następujące funkcje:
   - $\mathcal{O}(1)$, $\mathcal{O}(\lg n)$, $\mathcal{O}(n)$, $\mathcal{O}(n \lg n)$, $\mathcal{O}(n^2)$, $\mathcal{O}(n^3)$, $\mathcal{O}(2^n)$, $\mathcal{O}(n!)$
5. Narysować wykresy funkcji (wykorzystać polecenie **subplot**) w układzie biegunowym przy pomocy ciągłej linii, kropek oraz kresek:
   - $y(t) = \sin(t)$, $-2\pi \leqslant x \leqslant 2\pi$
   - $y(t) = \sin(2t)$, $-2\pi \leqslant x \leqslant 2\pi$
   - $y(t) = \sin(2t)\cos(2t)$, $-2\pi \leqslant x \leqslant 2\pi$
6. Narysować wykresy dla następujących funkcji:
   - $y(t) = 1 - 2e^{-t}\sin(t)$, $0 \leqslant t \leqslant 8$
   - $y(t) = e^{-\alpha t}\sin(\frac{t}{2})$, $0 \leqslant t \leqslant 80$, $\alpha = 0.055$
   - $y(t) = 5e^{-0.2t}\cos(0.9t - \frac{\pi}{6}) + 0.8e^{-2t}$, $0 \leqslant t \leqslant 30$
7. Narysować wykresy następujących funkcji:
   - $z(x, y) = x^2 + y + 2$, dla $-2 \leqslant x \leqslant 2$, $-2 \leqslant y \leqslant 2$
   - $z(x, y) = \frac{1}{(x+1)^2+(y+1)^2+1} - \frac{1.5}{(x-1)^2+(y-1)^2+1}$ dla $-5 \leqslant x \leqslant 5$, $-5 \leqslant y \leqslant 5$
8. Narysować krzywe parametryczne za pomocą polecenia **plot3**:
   - (t^2, t, t^3)
   - $(\sin(t), \cos(t), t)$
9. Na rysować „Ślimaka Pascala” (np.: o parametrach $a = 2$, $b = 0.5$) o wzorze:
$$r = a\cos(\varphi) + b$$
10. Opracować skrypt do badania krzywych Lissajous opisanych w następujący sposób:
$$x = A_x \cos(\omega_x t + \delta)$$
$$y = A_y \cos(\omega_y t + \phi)$$
Gdzie przez A_x, A_y oznaczono amplitudy drgań wzdłuż osi x i y, natomiast ω_x,ω_y to kołowe częstości drgań a zmienna t oznacza czas. Greckie litery δ oraz ϕ to różnica faz między drganiem pionowym a poziomym.

11. Utworzyć krótką animacje krzywych Lissajous dla wybranych parametrów (*opcjonalnie* – utworzyć film w formacie avi korzystając min. z polecenia **avifile**).

12. Utworzyć skrypt o następującej treści:

```
k = 5;
n = 2^k-1;
theta = pi*(-n:2:n)/n;
phi = (pi/2)*(-n:2:n)'/n;
X = cos(phi)*cos(theta);
Y = cos(phi)*sin(theta);
Z = sin(phi)*ones(size(theta));
colormap([0 0 0;1 1 1])
C = hadamard(2^k);
surf(X,Y,Z,C)
axis square
```

    Dokonać eksportu otrzymanego wykresu do programu *Microsoft Word* w postaci bitmapy oraz formatu wektorowego. Jak również zapisać, na dysk wykres w postaci pliku wektorowego i bitmapy.

13. Matlab oferuje polecenie **ezplot**. Czym się ono różni od standardowego polecenia **plot**?

14. W jaki sposób generowane są wartości (i dlaczego w ten sposób) przez polecenie **meshgrid**?

4 Wstęp do programowania, tworzenie skryptów

4.1 Wstęp

W Matlab oferuje język skryptowy zawierający typowe instrukcje jak `if`, `while`, `for`. Programy w Matlab'ie przyjmują dwie postacie: skryptów które można nazywać listą instrukcji do wykonania oraz m-funkcji reprezentujące sobą instrukcje zapisane w postaci funkcji przyjmującej argumenty oraz zwracającej wartości. W odróżnieniu od innych języków programowania funkcje w Matlab'ie mogą zwracać więcej niż jedną wartości.

4.2 Instrukcja warunkowa

Ogólny zapis syntaktyczny instrukcji warunkowej przedstawia się następująco:

```
if expression
  statements
elseif expression
  statements
else
  statements
end
```

W instrukcji w opisie warunku stosujemy następujące operatory relacji: `==`, `<`, `>`, `<=`, `>=`, `~=`. Operatory logiczne są następujące: `&&` (and), `||`, (or), `~` (not).

4.3 Instrukcja wielokrotnego wyboru

Zapis instrukcji wielokrotnego wyboru, czyli instrukcji odpowiadającej konstrukcji `case` z języka Pascal albo `switch` z języka C przedstawia się następująco:

```
switch switch_expr
  casa case_expr,
     statement, ..., statement
  casa {case_expr1, case_expr2, case_expr3,...}
```

```
    statement, ..., statement
  ...
  otherwise,
    statement, ..., statement
end
```

4.4 Pętla for

Ogólny zapis syntaktyczny jest następujący:

```
for variable = expr,
    statement
    ...
    statement
end
```

Bardzo często pętla `for` korzysta z wyrażenia zakresu. Czego przykładem są następujace linie kodu, gdzie wyświetlamy liczby od jedności do dziesięciu:

```
for i=1:10,
    disp(i)
end
```

4.5 Pętla while

Zapis syntaktyczny pętli while jest następujący:

```
while expression
         statements
end
```

4.6 Funkcje w Matlabie

Funkcje są definiowana w oddzielnych plikach np.: poniższy kod jest zawarty w jednym pliku:

```
function [mean,stdev] = stat(x)
n = length(x);
mean = avg(x,n);
stdev = sqrt(sum((x-avg(x,n)).^2)/n);
```

Inny przykład jest następujący:

```
function ret_val=fhad2()
ret_val=(1/sqrt(2)) ./ [1 1 ; 1 -1];
```

Trywialny przykład funkcji które nie przyjmuje argumentów i nie zwraca żadnych wartości:

```
function []=moja_fnc()
disp('moja super funkcja')
```

4.7 Zadania

1. Napisać skrypt Matlaba, który zmiennej a przypisze wartość 11. Wywołać ten skrypt i sprawdzić poprawność wyniku
2. Napisać skrypt który każdemu elementowi $a_{i,j}$ macierzy A przypisze wartość $\frac{i+j}{2i}$. Wskazówka: użyć funkcji *size*.
3. Napisać funkcję o nazwie *zwieksz* przyjmującą jeden argument x i zwracającą argument powiększony o jeden (tj. zwracającą $x + 1$).

4. Uzupełnić funkcję z zadania 3 o pomoc następującej treści:

```
ZWIEKSZ Zwieksza swoj argument o jeden.
   Funkcja ZWIEKSZ(x) wykonuje dzialanie ZWIEKSZ(x)->x+1.
 Przyklad:
   ZWIEKSZ(2) = 3
```

5. Napisać funkcję *delta* wyznaczającą wyróżnik Δ trójmianu kwadratowego postaci $y = ax^2 + bx + c$ wg. wzoru $\Delta = b^2 - 4ac$. Funkcję uzupełnić o opis „Pomocy" wg. przykładu z zadania 4.

6. Napisać funkcję do wyznaczania zer trójmianu kwadratowego przy wykorzystaniu funkcji *delta* z zadania 5. Wykorzystać wzór: $x_{1,2} = \frac{-b \pm \sqrt{\Delta}}{2a}$

7. Napisać skrypt, którego zadaniem będzie zapytanie użytkownika o współczynniki a, b, c trójmianu kwadratowego postaci $y = ax^2 + bx + c$ a następnie wyświetlenie wszystkich rzeczywistych zer tego trójmianu (jeśli istnieją) lub informacji o ich braku. W programie wykorzystać funkcję *delta* z zadania 5. Skrypt ten ma działać aż do wprowadzenia współczynnika a równego 0. Wskazówka: użyć funkcji *input*

8. Napisać funkcję który każdemu elementowi $a_{i,j}$ macierzy A przypisze wartość $\frac{i+j}{2i}$. Jako argument funkcji ma zostać podana tylko macierz A.

9. Napisać funkcję *wolny_kwadrat* o argumencie x (i elementach x_i) będącym wektorem N liczb wyznaczającą wartość $f(x_i) = x_i^2$ dla $i = 1, 2, 3, \ldots, N$. Zadanie wykonać przy pomocy pętli metodą „element po elemencie". Wskazówka: Liczbę elementów wektora można wyznaczyć przy pomocy funkcji *length*.

10. Napisać funkcję *szybki_kwadrat* o argumencie x (i elementach x_i) będącym wektorem N liczb wyznaczającą wartość $f(x_i) = x_i^2$ dla $i = 1, 2, 3, \ldots, N$. Zadanie wykonać bez pomocy pętli metodą tablicową.

11. Porównać czas działania funkcji *wolny_kwadrat* i *szybki_kwadrat* dla dużych wartości N. Do porównania użyć instrukcji *tic* i *toc*. Porównać wynik z uzyskanym przy pomocy polecenia $profile$

12. Porównać następujące implementacje algorytmu wyznaczającego wartość N-tej liczby Fibonacciego.
Algorytm 1:

```
function f=fibon1(n)
  %drukuje kolejne liczby
  f=zeros(n, 1);
  f( 1 )=1; f( 2 )=1;
  for k=1:n
  f( k ) = f( k - 1 ) + f( k - 2 )
end
```

Algorytm 2:

```
function f=fibon2(n)
  %drukuje te same liczby ale znacznie wolniej
  if n <= 1
    f = 1
  else
    f = f( n - 1 ) + f( n - 2 )
end
```

Do porównania użyć instrukcji *tic* i *toc*. Porównać ich działanie z instrukcją $profile$.

13. Na podstawie tablicowania funkcji `sin` sprawdzić czas wykonana się dwóch poniższych skryptów:

Skrypt m1.m:

```
tic; start = 0; stop = 2*pi; accuracy = 1e4;
h = (stop - start) / accuracy;
for i=1:accuracy
    x( i ) = ( i - 1 ) * h ;
    y( i ) = sin( x( i ) );
end
s=toc;
disp(['Czas trwania:' num2str( s )]);
```

Skrypt m2.m:

```
tic; start = 0; stop = 2 * pi; accuracy=1e4;
x = start : ( stop - start ) / accuracy : stop;
y = sin( x );
s = toc;
disp(['Czas trwania:' num2str( s )]);
```

Sprawdzić jak zmienia się czas wykonania dla różnych dokładności 1e2, 1e3, 1e4, 1e5, 1e6. Wyniki przedstawić na wykresie.

5 Błędy związane z obliczeniami numerycznymi konwersje liczb

5.1 Zadania

1. Zapisać[8] w systemie dwójkowym, ósemkowym (oktalnym) i szesnastkowym (heksadecymalnym) liczby systemu dziesiętnego:
   (a) 24
   (b) 232
   (c) 1025
   (d) $4^6 - 1$
   (e) 125,625
   (f) 0,325
2. Zmienić zapis z dziesiętnego na ósemkowy i szesnastkowym:
   (a) 16
   (b) 157
   (c) 2044
3. Dokonać następujących konwersji:
   (a) $(101101110110)_2 \longrightarrow (?)_8$
   (b) $(110101010110)_2 \longrightarrow (?)_{16}$
   (c) $(2716)_8 \longrightarrow (?)_{16}$
   (d) $(F2A)_{16} \longrightarrow (?)_8$
4. Zapisać dziesiętnie:

[8] W celu rozwiązania pierwszych ośmiu zadań, należy napisać własne funkcje Matlaba implementujace algorytmy konwersji, pomocne mogą być standardowe funkcje konwertujace Matlaba: *dec2bin*, *dec2base*, *dec2hex*, *bin2dec*, *hex2dec*, *base2dec*.

(a) $(100111)_2$

(b) $(111001001101)_2$

(c) $(77)_8$

(d) $(263)_8$

(e) $(7F)_{16}$

(f) $(F8FE)_{16}$

(g) $(1A6, E2)_{16}$

(h) $(77, 44)_8$

(i) $(111101, 101)_2$

5. Przygotować następujące tabelki dla działań:

   (a) dodawania dla systemów o podstawie równej 5 i 6

   (b) mnożenia dla systemów o podstawie równej 6 i 7

6. Wykonać następujące operacje:

   (a) $(24)_6 + (1254)_6$

   (b) $(24)_7 + (1254)_7$

   (c) $(121)_6 \cdot (335)_6$

   (d) $(121)_7 \cdot (335)_7$

7. Wykonać konwersję na liczby zmiennoprzecinkowe w formacie IEEE-754 dla przypadku liczby typu float o szerokości trzydziestu-dwu bitów:

   (a) 256, -245,25,

   (b) -183,625,

   (c) 0,4023, 10,2325.

8. Przy założeniu, że zapisujemy liczbę w pamięci operacyjnej binarnie w zapisie stałopozycyjnym ze znakiem, podać zakres liczb, które możemy przedstawić na 10 bitach. Jaki będzie zakres dla zapisu zmiennopozycyjnego przy założeniu, że mantysa ma 5 bitów?

9. Przedstawić liczbę $-245,25$ w zapisie stałopozycyjnym, a następnie zmiennnopozycyjnym przy założeniu, że baza systemu jest liczba 2. Ile wynosi minimalna liczba bitów potrzebna do przechowania tej liczby w pamięci?

10. Jaką liczbę dziesiętną reprezentują liczby maszynowe (t=4, w=2, b=2, gdzie b - baza, t- liczba cyfr mantysy, w - liczba cyfr cechy bez znaku):

    (a) (1)1101(0)10

    (b) (0)1001(0)00

    (c) (0)1111(0)11

    (d) (0)1000(1)11

    (e) (1)1001(1)01

11. W wyniku wystąpienia zjawiska zaokrąglenia liczba $0,2$ została zapisana w pamięci maszynowej jako $0,1875$. Wyznaczyć popełniony błąd bezwzględny i względny. Czy można oszacować na podstawie policzonych błędów dokładność reprezentacji zmiennoprzecinkowej użytej maszyny cyfrowej (ilość bitów mantysy)?

12. **Spontaniczna generacja cyfr nieznaczących.**
    Należy wykonać ciąg poleceń Matlaba:

```
>> format long e
>> 2.6 + 0.2
>> ans + 0.2
>> ans + 0.2
>> 2.6 + 0.6
```

Wyjaśnić przyczynę pojawienia się błędnej cyfry na najmniej znaczącym miejscu.

13. Obliczyć wartość wyrażeń: $0.1 + 0.1 + 0.1 - 0.3$ i $0.3 - 0.3$. Wyjaśnić różnice w wynikach.

14. **Arytmetyka zmiennoprzecinkowa.**
Należy porównać efekty wykonania dwóch ciągów poleceń Matlaba:

```
>> format long e
>> u=29/13
>> v=29-13*u
```

oraz

```
>> x=29/1300
>> y=29-1300*x
```

Z czego wynika różnica w wyniku dla pierwszego i drugiego przykładu? Następnie wprowadzić i zinterpretować wyniki wykonania następujących poleceń Matlaba (jaka jest przyczyna błędu?):

```
>> maks=realmax
>> maks=2*maks
>> minim=realmin
>> minim=minim/1e16
```

Czemu można wykonać bez wystąpienia błędu niedomiaru i jak zinterpretować wynik wykonania operacji:

```
>> realmin/1e14
>> realmin/1e15
```

15. **Precyzja maszynowa**
Wartość popełnionego błędu zaokrąglenia jest limitowana dostępna dla danej maszyny wartością precyzji ε, definowanej jako taka liczba ε, dla której $\forall_{\delta<\varepsilon} 1 + \delta = 1$. W programie Matlab wartość ta jest dostępna w zmiennej predefiniowanej:

```
>> eps
```

Wartość tę można również znaleźć interakcyjnie w sposób przybliżony, korzystając z definicji - startujemy od pewnej wartości poszukiwanej zmiennej, np. epsilon = 1, a następnie dzielimy ja na pół tak długo, aż po dodaniu do jedności wynik nie zmieni się (tzn. 1 + epsilon = 1). Należy napisać skrypt Matlaba wyznaczający precyzję maszynowa wykonywanych obliczeń.

16. **Błąd obcięcia, błąd względny i bezwzględny**
Przy mierzeniu pewnej wielkości uzyskano wynik $p = 32.65$. Wiedząc, iż maksymalny błąd względny tego pomiaru wynosi 0.5% znaleźć przedział, w którym zawarta jest wielkość p.

17. **Błąd obcięcia, błąd względny i bezwzględny**
Rozwinięcie funkcji $\sin(x)$ w szereg Taylora ma postać:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \dots$$

Dla małych wartości zmiennej x ($x < 1$) można aproksymować wartość funkcji $\sin(x) \approx x$ (szereg rozwinięcia ograniczamy do kilku pierwszych wyrazów). Należy napisać skrypt Matlaba rysujący wykres błędu względnego i bezwzględnego popełnianego podczas takiej aproksymacji w zakresie $x \in [-0.3..0..3]$ przy

założeniu, że bierzemy pod uwagę odpowiednio pierwsze 1, 2 i 3 człony rozwinięcia funkcji w szereg. Wskazówka – błąd bezwzględny definiujemy jako $\Delta = |\hat{\alpha} - \alpha|$, gdzie $\hat{\alpha}$ - wartość obliczona, α - wartość rzeczywista; błąd względny definiujemy jako $\delta = \frac{\Delta}{\alpha}$. Dla obcięcia rozwinięcia funkcji $\sin(x)$ do pierwszego wyrazu:

$$\Delta = x - \sin(x) = \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots \quad \delta = \frac{x-\sin(x)}{\sin(x)} = \frac{x}{\sin(x)} - 1$$

6 Przybliżone metody rozwiązywania równań nieliniowych

6.1 Wstęp

Często istnieje potrzeba rozwiązania równania postaci $f(x) = 0$. Dla niektórych funkcji (np. $f(x) = x^2 - 4x + 1$) znane są sposoby analitycznego wyznaczenia pierwiastków tego równania. przypadku większości funkcji rozwiązanie analityczne jest trudne, czasochłonne lub nawet niemożliwe. W wielu przypadkach zadowalające okazuje się użycie szybszych metod przybliżonych. Pierwiastek równania odnaleziony przy ich pomocy obarczony jest jednak pewnym (z reguły możliwym do oszacowania) błędem.

Istnieje wiele metod przybliżonego rozwiązywania równań nieliniowych. Do najpopularniejszych należą metody iteracyjne:

- bisekcji,
- siecznych (oparte o regułę falsi),
- stycznych (Newtona),
- punktu stałego

6.2 Metoda bisekcji

Niech funkcja $f(x)$ będzie funkcją ciągłą w przedziale $[a, b]$ i $f(a)f(b) < 0$. Oznacza to, że wartość funkcji f zmienia znak w tym przedziale. Ponadto równanie $f(x) = 0$ ma w przedziale $[a, b]$ co najmniej jedno rozwiązanie. Idea metody polega na połowieniu przedziału poszukiwań, za każdym razem biorąc tę część przedziału, na której wartość funkcji zmienia znak. Połowienie takie jest kontynuowane tak długo, aż nie zostanie osiągnięta określona dokładność obliczeń (oznaczana zwykle ϵ lub *eps*). Prowadzi to do następującego algorytmu, znanego jako *metoda bisekcji* lub *metoda połowienia*:

```
1) c:=(a+b)/2
2) jeśli b-c<=eps, to
     przyjmij, że pierwiastkiem równania f(x)=0 jest
     wartość c, a więc f(c)=0 i zakończ program.
3) jeśli znak(f(b))*znak(f(c))<=0 to
     a:=c
   w przeciwnym razie
     b:=c
4) skocz do punktu 1.
```

Funkcja *znak*(x) zwraca wartość 1 jeśli $x > 0$, -1 jeśli $x < 0$ oraz 0 dla $x = 0$. Szczególną zaletą metody bisekcji jest fakt, iż jest ona zawsze zbieżna do rozwiązania. Główną wadą tej metody jest wolna zbieżność do rozwiązania.

6.2.1 Oszacowanie błędu

Niech a_n, b_n, c_n oznaczają n-tą obliczoną wartość odpowiednio a, b i c. Ponadto niech α oznacza prawdziwą wartość pierwiastka równania $f(x) = 0$. Błąd popełniony w n-tym kroku w metodzie bisekcji $|\alpha - c_n|$ jest określony wzorem:

$$|\alpha - c_n| \leqslant \frac{1}{2^n}(b - a) \tag{1}$$

6.3 Metoda Newtona

Metoda Newtona, zwana też metodą stycznych, należy do metod iteracyjnych. W metodzie tej korzysta się z założenia, że funkcja f posiada ciągłą co najmniej pierwszą pochodną (oznaczoną f'). Ponadto zakłada się, że znane jest pierwsze przybliżenie x_0 pierwiastka równania $f(x) = 0$. W metodzie tej iteracyjnie wyznacza się wartość wyrażenia:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{2}$$

w pierwszym kroku oraz

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{3}$$

dla $n > 1$. Program kończy się, gdy błąd metody jest zadowalająco mały: $x_n - x_{n-1} \leqslant \epsilon$ lub wykonano zadaną ilość iteracji n_{max}. Uwaga: błędny dobór wartości początkowej x_0 może spowodować brak zbieżności metody.

6.3.1 Oszacowanie błędu

Niech pierwsza (f') i druga (f'') pochodna funkcji f będzie ciągła oraz wartość pierwszej pochodnej $f'(\alpha) \neq 0$, gdzie α jest pierwiastkiem równania $f(x) = 0$, tzn. $f(\alpha) = 0$. Można wykazać, że błąd metody Newtona wynosi

$$\alpha - x_n \approx x_{n+1} - x_n \tag{4}$$

6.4 Metoda siecznych

Zakładając, że znane są dwa początkowe oszacowania (x_0 oraz x_1) wartości α pierwiastka równania $f(x) = 0$. Przy takich założeniach możliwe jest użycie metody siecznych:

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \tag{5}$$

dla $n = 1, 2, 3, n_{max}$. Program kończy się, gdy błąd metody jest zadowalająco mały: $x_n - x_{n-1} \leqslant \epsilon$ lub wykonano zadaną ilość iteracji n_{max}. Metoda siecznych może być uważana za przybliżenie metody Newtona bazujące na fakcie, iż

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

6.4.1 Oszacowanie błędu

Oszacowanie błędu w metodzie siecznych odbywa się podobnie do metody Newtona. Błąd metody wynosi w przybliżeniu

$$\alpha - x_{n-1} \approx x_n - x_{n-1} \tag{6}$$

6.5 Porównanie metody siecznych i Newtona

Należy podkreślić, iż metoda siecznych w ogólnym przypadku wymagać będzie większej liczby iteracji od metody Newtona. Współczynnik zbieżności (informujący, jak błąd w kroku $n+1$ zależy od błędu w kroku n) dla metody siecznych wynosi ok. 1.62:

$$|\alpha - x_{n+1}| \approx c\,|\alpha - x_{n+1}|^{1.62}$$

zaś w metodzie Newtona — dokładnie 2:

$$\alpha - x_{n+1} = \left[\frac{-f''(c_n)}{2f'(x_n)}\right](\alpha - x_{n+1})^2$$

Nie należy jednak zapominać, że metoda Newtona wymaga wyliczenia zarówno wartości funkcji, jak i wartości pochodnej tej funkcji. Z tego powodu, pomimo mniejszej liczby iteracji, w porównaniu z metodą siecznych, może ona w praktyce działać wolniej. Zależy to od trudności w wyznaczeniu pochodnej funkcji w punkcie x_n.

6.6 Metody punktu stałego

Teoria punktu stałego stanowi uogólnienie iteracyjnych metod wymagających podania jednego punktu startowego. Teoria ta może więc zostać użyta do analizy metody Newtona.

Wszystkie metody iteracyjne wymagające podania jednego punktu startowego można zapisać jako:

$$x_{n+1} = g(x_n) \tag{7}$$

Jeśli

$$\lim_{n \leftarrow \infty} x_{n+1} = \lim_{n \leftarrow \infty} g(x_n) \tag{8}$$

to

$$\alpha = g(\alpha) \tag{9}$$

Stąd α jest rozwiązaniem równania $x = g(x)$. Wartość α nazywana jest **punktem stałym** funkcji g.

Można dowieść, że metoda iteracyjna oparta na wzorze (7) jest zbieżna, gdy

$$|g'(\alpha)| < 1$$

Jeśli $|g'(\alpha)| = 1$, wtedy zbieżność metody należy badać innymi metodami. Gdy natomiast $|g'(\alpha)| > 1$, to metoda jest rozbieżna.

6.7 Zestaw zadań I

Poniższe zadanie są przeznaczone do wykonania bez udziały metod implementowanych w komputerze. Można jednak wykorzystać program Matlab/Octave w roli kalkulatora.

1. Wyznaczyć wzór na minimalną liczbę iteracji n w metodzie bisekcji, która zapewni błąd oszacowania pierwiastka równania $f(x) = 0$ nie większy, niż założona wartość ϵ. Wskazówka: skorzystać z wzoru (1).
2. Za pomocą metody Newtona wyznaczyć przybliżoną wartość: $\sqrt{2}$, czyli pierwiastka równania $x^2 = 2$.
3. Wiadomo, że równanie $x^3 - x + 1 = 0$ posiada tylko jeden pierwiastek rzeczywisty (wyjaśnić, dlaczego tak jest?). Znaleźć ów pierwiastek metodą bisekcji (połowienia).
4. Metodę siecznych zastosować do funkcji $f(x) = x^2 - 2$, gdzie $x_0 = 0$ i $x_1 = 1$ oznaczają kolejne przybliżenia jednego z pierwiastków równania $f(x) = 0$. Wyliczyć wartość x_2.
5. Sprawdzić, które z poniższych wyrażeń użyte w metodzie punktu stałego gwarantują znalezienie rozwiązania równania: $x^2 - 5 = 0$:
   (a) $x_{n+1} = 5 + x_n - x_n^2$
   (b) $x_{n+1} = 5/x_n$
   (c) $x_{n+1} = 1 + x_n - \frac{1}{5}x_n^2$
   (d) $x_{n+1} = \frac{1}{2}\left(x_n + \frac{5}{x_n}\right)$
6. Korzystając z metody siecznych odszukać obydwa pierwiastki poniższego równania:

   $$x^3 + x^2 - 3x - 3 = 0$$

   Wskazówka: pierwiastek znajduje się w przedziale $(1, 2)$.

6.8 Zestaw zadań II

W poniższym zbiorze wykorzystujemy metody wbudowane w Matlaba, metody z pakietu NCM[9] oraz własnoręcznie napisane funkcje.

1. Napisać funkcje implementujące następujące metody:
   (a) bisekcji
   (b) falsi
   (c) Newtona
   (d) siecznych
2. Rozwiązać następujące równanie:
   $$x^3 - 2x - 5 = 0$$
   za pomocą pakietu *Symbolic Tools*[10], funkcji **roots** Matlaba'a oraz funkcji **fzerotx** z pakietu NCM.
3. Stosując funkcję **fzerogui** z pakietu NCM odszukać pierwiastki poniższych równań:
   (a) $x^3 - 2x - 5$, $[0, 3]$
   (b) $\sin(x)$, $[1, 4]$
   (c) $\frac{1}{x-\pi}$, $[0, 5]$
4. Znaleźć metodą połowienia pierwiastek następującego równania:
   $$x^3 + x^2 - 3x - 3 = 0$$
   Poszukiwania pierwiastka najlepiej rozpocząć w przedziale $\langle 1, 2\rangle$.
5. Równanie $2x^4 + 24x^3 + 61x^2 - 16x + 1 = 0$ ma dwa pierwiastki w otoczeniu punktu 0.1. Posługując się metoda Newtona znaleźć je.
6. Korzystając z metody Newtona znaleźć przybliżenia następujących wartości:
   (a) $\sqrt[3]{8}$
   (b) $\sqrt[3]{20}$

   Przyjąć dokładność $\varepsilon = 10^{-4}$. Zwrócić uwagę na szybkość zbieżności w zależności od dobranego punktu startowego. Wskazówka: $\sqrt[n]{c}, c \in \mathbb{R}_+$ jest rozwiązaniem równania nieliniowego $x^n - c = 0$.
7. Korzystając z metody siecznych, cięciw oraz reguły falsi, wyznaczyć rozwiązania następujących równań:
   (a) $\cos x = 0.5 - sin(x)$, $x \in (1, 2)$
   (b) $x = e^{-x}$, $x \in (0, 2)$

   Założyć dokładność rozwiązań na poziomie $\varepsilon = 10^{-6}$. Porównać skuteczność metod.
8. Wyznaczyć (jeśli to możliwe) pierwiastki poniższych równań za pomocą dowolnej z poznanych metod:
   (a) $\sin(x^2) - x^2 = 0$
   (b) $x^2 = 0$

   Wskazówka: dokładna wartość pierwiastka obu równań wynosi $\alpha = 0$.

[9] Pakiet jest dostępny pod adresem: http://www.mathworks.com/moler/
[10] Wskazówka: zastosować funkcje **solve** oraz **vpa**.

7 Metody rozwiązywania układów równań liniowych

7.1 Zestaw I

1. Za pomocą metody Cramera rozwiązać następujące równania:

   (a) $\begin{cases} 2x_1 - 2x_2 = 4 \\ 3x_1 + 2x_2 = 1 \end{cases}$

   (b) $\begin{cases} 2x_1 - 2x_2 = 4 \\ -x_1 + x_2 = 1 \end{cases}$

   (c) $\begin{cases} 2x_1 - 2x_2 = 4 \\ -x_1 + x_2 = -2 \end{cases}$

   (d) $\begin{cases} 2x_1 + 2x_2 - x_3 + x_4 = 7 \\ x_1 - x_2 + x_3 - x_4 = -2 \\ x_1 + x_2 + x_3 + x_4 = 10 \\ 4x_1 + 3x_2 - 2x_3 - x_4 = 0 \end{cases}$

2. Dysponujemy następującym równaniem:

   $$\begin{cases} x_1 + 2x_2 + x_3 = b_1 \\ 2x_1 + x_2 - 5x_3 = b_2 \\ - x_2 - 2x_3 = b_3 \end{cases}$$

   Rozwiązać otrzymane równanie metodą macierzową (operacja **inverse** bądź operator **./**). Następnie w miejsce wyrazów wolnych wstawić następujące kolumny wyrazów wolnych i rozwiązać powyższy układ równań ponownie dla każdego przypadku:

   (a) $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$

   (b) $\begin{pmatrix} -1/2 \\ 1/3 \\ -1/4 \end{pmatrix}$

   (c) $\begin{pmatrix} 1 \\ \sqrt{2} \\ \sqrt{3} \end{pmatrix}$

   (d) $\begin{pmatrix} 1-j \\ -2j \\ 3 \end{pmatrix}$

3. Rozwiązać równanie macierzowe:

   $$\begin{pmatrix} 2 & -3 & 0 \\ 3 & 1 & 1 \\ -1 & 2 & 0 \end{pmatrix} \cdot \mathbf{X} = \mathbf{B}$$

   dla macierzy $\mathbf{B}$ opisanej w następujący sposób:

   (a) $\mathbf{B} = \begin{pmatrix} -3 & 0 \\ 1 & 1 \\ 2 & -1 \end{pmatrix}$

   (b) $\mathbf{B} = \begin{pmatrix} 2 & -3 & 1 & -3 & 0 \\ 3 & 1 & 0 & 1 & 1 \\ -1 & 2 & -2 & 2 & -1 \end{pmatrix}$.

4. Za pomocą metody iteracyjnej Jacobiego rozwiązać następujące równania (sprawdzić za każdym razem czy metoda będzie zbieżna):

(a) $\begin{cases} 4x_1 - x_2 + x_3 = 7 \\ 4x_1 - 8x_2 + x_3 = -21 \\ -2x_1 + x_2 + 5x_3 = 15 \end{cases}$

(b) $\begin{cases} -2x_1 + x_2 + 5x_3 = 15 \\ 4x_1 - 8x_2 + x_3 = -21 \\ 4x_1 - x_2 + x_3 = 7 \end{cases}$

(c) $\begin{cases} 9x_1 + x_2 + x_3 = 10 \\ 2x_1 + 10x_2 + 3x_3 = 19 \\ 3x_1 + 4x_2 + 11x_3 = 0 \end{cases}$

5. Za pomocą metody iteracyjnej Gaussa-Seidela rozwiązać następujące równania (sprawdzić za każdym razem czy metoda będzie zbieżna):

(a) $\begin{cases} x_1 - 5x_2 - x_3 = -8 \\ 4x_1 + x_2 - x_3 = 13 \\ 2x_1 - x_2 - 6x_3 = -2 \end{cases}$

(b) $\begin{cases} 5x_1 - x_2 + x_3 = 10 \\ 2x_1 + 8x_2 - x_3 = 11 \\ -x_1 + x_2 + 4x_3 = 3 \end{cases}$

(c) $\begin{cases} 4x_1 + x_2 - x_3 = 13 \\ x_1 - 5x_2 - x_3 = -8 \\ 2x_1 - x_2 - 6x_3 = -2 \end{cases}$

7.2 Zestaw II

1. Stosując metodę eliminacji Gaussa najpierw "ręcznie" a potem przy użyciu "maszyny elektronowej" rozwiązać następujące równania:

(a) $\begin{cases} 2x_1 + x_2 + x_3 + x_4 = 7 \\ x_1 + x_2 + 2x_4 = 8 \\ 2x_1 + 2x_2 + 3x_3 = 10 \\ -7x_1 - x_2 - 2x_3 + 2x_4 = 0 \end{cases}$

(b) $\begin{cases} x_1 + x_2 + x_3 + x_4 = 7 \\ x_1 + 2x_4 = 5 \\ 2x_1 + 2x_2 + 3x_3 = 10 \\ -x_1 - x_2 - x_3 + 2x_4 = 0 \end{cases}$

(c) $\begin{cases} 0.9501x_1 - 0.7621x_2 + 0.6154x_3 - 0.4057x_4 - 0.0579x_5 = -1.23 \\ 0.2311x_1 + 0.4565x_2 + 0.7919x_3 - 0.3529x_5 = 12.3 \\ 0.6068x_1 - 0.9218x_3 + 0.9169x_4 + 0.8132x_5 = 0.1 \\ 0.8214x_2 - 0.7382x_3 + 0.4103x_4 - 0.0099x_5 = -0.12 \\ 0.8936x_1 - 0.1389x_4 = -2.12 \end{cases}$

2. Rozwiązać równania z poprzedniego punktu stosując algorytm Gaussa-Jordana. Spróbować porównać wydajność obydwu metod.

3. Za pomocą arytmetyki czterocyfrowej rozwiązać metodą prostej eliminacji Gaussa układ równań:

$$\begin{aligned} 0.003x_1 + 59.14x_2 &= 59.17 \\ 5.291x_1 - 6.130x_2 &= 46.78 \end{aligned}$$

Odpowiedź to : $x_2 = 1.001, x_1 = -10.00$ ale prawdziwe rozwiązanie jest następujące: $x_1 = 10.00, x_2 = 1.000$!

2. Przeanalizuj to samo zagadnienie dla dwóch następujących przypadków:

(a) $\begin{cases} 58.09x_1 & + & 1.003x_2 & = & 68.12 \\ 5.31x_1 & - & 6.100x_2 & = & 47.00 \end{cases}$

(b) $\begin{cases} 58.9x_1 & + & 0.03x_2 & = & 59.20 \\ -6.10x_1 & + & 5.31x_2 & = & 47.00 \end{cases}$

7.3 Zadania do wykonania w domu

1. Zaimplementować w Matlabie funkcję do rozwiązywania układów równań macierzowych w postaci:

$$Ax = b$$

metodą:

(a) Cramera

(b) macierzową (metoda oparta o macierz odwrotną)

Programy powinny posiadać zabezpieczenia przed podaniem niepoprawnych danych wejściowych: macierzy A oraz wektora b. Uwzględnić przypadek gdy $det(A) = 0$.

8 Interpolacja

8.1 Interpolacja liniowa

Jest to najprostszy przypadek interpolacji. Poniższy wzór został tak skonstruowany aby w punktach węzłowych wartość błędu była równa zero:

$$y = P(x) = y_0 + (y_1 - y_0)\frac{x - x_0}{x_1 - x_0} \tag{10}$$

Wzór ten jest wygodniej zapisywać w następującej postaci:

$$y = P_1(x) = y_0\frac{x - x_1}{x_0 - x_1} + y_1\frac{x - x_0}{x_1 - x_0} \tag{11}$$

8.2 Wielomian Lagrange

Wielomian Lagrange stanowi uogólnienia interpolacji liniowej dla wielomianów dowolnego stopnia. Zapisujemy go w następujący sposób:

$$P_N(x) = \sum_{k=0}^{N} y_k L_{N,k}(x) \tag{12}$$

Gdzie symbol $L_{N,k}(x)$ oznacza współczynniki wielomianu dla podanych węzłów:

$$L_{N,k}(x) = \frac{(x - x_0)\dots(x - x_{k-1})(x - x_{k+1})\dots(x - x_N)}{(x_k - x_0)\dots(x_k - x_{k-1})(x_k - x_{k+1})\dots(x_k - x_N)} \tag{13}$$

Możemy to zapisać w przy zastosowaniu symbolu produktowego, co skróci zapis:

$$L_{N,k}(x) = \frac{\prod_{j=0,j\neq k}^{N}(x - x_j)}{\prod_{j=0,j\neq k}^{N}(x_k - x_j)} \tag{14}$$

8.3 Oszacowanie błędu interpolacji

Interesuje nasz błąd ε dla punktów leżących wewnątrz przedziału określamy w następujący sposób:

$$\varepsilon(x) = f(x) - P_N(x) \tag{15}$$

Bowiem wartość błędu dla węzłów wynosi zero. Możemy skorzystać z następującej zależności:

$$|f(x) - P_N(x)| \leqslant \frac{M_{n+1}}{(n+1)!}|\omega_x(x)| \tag{16}$$

Symbolem M_{n+1} oznaczamy kres górny $(n+1)$-szej pochodnej interpolowanej funkcji:

$$M_{n+1} = \sup_{x\in\langle a,b\rangle}|f^{(n+1)}(x)| \tag{17}$$

Natomiast pod symbolem $\omega_x(x)$ ukrywa się następujące wyrażenie:

$$\omega_n(x) = (x - x_0)(x - x_1)\dots(x - x_n) \tag{18}$$

8.4 Wzór interpolacyjny Newtona

Wielomian interpolacyjny jest dany następującym wzorem:

$$P_N(x) = a_0 + a_1(x - x_0) + \ldots + a_N(x - x_0)(x - x_1)\dots(x - x_{N_1}) \tag{19}$$

Przez a_k oznaczamy ilorazy różnicowe funkcji: $a_k = f[x_0, x_1, ..., a_k]$, gdzie $k = 0, 1, 2, \dots, N$.

Wzory na iloraz różnicowy funkcji dwóch wartości. czyli iloraz różnicowy pierwszego rzędu są następująco zdefiniowane:

$$\begin{aligned} f(x_0; x_1) &= \frac{f(x_1) - f(x_0)}{x_1 - x_0} \\ f(x_1; x_2) &= \frac{f(x_2) - f(x_1)}{x_2 - x_1} \\ &\cdots\cdots\cdots\cdots\cdots \\ f(x_{n-1}; x_n) &= \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}} \end{aligned}$$

W podobny sposób określone są ilorazy drugiego rzędu:

$$\begin{aligned} f(x_0; x_1; x_2) &= \frac{f(x_1; x_2) - f(x_0; x_1)}{x_2 - x_0} \\ &\cdots\cdots\cdots\cdots\cdots \\ f(x_{n-2}; x_{n-1}; x_n) &= \frac{f(x_{n-1}; x_n) - f(x_{n-2}; x_{n-1})}{x_n - x_{n-2}} \end{aligned}$$

Ogólnie wzór na dowolny rząd ilorazy różnicowego funkcji jest następujący:

$$f(x_i; x_{i+1}; \dots; x_{i+n}) = \frac{f(x_{i+1}; x_{i+2}; \dots; x_{i+n}) - f(x_i; x_{i+1}); \dots; x_{i+n-1}}{x_{i+n} - x_i} \tag{20}$$

Do wyznaczanie ilorazów różnicowych pomocą może być następująca tabela:

x_i	$f(x_i)$	rzędu 1	rzędu 2	rzędu 3	rzędu 4	rzędu 5
x_0	$f(x_0)$					
		$f(x_0; x_1)$				
x_1	$f(x_1)$		$f(x_0; x_1; x_2)$			
		$f(x_1; x_2)$		$f(x_0; x_1; x_2; x_3)$		
x_2	$f(x_2)$		$f(x_1; x_2; x_3)$		$f(x_0; x_1; x_2; x_3; x_4)$	
		$f(x_2; x_3)$		$f(x_1; x_2; x_3; x_4)$		$f(x_0; x_1; x_2; x_3; x_4; x_5)$
x_3	$f(x_3)$		$f(x_2; x_3; x_4)$		$f(x_1; x_2; x_3; x_4; x_5)$	
		$f(x_3; x_4)$		$f(x_2; x_3; x_4; x_5)$		
x_4	$f(x_4)$		$f(x_3; x_4; x_5)$			
		$f(x_4; x_5)$				
x_5	$f(x_5)$					

Dla funkcji $f(x) = x^3$ tablica ilorazów różnic jest następująca:

x_i	$f(x_i)$	$f(x_i; x_{i+1})$	$f(x_i; x_{i+1}; x_{i+2})$	$f(x_i; x_{i+1}; x_{i+2}; x_{i+3})$	$f(x_i; x_{i+1}; x_{i+2}; x_{i+3}; x_{i+4})$
0	0				
		4			
2	8		5		
		19		1	
3	27		10		0
		49		1	
5	125		14		
		91			
6	216				

8.5 Interpolacja za pomocą funkcji sklejanych

Podobnie jak w interpolacji liniowej w podobny sposób można podać wzór na interpolację za pomocą funkcji sklejanych za pomocą funkcji liniowych, gdzie $d_k = (y_{k+1} - d_k)/(x_{k+1} - x_k)$:

$$S(x) = \begin{cases} y_0 + d_0(x - x_0) & x \in [x_0, x_1] \\ y_1 + d_1(x - x_1) & x \in [x_1, x_2] \\ y_2 + d_2(x - x_2) & x \in [x_2, x_3] \\ \vdots & \vdots \\ y_k + d_k(x - x_k) & x \in [x_k, x_{k+1}] \\ \vdots & \vdots \\ y_{N-1} + d_{N-1}(x - x_{N-1}) & x \in [x_{N-1}, x_N] \end{cases}$$

Krzywą interpolująca sklejaną nazywamy naturalną krzywą interpolującą (ang. *natural cubic spline*) wykorzystującą wielomiany stopnia trzeciego, gdy spełnione są następujące warunki:

- $s(x)$ jest wielomianem stopnia $\leqslant 3$ na każdym przedziale $[x_{j-1}, x_j]$ dla $j = 2, 3, ..., n$.
- $s(x)$, $s^{'}(x)$,$s^{''}(x)$, są ciągłe ma przedziale $a \leqslant x \leqslant b$
- $s^{''}(x_1) = s^{''}(x) = 0$

Uzyskanie krzywej dla n węzłów rozpoczniemy od uzyskania wartości współczynników M_j według następującego wzoru:

$$\frac{x_j - x_{j-1}}{6} M_{j-1} + \frac{x_{j+1} - x_{j-1}}{3} M_j + \frac{x_{j+1} - x_j}{6} M_{j+1} = \frac{y_{j+1} - y_j}{x_{j+1} - x_j} - \frac{y_j - y_{j-1}}{x_j - x_{j-1}}$$

Gdzie $j = 2, 3, ..., n-1$, zakładamy również, że $M_1 = M_n = 0$. Poszczególne wielomiany otrzymujemy wykorzystując następującą relację:

$$s(x) = \frac{(x_j - x)^3 M_{j-1} + (x - x_{j-1})^3 M_j}{6(x_j - x_{j-1})} + \frac{(x_j - x) y_{j-1} + (x - x_{j-1}) y_j}{x_j - x_{j-1}} - \frac{1}{6}(x_j - x_{j-1})[(x_j - x) M_{j-1} + (x - x_{j-1}) M_j]$$

8.6 Zadania

1. Stosując interpolację liniową wyznaczyć wielomian interpolacyjny dla funkcji $f(x) = \sqrt{x}$. Określić błąd średniokwadratowy dla otrzymanego wielomianu dla przedziału $\langle 1, 4 \rangle$.
2. Podać wielomian interpolacyjny (Lagrange'a, Newtona), jeśli dane są następujące węzły: $(-2, 3)$, $(1, 1)$, $(2, -3)$, $(4, 8)$.
3. Stosując metody Lagrange'a i Newtona zbudować wielomian interpolacyjny 4-go stopnia dla następującej tablicy:

x	0.0	0.1	0.3	0.6	1.0
$f(x)$	-6.00000	-5.89483	-5.65014	-5.17788	-4.28172

4. Z jaką dokładnością można oszacować wartość $\sqrt{2}$ wielomianem Lagrange dla następujących punktów węzłowych: 1, 25/16, 16/9, 9/4. Wykorzystać wzór 16.

5. Ocenić dokładność z jaką można obliczyć wartość $\ln 100.5$ przy zastosowaniu wzoru interpolującego Lagrange'a. Dane są następuję wartości węzłowe: $\ln 100$, $\ln 101$, $\ln 102$, $\ln 103$.

6. Dane sa 3 punkty $(1, 0)$, $(3, 3)$, $(4, 1)$. Znaleźć wielomian posiadający w każdym z nich punkt ekstremalny lub punkt przegięcia. Jaki jest minimalny stopień takiego wielomianu?

7. Populacja ludności w USA od 1930 do 1980 wynosiła (w mln):

rok	1930	1940	1950	1960	1970	1980
ludność	123.203	131.669	150.697	179.323	203.212	226.505

   Wyznaczyć wielomian interpolacyjny Lagrange'a 5-go stopnia używając powyższych danych. Wykorzystaj wyznaczony wielomian do oceny populacji w latach 1920, 1965 i 2002. Co można powiedzieć o dokładności oceny populacji w 1965 oraz w 2002 roku?

8. Dla następujących węzłów: $(0, 0), (1, 0.5), (2, 2.0), (3, 1.5)$ wyznaczyć naturalną krzywą sklejaną.

9. Poniższa tabela prezentuje wartości temperatury (w stopniach Fahrenheita) w Los Angeles. Wyznaczyć naturalną krzywą składaną i narysować jej wykres:

Czas	Stopnie	Czas	Stopnie
1	58	7	57
2	58	8	58
3	58	9	60
4	58	10	64
5	57	11	67
6	57	12	68

9 Aproksymacja

9.1 Definicje norm

Istnieje wiele definicji norm dla błędów. Oto trzy najszerzej stosowane:

$$\text{Błąd maksymalny}: \ E_\infty(f) = \max_{1 \leqslant k \leqslant N} \{|f(x_k) - y_k|\} \tag{21}$$

$$\text{Błąd średni}: \ E_1(f) = \frac{1}{N} \sum_{k=1}^{N} |f(x_k) - y_k| \tag{22}$$

$$\text{Błąd średniokwadratowy}: \ E_2(f) = \frac{1}{N} \sum_{k=1}^{N} |f(x_k) - y_k|^2 \tag{23}$$

9.2 Wielomian aproksymacyjny stopnia pierwszego

Gdy mamy n węzłów szukamy funkcji liniowej o następującej postaci:

$$y = Ax + B \tag{24}$$

Współczynniki A i B można wyznaczyć rozwiązując następujący układ dwóch równań (tzw. układ normalny):

$$\begin{aligned} \left(\sum_{k=1}^{n} x_k^2\right) A + \left(\sum_{k=1}^{n} x_k\right) B &= \sum_{k=1}^{n} x_k y_k \\ \left(\sum_{k=1}^{n} x_k\right) A + nB &= \sum_{k=1}^{n} y_k \end{aligned} \tag{25}$$

9.3 Aproksymacja wielomianem dowolnego stopnia

Uogólniony wzór na układ normalny z którego można wyznaczyć współczynniki wielomianu aproksymacyjnego dowolnego stopnia dla zbioru (x_j, y_j) o n elementach prezentuje się następująco:

$$\sum_{i=0}^{m}\left(a_i\sum_{j=0}^{n}x_j^{i+k}\right)=\sum_{j=0}^{n}y_jx_j^k,\quad k=0,1,2,...,m \tag{26}$$

9.4 Aproksymacja wielomianem tygonometrycznym

Poniższe zależności są określone dla parzystej liczby węzłów. Wielomian interpolacyjny rzędu m dla n węzłów jest dany następującym wzorem:

$$T_m(x)=\frac{a_0}{2}+\sum_{j=1}^{m}(a_j\cos(jx)+b_j\sin(jx)) \tag{27}$$

Współczynniki a_j oraz b_j (zwane także współczynnikami Fouriera) wyznaczamy według następujących wzorów:

$$\begin{aligned}a_j&=\tfrac{2}{n}\textstyle\sum_{k=1}^{n}f(x_k)\cos(jx_k)\ \ j=0,1,2,...,m\\ b_j&=\tfrac{2}{n}\textstyle\sum_{k=1}^{n}f(x_k)\sin(jx_k)\ \ j=1,2,...,m\end{aligned} \tag{28}$$

9.5 Ortogonalny wielomian aproksymacyjny – wielomiany Grama

Wielomiany tego typu oferują najlepszy w sensie aproksymacji średnikwadratowej wielomian przybliżający daną funkcję. Wielomian aproksymacyjny dla m równo odległych węzłów ma następującą postać:

$$P_m(x)=\sum_{k=0}^{m}\frac{c_k}{s_k}\hat{F}_k^{(n)}\left(\frac{x-x_0}{h}\right) \tag{29}$$

Oznaczenie $\hat{F}$ określa wielomiany Grama:

$$\hat{F}_k^{(\bar{n})}(q)=\sum_{s=0}^{k}(-1)^s\begin{pmatrix}k\\ s\end{pmatrix}\begin{pmatrix}k+s\\ s\end{pmatrix}\frac{q(q-1)\dots(q-s+1)}{n(n-1)\dots(n-\bar{n}+1)} \tag{30}$$

Działają one na $n+1$ węzłach a zmienna k przyjmuje następujące wartości: $k=0,1,2,...,m$ Współczynniki s_k i c_k określamy następująco:

$$\begin{aligned}c_k&=\textstyle\sum_{i=0}^{n}y_i\hat{F}_k^{(n)}(x_i)\\ s_k&=\textstyle\sum_{q=0}^{n}[\hat{F}_k^{(n)}(q)]^2\end{aligned} \tag{31}$$

9.6 Aproksymacja wielomianem Chebyszewa

Wielomian aproksymujący Czebyszewa stopnia n na przedziale $\langle -1,1\rangle$ jest określony jako następująca suma:

$$P_N(x)=\sum_{j=0}^{n}c_jT_j(x) \tag{32}$$

Oznaczeniu $T_j(x)$ oznacza odpowiedni wielomian Chebyszewa (definicja znajduje się w wykładzie dot. interpolacji). Współczynniki c_j są określone w następujący sposób dla wyznaczenia wartości c_0 korzystamy z poniżej relacji :

$$c_0=\frac{1}{n+1}\sum_{k=0}^{n}f(x_k) \tag{33}$$

Pozostałe wartości współczynników wyznaczamy w następujący sposób:

$$c_j = \frac{2}{n+1} \sum_{k=0}^{n} f(x_k) \cos\left(\frac{j\pi(2k+1)}{2n+2}\right) \quad j = 1, 2, 3, \ldots, n \tag{34}$$

Naturalnie aproksymacji dokonuje się na ściśle określonych węzłach wyznaczanych według wzoru:

$$x_k = \cos\left(\frac{\pi(2k+1)}{8}\right) \tag{35}$$

9.7 Dopasowanie funkcji $y = Ax^M$

Podobnie jak w poprzednim przypadku dysponujemy zbiorem N par $\{x_i, y_i\}$. Dla krzywej aproksymacyjnej w postaci:

$$y = Ax^M \tag{36}$$

Dla arbitralnie wybranej wartości M współczynnik A wyznaczamy według następującego wzoru:

$$A = \frac{\sum_{k=1}^{N} x_k^M y_k}{\sum_{k=1}^{N} x_k^{2M}} \tag{37}$$

9.8 Dopasowanie funkcji $y = Ce^{Ax}$

Dopasowanie danych do następującej funkcji wykładniczej:

$$y = Ce^{Ax} \tag{38}$$

Wymaga pewnym elementarnych przekształceń. W pierwszej kolejności należy z logarytmować obydwie strony:

$$\ln(y) = Ax + \ln(C)$$

Po wprowadzenie dodatkowych oznaczeń: $Y = \ln(y)$, $X = x$, $B = \ln(C)$. Otrzymamy liniową relację pomiędzy zmiennymi X i Y:

$$Y = AX + B \tag{39}$$

Współczynniki A, B wyznaczamy za pomocą układu (25). Natomiast współczynnik C obliczamy następująco: $C = e^B$.

9.9 Zadania

1. Wyznaczyć wielomian aproksymacyjny pierwszego stopnia (funkcja liniowa) dla następujących danych:

x_i	-1	0	1	2	3	4	5	6
y_i	10	9	7	5	4	3	0	-1

   Narysować powyższe punkty oraz wykres wielomianu aproksymacyjnego. Wyznaczyć błędy dla otrzymanej funkcji.

2. Znaleźć wielomian aproksymujący stopnia drugiego dla następujących danych:

x_i	0	0.2	0.4	0.6	0.8	1.0
y_i	1.026	0.768	0.648	0.401	0.272	0.193

3. Dokonać interpolacji wielomianem trygonometrycznym następujące dane:

x_i	$\frac{\pi}{3}$	$\frac{\pi}{2}$	$\frac{2\pi}{3}$	π	$\frac{4\pi}{3}$	$\frac{3\pi}{2}$
y_i	-1	1	-1	1	-1	1

   Dla jakiego rzędu $n = 1, 2, 3, 4$ otrzymamy najlepsze przybliżenie.

4. Wyznaczyć wielomian aproksymacyjny stopnia drugiego dla następujących danych:

x_i	1	1.5	2	2.5	3
y_i	3	4.75	7	9.75	13

Następnie wyznaczyć ortogonalny wielomian aproksymacyjny stopnia drugiego z wykorzystaniem wielomianów Grama. Porównać obydwa otrzymane wielomiany.

5. W tabeli zostały zebrane dane z pewnego eksperymentu:

Czas w [s]	Odległość w [m]
0.200	0.1960
0.400	0.7850
0.600	1.7665
0.800	3.1405
1.000	4.9075

Okazuje się, że dane z tabeli są opisane za pomocą następującej relacji: $d = \frac{1}{2}gt^2$, gdzie d jest odległością w metrach a t to czas mierzony w sekundach. Wyznaczyć wartość przyspieszenia ziemskiego g.

6. Wyznaczyć krzywą typu e^x dla następujących danych.

x_i	0	1	2	3	4
y_i	1.5	2.5	3.5	5.0	7.5

7. Wyznaczyć wielomian aproksymujący stopnia co najwyżej drugiego dla następującej funkcji $f(x) = \sin(x)$ na przedziale $\langle 0, \pi/2 \rangle$.

8. Znaleźć postać wielomianu Czebyszewa dla funkcji e^x na przedziale $\langle -1, 1 \rangle$. Zastosować cztery węzły.

10 Całkowanie numeryczne

10.1 Kwadratury Newtona-Cotesa

Kwadratury Newtona-Cotesa polegają na całkowaniu wielomianu Lagrange'a, który jest reprezentowany następująco:

$$f(x) \approx P_n(x) = \sum_{k=0}^{n} f_k L_{n,k}(x) \tag{40}$$

Ogólnie metoda całkowania Newtona-Cotesa posiada następującą postać:

$$\begin{aligned} &\int_{x_0}^{x_n} f(x)dx \approx \int_{x_0}^{x_n} P_n(x)dx = \\ &\int_{x_0}^{x_n} \left(\sum_{k=0}^{n} f_k L_{n,k}(x) \right) dx = \sum_{k=0}^{n} \left(\int_{x_0}^{x_n} f_k L_{n,k}(x)dx \right) = \\ &= \sum_{k=0}^{n} \left(\int_{x_0}^{x_n} L_{n,k}(x)dx \right) f_k = \sum_{k=0}^{n} w_k f_k \end{aligned} \tag{41}$$

Pierwsze cztery kwadratury są następujące:

$$\begin{aligned}
&\int_{x_0}^{x^1} f(x)dx \approx \tfrac{h}{2}(f_0+f_1) && \text{reguła trapezu}\\
&\int_{x_0}^{x^2} f(x)dx \approx \tfrac{h}{3}(f_0+4f_1+f_2) && \text{reguła Simpsona}\\
&\int_{x_0}^{x^3} f(x)dx \approx \tfrac{3h}{8}(f_0+3f_1+3f_2+f_3) && \text{reguła Simpsona3/8}\\
&\int_{x_0}^{x^4} f(x)dx \approx \tfrac{2h}{45}(7f_0+32f_1+12f_2+32f_3+7f_4) && \text{reguła Boole'a}
\end{aligned} \tag{42}$$

10.2 Wyprowadzenie metody Simpsona

Rozpoczynamy od zdefiniowania wielomianu Lagrange'a (lista zadań o interpolacji) stopnia drugiego:

$$P_2(x) = f_0\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f_1\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + f_2\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} \tag{43}$$

Całkujemy wielomian wyciągając przed znak całki wartości funkcji:

$$\int_{x_0}^{x_2} P_2(x) = f_0\int_{x_0}^{x_2}\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}dx + f_1\int_{x_0}^{x_2}\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}dx + f_2\int_{x_0}^{x_2}\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}dx \tag{44}$$

Aby wykonać całkowanie należy wprowadzić nowe oznaczenia, $x = x_0 + ht$ oraz $dx = h\,dt$. Zmieniamy zakres całkowania od 0 to t Ponieważ węzły są równo rozmieszczone, czyli $x_k = x_0 + kh$ co oznacza, że różnicę elementów $x_k - x_j$ możemy zastąpić przez $(k-j)h$ oraz różnicę $x - x_k$ przez $h(t-k)$. Po podstawieniu możemy wykonać dalsze przekształcenia:

$$\begin{aligned}
&\int_{x_0}^{x_2} P_2(x) \approx f_0\int_0^2 \tfrac{h(t-1)h(t-2)}{(-h)(-2h)}h\,dt + f_1\int_0^2 \tfrac{h(t-0)h(t-2)}{(h)(-h)}h\,dt + f_2\int_0^2 \tfrac{h(t-0)h(t-1)}{(2h)(h)}h\,dt =\\
&= f_0\tfrac{h}{2}\int_0^2(t^2-3t+2)dt - f_1h\int_0^2(t^2-2t)dt + f_2\tfrac{h}{2}\int_0^2(t^2-t)dt =\\
&= f_0\tfrac{h}{2}\left(\tfrac{t^3}{3}-\tfrac{3t^2}{2}+2t\right)\Big|_{t=0}^{t=2} - f_1h\left(\tfrac{t^3}{3}-t^2\right)\Big|_{t=0}^{t=2} + f_2\tfrac{h}{2}\left(\tfrac{t^3}{3}-\tfrac{t^2}{2}\right)\Big|_{t=0}^{t=2} =\\
&= f_0\tfrac{h}{2}\tfrac{2}{3} - f_1h\tfrac{-4}{3} + f_2\tfrac{h}{2}\tfrac{2}{3} = \tfrac{h}{3}(f_0+4f_1+f_2)
\end{aligned} \tag{45}$$

10.3 Złożona metoda trapezów

Znaczące polepszenie numerycznego całkowania uzyskamy, jeśli poszczególne metody zostaną zastosowane dla podprzedziałów. Załóżmy że mamy pięć punktów: $x_0, \ldots, x_4$. Gdy będziemy całkować każdy podprzedział oddzielnie do możemy zastosować np.: metodę trapezów:

$$\begin{aligned}
&\int_{x_0}^{x_4} f(x)dx = \int_{x_0}^{x_1} f(x)dx + \int_{x_1}^{x_2} f(x)dx + \int_{x_2}^{x_3} f(x)dx + \int_{x_3}^{x_4} f(x)dx \approx\\
&\approx \tfrac{h}{2}(f_0+f_1) + \tfrac{h}{2}(f_1+f_2) + \tfrac{h}{2}(f_2+f_3) + \tfrac{h}{2}(f_3+f_4) =\\
&= \tfrac{h}{2}(f_0+2f_1+2f_2+2f_3+f_4)
\end{aligned} \tag{46}$$

Błąd $E_n^T(f)$ w złożonej metodzie trapezów można wyznaczyć ze wzoru

$$E_n^T(f) = \frac{-h^2(b-a)}{12}f''(c_n)$$

gdzie c_n jest nieznanym punktem w przedziale $[a, b]$, zaś $h = (b-a)/n$. Natomiast oszacowanie błędu dla dużych wartości n określone jest wzorem

$$E_n^T(f) \approx \frac{-h^2}{12}(f'(b) - f'(a)).$$

10.4 Złożona metoda Simpsona

Podobnie jak w poprzedniej metodzie również dla metody Simpsona możemy polepszyć jakość całkowania, jeśli będziemy stosować dla mniejszych przedziałów.

$$\begin{aligned} &\int_{x_0}^{x_4} f(x)dx = \int_{x_0}^{x_2} f(x)dx + \int_{x_2}^{x_4} f(x)dx \approx \\ &\approx \tfrac{h}{3}(f_0 + 4f_1 + f_2) + \tfrac{h}{3}(f_2 + 4f_3 + f_4) = \\ &\tfrac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + f_4) \end{aligned} \tag{47}$$

Błąd $E_n^S(f)$ w metodzie Simpsona można wyznaczyć ze wzoru

$$E_n^S(f) = \frac{-h^4(b-a)}{180} f^{(4)}(c_n)$$

gdzie c_n jest nieznanym punktem w przedziale $[a, b]$, $h = (b-a)/n$, zaś $f^{(4)}$ oznacza czwartą pochodną funkcji f. Oszacowanie błędu dla dużych wartości n określone jest wzorem

$$E_n^S(f) \approx \frac{-h^4}{180}(f'''(b) - f'''(a))$$

11 Metoda Gaussa

11.1 Wstęp

Stosowanie metod opartych na kwadraturach Newtona-Cotesa niesie ze sobą pewien błąd a ponadto dla osiągnięcia odpowiedniej dokładności wymaga wykonania obliczeń w wielu punktach. Powstaje więc pytanie: Czy jest możliwe wyznaczenie przybliżonej wartości całki wykonując obliczenia dla małej liczby punktów? Ponadto, czy metoda taka może dawać idealne (pomijając błędy numeryczne obliczeń) wyniki dla pewnych klas funkcji? Odpowiedzią na te pytania jest metoda Gaussa.

11.2 Sformułowanie problemu

Załóżmy, że należy wyznaczyć wartość całki

$$I(f) = \int_{-1}^{1} f(x)dx$$

przy pomocy przybliżonej metody

$$I(f) \approx I_n(f) = \sum_{j=1}^{n} w_j f(x_j)$$

dodatkowo zakładając, że wagi w_j oraz węzły x_j będą dobrane tak, aby $I_n(f) = I(f)$ dla wszystkich wielomianów stopnia $2n-1$. Błąd musi zatem wynosić zero (a więc $I(f) = I_n(f)$) dla wszystkich **wielomianów** stopnia niższego lub równego $2n-1$. Konstrukcję metody przeprowadza się dla wielomianów postaci $f(x) = x^k$ dla $k = 0, 1, 2, 3, \ldots 2n-1$. Pozwala to na wyznaczenie niewiadomych w_j oraz x_j. Dokładność tej metody całkowania jest wprost proporcjonalna do wartości n.

Dla metod całkowania definiuje się ponadto stopień precyzji. Metoda całkowania posiada stopień precyzji równy k, jeśli daje idealnie dokładny wynik dla wszystkich wielomianów stopnia $\leqslant k$, zaś dla wielomianu stopnia $k+1$, błąd jest niezerowy. Stopień precyzji metody Gaussa stopnia n jest zatem równy $2n-1$.

11.3 Przypadek $n = 1$

$$I(f) = \int_{-1}^{1} f(x)dx \approx I_1(f) = \sum_{j=1}^{n} w_j f(x_j) = w_1 f(x_1) \tag{48}$$

Z założenia metoda ma być idealnie dokładna dla wielomianu stopnia 0, a więc $f(x) = 1$. Podstawiając $f(x) = 1$ do wzoru (48) i uwzględniając założenie o zerowym błędzie metody, otrzymuje się

$$I(f) = \int_{-1}^{1} (1)dx = I_1(f) = w_1 * 1$$

Aby równość była spełniona, należy tak dobrać współczynnik w_1, aby

$$\int_{-1}^{1} (1)dx = w_1$$

Całka po lewej stronie wynosi

$$\int_{-1}^{1} (1)dx = x|_{x=1} - x|_{x=-1} = 1 + 1 = 2$$

a więc

$$\int_{-1}^{1} (1)dx = 2 = w_1$$

Współczynnik w_1 wynosi więc 2. Pozostało wyznaczenie wartości x_1. Aby to uczynić, możliwe jest wymuszenie, aby błąd dla wielomianu stopnia 1 (a więc $f(x) = x$) był zerowy. Oznacza to, że

$$I(f) = \int_{-1}^{1} f(x)dx = I_1(f) = w_1 * f(x_1)$$

Podstawiając $f(x) = x$ oraz wcześniej wyznaczony współczynnik $w_1 = 2$

$$I(f) = \int_{-1}^{1} (x)dx = 2 * x_1$$

Całka po lewej stronie wynosi

$$I(f) = \int_{-1}^{1} (x)dx = \left.\frac{x^2}{2}\right|_{x=1} - \left.\frac{x^2}{2}\right|_{x=-1} = \frac{1}{2} - \frac{1}{2} = 0$$

a zatem

$$I(f) = 0 = 2 * x_1$$

Z tego wynika, że $x_1 = 0$.

Podsumowując, dla $n = 1$ i dowolnej funkcji f zachodzi

$$\int_{-1}^{1} f(x)dx \approx 2f(0)$$

Metoda jest idealnie dokładna dla wszystkich wielomianów stopnia $n \leqslant 1$, a więc jej stopień precyzji wynosi 1. Łatwo pokazać, że dla wielomianów wyższych stopni (np. dla x^2) metoda nie daje idealnie dokładnego wyniku.

11.4 Przypadek $n = 2$

Ponieważ $n = 2$, więc obliczenia należy przeprowadzić dla wielomianów stopnia $0, 1, 2, 3$. Podstawiając do wzoru ogólnego na całkowanie numeryczne metodą Gaussa otrzymuje się

$$I(f) = \int_{-1}^{1} f(x)dx \approx I_2(f) = \sum_{j=1}^{n} w_j f(x_j) = w_1 f(x_1) + w_2 f(x_2) \tag{49}$$

Z założenia metoda ma być idealnie dokładna dla wielomianu stopnia 0, a więc $f(x) = 1$. Podstawiając $f(x) = 1$ do wzoru (49) i uwzględniając założenie o zerowym błędzie metody, otrzymuje się

$$I(f) = \int_{-1}^{1} (1)dx = I_2(f) = w_1 * 1 + w_2 * 1$$

upraszczając

$$2 = w_1 + w_2$$

Z założenia metoda ma być również idealnie dokładna dla wielomianu stopnia 1, a więc $f(x) = x$. Podstawiając $f(x) = x$ do wzoru (49) i uwzględniając założenie o zerowym błędzie metody, otrzymuje się

$$I(f) = \int_{-1}^{1} (x)dx = w_1 * x_1 + w_2 * x_2$$

z czego wykonując całkowanie otrzymuje się

$$0 = w_1 * x_1 + w_2 * x_2$$

Z założenia metoda ma być również idealnie dokładna dla wielomianu stopnia 2, a więc $f(x) = x^2$. Podstawiając $f(x) = x^2$ do wzoru (49) i uwzględniając założenie o zerowym błędzie metody, otrzymuje się

$$I(f) = \int_{-1}^{1} (x^2)dx = w_1 * x_1^2 + w_2 * x_2^2$$

Całka po lewej stronie wynosi

$$I(f) = \int_{-1}^{1} (x^2)dx = \frac{1}{3}x^3\Big|_{x=1} - \frac{1}{3}x^3\Big|_{x=-1} = \frac{2}{3}$$

Z tego wynika, że

$$\frac{2}{3} = w_1 * x_1^2 + w_2 * x_2^2$$

Z założenia metoda ma być również idealnie dokładna dla wielomianu stopnia 3, a więc $f(x) = x^3$. Podstawiając $f(x) = x^2$ do wzoru (49) i uwzględniając założenie o zerowym błędzie metody, otrzymuje się

$$I(f) = \int_{-1}^{1} (x^3)dx = w_1 * x_1^3 + w_2 * x_2^3$$

Całka po lewej stronie wynosi

$$I(f) = \int_{-1}^{1} (x^3)dx = \frac{1}{4}x^4\Big|_{x=1} - \frac{1}{4}x^4\Big|_{x=-1} = 0$$

Z tego wynika, że

$$0 = w_1 * x_1^3 + w_2 * x_2^3$$

Uwzględnienie wszystkich obliczeń dla $f(x) = 1, x, x^2, x^3$ prowadzi do układu równań

$$\begin{array}{rcl} 2 & = & w_1 + w_2 \\ 0 & = & w_1 x_1 + w_2 x_2 \\ \frac{2}{3} & = & w_1 x_1^2 + w_2 x_2^2 \\ 0 & = & w_1 x_1^3 + w_2 x_2^3 \end{array}$$

Można pokazać, że rozwiązaniem tego układu równań są liczby

$$\begin{array}{rcl} w_1 & = & 1 \\ w_2 & = & 1 \\ x_1 & = & -\frac{\sqrt{3}}{3} \\ x_2 & = & \frac{\sqrt{3}}{3} \end{array}$$

oraz

$$\begin{array}{rcl} w_1 & = & 1 \\ w_2 & = & 1 \\ x_1 & = & \frac{\sqrt{3}}{3} \\ x_2 & = & -\frac{\sqrt{3}}{3} \end{array}$$

Podsumowując, dla $n = 2$ i dowolnej funkcji f zachodzi

$$\int_{-1}^{1} f(x)dx \approx f(-\frac{\sqrt{3}}{3}) + f(\frac{\sqrt{3}}{3})$$

Metoda jest idealnie dokładna dla wszystkich wielomianów stopnia $n \leqslant 3$, a więc jej stopień precyzji wynosi 3. Łatwo pokazać, że dla wielomianów wyższych stopni (np. dla x^4) metoda nie daje idealnie dokładnego wyniku.

11.5 Metoda Gaussa wyższych stopni

Analogicznie do poprzednio opisanych przypadków, dla $n > 2$ zadanie polega na wyznaczeniu wartości $x_1, x_2, \ldots, x_n$ oraz $w_1, w_2, \ldots, w_n$ tak, aby wyrażenie

$$\int_{-1}^{1} f(x)dx = \sum_{j=1}^{n} w_j f(x_j)$$

było prawdziwe dla $f(x) = 1, x, x^2, x^3, \ldots, x^{2n-1}$ Wykonanie całkowań analogicznych dla pokazanych poprzednio prowadzi do następującego układu równań

$$\left\{ \begin{array}{rcl} 2 & = & w_1 + w_2 + \cdots + w_n \\ 0 & = & w_1 x_1 + w_2 x_2 + \cdots + w_n x_n \\ \frac{2}{3} & = & w_1 x_1^2 + w_2 x_2^2 + \cdots + w_n x_n^2 \\ 0 & = & w_1 x_1^3 + w_2 x_2^3 + \cdots + w_n x_n^3 \\ \vdots & & \vdots \\ \frac{2}{2n-1} & = & w_1 x_1^{2n-2} + w_2 x_2^{2n-2} + \cdots + w_n x_n^{2n-2} \\ 0 & = & w_1 x_1^{2n-1} + w_2 x_2^{2n-1} + \cdots + w_n x_n^{2n-1} \end{array} \right.$$

Metoda taka ma stopień precyzji równy $2n-1$. Rozwiązanie takiego równania jest zadaniem bardzo trudnym. Z tego powodu wartości x_k oraz w_k często umieszcza się w tabelach.

n	x_k	w_k
1	0.0	2.0
2	$\pm$0.5773502692	1.0
3	$\pm$0.7745966692	0.5555555556
	0.0	0.8888888889
4	$\pm$0.8611363116	0.3478548451
	$\pm$0.3399810436	0.6521451549
5	$\pm$0.9061798459	0.2369268851
	$\pm$0.5384693101	0.4786286705
	0.0	0.5688888889
6	$\pm$0.9324695142	0.1713244924
	$\pm$0.6612093865	0.3607615730
	$\pm$0.2386191861	0.4679139346
7	$\pm$0.9491079123	0.1294849662
	$\pm$0.7415311856	0.2797053915
	$\pm$0.4058451514	0.3818300505
	0.0	0.4179591837
8	$\pm$0.9602898565	0.1012285363
	$\pm$0.7966664774	0.2223810345
	$\pm$0.5255324099	0.3137066459
	$\pm$0.1834336425	0.3626837834

11.6 Przystosowanie metody Gaussa dla dowolnych przedziałów

Metoda Gaussa nadaje się wyłącznie do wyznaczania całek w przedziale $[-1,1]$. Znacznie częściej w zastosowaniach praktycznych konieczne jest jednak wyznaczenie wartości całki

$$\int_a^b f(x)dx$$

Stosując podstawienie

$$x = \frac{b+a+t(b-a)}{2}$$

dla $t \in [-1,1]$. Prowadzi to do całki w postaci:

$$\frac{b-a}{2}\int_{-1}^{1} f\left(\frac{b+a+t(b-a)}{2}\right)dt$$

która może zostać wyznaczona przy pomocy metody Gaussa.

11.7 Zadania

1. Wyznaczyć wzory dla pierwszych czterech rzędów kwadratury Newtona-Cotesa.
2. Wyznaczyć wzory dla metod złożonych dla pierwszych czterech rzędów kwadratury Newtona-Cotesa.
3. Stosując kwadratury Newtona-Cotesa stopnia 1 - 4 wyznaczyć wartości całek oznaczonych z następujących funkcji:
   (a) $3x^3 - 1$
   (b) $\sin(x)$

(c) e^{-x}

Przedział całkowania można być dowolny ale wspólny dla wszystkich trzech funkcji. Ocenić na podstawie reszty kwadratury oraz wartości wyznaczanej analitycznie zależność dokładności obliczeń od stopnia kwadratury.

4. Zastosować metody złożone trapezu oraz Simpsona dla wymienionych w poprzednim punkcie funkcji. Czy wyniki uległy poprawie?

5. Zaimplementować metodę Simpsona

6. Metoda prostokątów polega na zastąpieniu całki z funkcji f całką (polem pod wykresem) z prostokąta o wysokości $f(x_1)$ i szerokości $(x_0 - x_1)$. Zaimplementować tą metodę.

7. Udowodnić, że metoda Gaussa stopnia 2 daje idealnie dokładny (pomijając błędy numeryczne) wynik przy całkowaniu wielomianu $f(x) = ax^2 + bx + c$, $a \neq 0$.

8. Wyznaczyć błąd metody Gaussa stopnia 2 przy całkowaniu funkcji $f(x) = x^4$.

9. Zaimplementować metodę Gaussa stopnia $1, 2, 3$ i 8 dla dowolnego przedziału całkowania.

10. Wyznaczyć korzystając z metody Gaussa stopnia $2, 3$ oraz 8 wartość całki $I(f) = \int_0^1 \frac{1}{1+x} dx$. Wskazówka: Przed wykonywaniem obliczeń zmodyfikować granice całkowania.

11. Korzystając z wyników zadań poprzednich porównać dokładność poznanych metod.