MEN: Projekt indywidualny

Studia Tutorskie

Termin oddania cz. A i B: 25.03.2024

Prezentacja wyników: 26-27.03.2024

Termin oddania cz. C, D i E: 15.04.2024

Prezentacja wyników: 16-17.04.2024

Termin oddania cz. F i G: 6.05.2024

Prezentacja wyników: 7-8.05.2024

I. WSTĘP - MATLAB

A. Konwersje liczb

W celu rozwiązania poniższych zadań należy napisać funkcje Matlaba implementujące algorytmy konwersji. Pomocne mogą być standardowe funkcje konwertujące Matlaba: `dec2bin`, `dec2base`, `dec2hex`, `bin2dec`, `hex2dec`, `base2dec`.

B. Precyzja maszyny

W programie Matlab wartość ta jest dostępna w zmiennej predefiniowanej:

```
>> eps
```

C. Szereg Taylora

Dostępne jest polecenie `taylor`, którym można wygenerować szereg Taylora. 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 najwyższa potęga jak zostanie zastosowana w rozwinięciu, `v` - oznacza niezależną zmienną, `a` - wartość wokół której wyliczany będzie szereg.

Ponieważ polecenie `taylor` zwraca funkcję, 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 ; ezplot(t)` zobaczymy na wykresie jakość przybliżenia szeregu Taylora odpowiada rzeczywistej funkcji zdefiniowanej w zmiennej `f`.

D. Liczenie pochodnych

Do wyznaczania pochodnych służy polecenie `diff`. Podobnie jak inne funkcje pakietu *Symbolic Math* wymaga ono zdefiniowania obiektów symbolicznych.

E. Rozwiązywanie równania macierzowego

Wprowadź macierz **A** oraz wektor **b** i znajdź rozwiązanie dla wektora **x** poprzez `x=A\b` (zwróć uwagę, że znak ten różni się od znaku dzielenia ”/”)

```
>> A=[5 -3 2; -3 8 4; 2 4 -9];
>> b=[10;20;9];
>> x=A\b
x =
    3.4442
    3.1982
    1.1868


>> C=A*x
```

```
C =

   10.0000

   20.0000

    9.0000


>> x=inv(A)*b

x =

    3.4442

    3.1982

    1.1868
```

Można tutaj użyć funkcji `linsolve`, która rozwiązuje układy liniowe, o ile macierz **A** ma odpowiednią strukturę (np. trójkątną, dodatnio określoną)

Przekształcanie macierzy do postaci normalnej (metdoą Gaussa-Jordana) można dokonać funkcją `rref`. Kolumna `Cr` jest rozwiązaniem **x**. Funkcję można użyć do sprawdzenia własnych wyników.

```
>> C=[A b]

C =

     5    -3     2    10

    -3     8     4    20

     2     4    -9     9


>> Cr=rref(C)

Cr =

    1.0000         0         0    3.4442

         0    1.0000         0    3.1982

         0         0    1.0000    1.1868
```

F. Faktoryzacja

- faktoryzacja LU: `[L U]=lu(A);`
- rozkład Cholesky'ego: `R=chol(A);`
- rozkład QR: `[Q,R]=qr(A);`

G. Użyteczne funkcje

Zapoznać się z funkcjami Matlaba określającymi rząd (`rank`), normę (`norm`) oraz współczynnik uwarunkowania (`cond`) macierzy. W matlabie $||x||_p$ możemy wyznaczyć za pomocą polecenia `norm(x,p)` (`norm(x)` jest tożsame z poleceniem `norm(x,2)`):

```
>> x = (1:4)/5
   x =
    0.2000    0.4000    0.6000    0.8000
>> norm1 = norm(x,1)
   norm1 =
        2
>> norm2 = norm(x,2)
   norm2 =
          1.0954
>> norminf = norm(x,inf)
   norminf =
           0.8000
```

Polecenie `cond(A,1)`, `cond(A,2)` (tożsame z `cond(A)`) oraz `cond(A,inf)` obliczają współczynnik uwarunkowania macierzy odpowiednio w normie L_1, L_2 i L_∞. Najczęściej wykorzystujemy `cond(A,1)` i `cond(A,inf)` ze względu na mniejszą złożoność obliczeniową. Normę L_2 zwykle wykorzystujemy dla macierzy o małych rozmiarach.

H. Techniki stosowane dla poprawy szybkości obliczeń

W programie Matlab istnieje kilka funkcji umożliwiających pomiar czasu wykonywania programu:

- `clock` – funkcja pobierająca czas systemowy w postaci wektora `T = [Rok Miesiąc Dzień Godzina Minuta Sekundy]`.
- `etime` – funkcja obliczająca różnicę czasu między dwiema wartościami otrzymanymi z zegara.
- tic – uruchomienie stopera.
- toc – zatrzymanie stopera i obliczenie odstępu czasowego od ostatniego wystąpienia funkcji tic.

Ponieważ program Matlab został opracowany i zoptymalizowany do pracy z macierzami, to operacje macierzowe są w nim wykonywane znacznie szybciej niż zewnętrzne pętle obliczeniowe. Starannie przemyślana wektoryzacja obliczeń może je znacznie przyspieszyć. Weźmy następujący przykład:

```
>> clear
>> tic
>> n = 0;
>> for t = 0:0.001:100
>>    n = n + 1;
>>    y(n) = sin (t);
>> end
>> toc
```

Dla tego programu uzyskano czas obliczeń ok. 2 min. Po zmianie kodu na następujący:

```
>> clear
>> tic
>> n = 0;
>> t = 0:0.001:100;
>> y = sin(t);
>> toc
```

czas obliczeń wyniósł jedynie 0.03 sek. Poza tym wstępna alokacja tablicy przyspieszyła obliczenia. W następnym przykładzie możemy zobaczyć, jak wielki wpływ na czas obliczeń ma dynamiczna zmiana wymiarów tablicy. Weźmy pod uwagę program:

```
>> clear
>> tic
>> y = zeros(1,100001);
>> n = 0;
>> for t=0:0.001:100
>>    n = n + 1;
>>    y(n) = sin(t);
>> end
>>toc
```

W tym przypadku czas obliczeń wyniósł jedynie 0.047 sek. Zaalokowanie w sposób jawny pamięci dla tablicy `y` dało znaczne przyspieszenie obliczeń w stosunku do pierwszego przykładu. I na koniec sprawdzimy, jaki efekt da jednoczesne jawne zarezerwowanie miejsca dla zmiennej `y` i zastąpienie pętli `for` podstawieniem macierzowym:

```
>> clear
>> tic
>> y = zeros(1,100001);
>> t = 0:0.001:100;
>> y = sin(t);
>> toc
```

W tym przypadku czas obliczeń wyniósł znowu 0.03 sek. Wyeliminowanie pętli for dało więc w tym przypadku ok. 50 % skrócenie obliczeń w stosunku do poprzedniego przykładu, jednak najistotniejsza była wstępna realokacja tablicy.

I. Rozwiązywanie równań nieliniowych

Zapoznać się z funkcjami wchodzącymi w skład zestawu *Symbolic Math Toolbox*): instrukcja `solve`, `roots` oraz `fzero`.

Funkcja `solve` rozwiązuje równania nieliniowe dla dowolnych równań algebraicznych wyrażonych w postaci symbolicznej:

```
>> syms x y;
>> solve('2*x^4+17*x^3+28*x^2-17*x-30','x')
   ans =
        -6
        -5/2
        -1
         1
```

W programie Matlab rzeczywiste zera funkcji można wyznaczyć wykorzystując polecenie`fzero`. W tym celu należy w skrypcie (z rozszerzeniem ‘m’) zadeklarować rozpatrywaną funkcję w sposób następujący:

```
>> function y = f(x)
>> y =
```

Następnie należy określić przybliżenie początkowe `x0`, czyli punkt wokół którego będzie poszukiwane zero, można również podać dokładność obliczeń `tol` oraz parametr `w`, który, jeżeli ma wartość niezerową, powoduje wyświetlenie pośrednich obliczeń. Składnia polecenia `fzero`, które należy napisać w oknie programu, jest następująca:

```
>> x_sol = fzero('plik',x0, tol, w)
```

gdzie plik jest to nazwa skryptu (bez rozszerzenia) zawierającego rozpatrywaną funkcję, natomiast `x0` jest to przybliżenie początkowe. Jeżeli nie zostanie podana wartość `tol` wówczas obliczenia będą wykonywane z dokładnością równą `eps` czyli `2.22*10-16`, natomiast brak parametru w powoduje pominięcie wyświetlania pośrednich obliczeń.

```
>> x_sol = fzero('sin',6)
   x_sol =
          6.2832
```

Funkcja `fzero` znajduje pierwiastek najbliższy przybliżenia początkowego; nie zwraca wszystkich pierwiastków. Chcąc odnaleźć wszystkie pierwiastki równania wielomianowego, należy użyć funkcji `roots`. Funkcja ta wymaga podania w wektorze współczynników przy potęgach (w porządku malejącym), w tym wyrazu wolnego (współczynnik przy x^0).

```
>> % Dla równania wielomianowego x^5-3x^3+x^2-9=0
>> C = [1 0 -3 1 0 -9];
>> roots(C)
   ans =
        1.9316 + 0.0000i
        0.5898 + 1.1934i
        0.5898 - 1.1934i
       -1.5556 + 0.4574i
       -1.5556 - 0.4574i
```

J. Interpolacja

Matlab pozwala interpolować dane przy użyciu funkcji sklejanych (spline) lub wielomianów Hermite'a. Wystarczy wykreślić surowe dane, a następnie skorzystać ze `spline interpolant` lub `hermite interpolant` w oknie **Basic Fitting**, które można wywołać z menu **Tools** okna graficznego.

Zapoznać się z funkcjami w Matlabie:

- `linspace(a,b,n)`: Generuje ciąg `n` równomiernie rozłożonych punktów z zakresu `[a,b]` (domyślnie `n = 100`).
- `interp1`: jednowymiarowa interpolacja danych.
- `spline`: jednowymiarowa interpolacja krzywą sklejaną trzeciego stopnia.
- `polyval(a,x)`: zwraca wartości wielomianu o współczynnikach zapisanych w wektorze `a` w punktach zapisanych w wektorze `x`.
- `polyval(a,x)`: zwraca wartości funkcji sklejanej o współczynnikach zapisanych w wektorze `a` w punktach zapisanych w wektorze `x`.

Program Matlab, dzięki swojej funkcji bibliotecznej `interp1`, umożliwia dokonanie interpolacji funkcji jednej zmiennej $y = f(x)$ w punktach x_i nie będących węzłami

```
>> yi = interp1(x, y, xi, 'metoda') %(x,y) - węzły interpolacji
```

następującymi metodami:

- ‘linear’ - interpolacja liniowa
- ‘spline’ - interpolacja funkcjami sklejanymi trzeciego stopnia
- ‘cubic’ - interpolacja wielomianami trzeciego rzędu

We wszystkich przypadkach elementy wektora z muszą stanowić ciąg rosnący, natomiast trzecią metodę należy stosować tylko dla węzłów równoodległych. W składni polecenia można pominąć nazwę metody - wówczas metodą domyślną jest interpolacja liniowa.

Istnieją jeszcze inne użyteczne funkcje:

- `interp2`: dwuwymiarowa interpolacja danych, opcjonalnie metodą najbliższego sąsiada, liniową, wielomianem trzeciego stopnia, funkcjami sklejanymi trzeciego stopnia.

- `interp3`: trójwymiarowa interpolacja danych, opcjonalnie metodą najbliższego sąsiada, liniową, wielomianem trzeciego stopnia, funkcjami sklejanymi trzeciego stopnia.
- `interpft`: interpolacja oparta na szybkiej transformacie Fouriera (FFT)

Interpolacja dokonywana jest w dwóch prostych etapach: dostarczenia listy (wektora) punktów, w których dane mają być interpolowane oraz wywołania odpowiedniej funkcji wybraną metodą interpolacji, np.:

```
>> x = .......
>> y = ......
>> xi = linspace(0,2*pi,50);
>> yi=interp1(x,y,xi,'linear');
>> plot(x,y), hold on, plot(xi,yi,'--')
```

K. Aproksymacja

W menu **Tools** okna graficznego znajduje się polecenie **Basic Fitting** pozwalające szybko dopasować wielomianowe krzywe (do dziesiątego stopnia) do danych. Oprócz tego umożliwia ono wyświetlanie reszt w punktach danych oraz obliczanie odchyleń. Ułatwia to porównywanie różnych dopasowań i wybór najlepszego. Podobnie można wykorzystać **Curve Fitting Tool**:

```
>> t=[-1 -0.5 0 0.5 1]';
>> y=[1 0.5 0 0.5 2]';
>> cftool
```

Zapoznać się z funkcjami w Matlabie

- `polyfit`: wielomianowa aproksymacja średniokwadratowa,
- `lsqcurvefit` i `lsqnonlin`: nieliniowa aproksymacja średniokwadratowa.

Funkcja

```
>> a = polyfit(x, y, r) %r - stopień wielomianu
```

dla danych wektorów `x` i `y` znajduje wektor współczynników a wielomianu stopnia `r` przybliżającego najlepiej w sensie średniokwadratowym zależność pomiędzy wartościami `x` a `y`. Dla `r = 1` otrzymuje się regresją liniową. Aby otrzymać wartości wielomianu przybliżającego należy posłużyć się funkcją `polyval`:

```
>> p = polyval(a, x)
```

która wyznacza wartości wielomianu o współczynnikach określonych wektorem `a` dla wszystkich elementów wektora `x`, zaś otrzymane wartości umieszcza w wektorze `p`.

W przypadku, gdy chodzi o dopasowanie do funkcji niewielomianowej np. $y = ae^{bx}$ lub $y = cx^d$ przekształcamy aproksymację za pomocą krzywych wykładniczych (potęgowych) na aproksymację liniową, logarytmując równania stronami. Najpierw dokonujemy przeskalowania, następnie wyznaczamy współczynniki prostej korzystając z polecenia `polyfit`. Na podstawie współczynników krzywej aproksymującej obliczamy pierwotne wartości a i b (lub c i d), pamiętając o przeliczeniu wartości y w danych punktach x zgodnie z otrzymaną zależnością.

L. Całkowanie

Matlab pozwala na całkowanie zarówno numeryczne, jak i na zmiennych symbolicznych. W *Symbolic Math Toolbox* do całkowania wyrażeń symbolicznych wykorzystuje się instrukcje:

- `int(S)` - zwraca całkę nieoznaczoną wyrażenia symbolicznego `S`.
- `int(S,v)` - zwraca całkę nieoznaczoną wyrażenia symbolicznego `S` ze względu na zmienną `v`.
- `int(S,a,b)` - zwraca całkę oznaczoną wyrażenia symbolicznego `S`, w granicach od `a` do `b`.
- `int(S,a,b,v)` - zwraca całkę oznaczoną wyrażenia symbolicznego `S`, w granicach od `a` do `b`, ze względu na zmienną `v`.

```
>> syms x y
>> s=1/((exp(x))+(exp(-x)));
>> int(s,x)
ans =
     atan(exp(x))
```

```
>> s=(x^2+1)/(x^3+3*x+1)^(1/3);
>> int(s,x,1,2)
ans =
     15^(2/3)/2 - 5^(2/3)/2
>> double(ans)
ans =
     1.5791
```

Instrukcje wbudowane, które można wykorzystać do wyznaczania całek metodami przybliżonymi są następujące:

- `quad(funkcja, a, b, tol, trace)` - całkuje wskazaną funkcję w określonych granicach `a` i `b`, na podstawie adaptacyjnej metody Simpsona niskiego rzędu (rząd jest dobierany w zależności od całkowanej funkcji). Argument `tol` określa błąd bezwzględny (wartość domyślna to 10^{-6}). Żądaną dokładność wstawiamy w formacie `[lewy koniec zakresu prawy koniec zakresu]`. Niezerowa wartość (1) argumentu `trace` pokazuje pewne obliczenia pośrednie na każdym etapie. Wizualizacja polega na tym, że dla kolejnych podziałów można odczytać rosnącą wartość całki.

- `quadl(funkcja, a, b, tol, trace)` - całkuje wskazaną funkcję w określonych granicach `a` i `b`, na podstawie adaptacyjnej metody Lobatto. jest to metoda dokładniejsza niż w przypadku `quad` - jej efektywność wzrasta jeśli funkcja podcałkowa jest funkcją gładką.

Dla powyższych instrukcji, funkcję podcałkową można zdefiniować jednym z trzech przedstawionych sposobów:

- jako ciąg znaków, z wykorzystaniem apostrofów: `'funkcja'`, wówczas `I=quad('1./x,...)`,

- jako obiekt: `F=inline('funkcja')`, wówczas `I=quad(F,...)`,
- jako uchwyt w m-pliku:

  ```
  function f=funkcja(x)
  f=1./x
  ```

  wówczas `I=quad(@funkcja,...)`

```
>> I=quad('exp(-x.^2)',1,2)
I =
    0.1353
>> J=quadl('exp(-x.^2)',1,2)
J =
    0.1353
```

Do przybliżonego całkowania można także wykorzystać inną instrukcję:

```
>> x=2:.1:4;
>> y=exp(x);
>> z=trapz(x,y)
z =
   47.2484
```

która zwraca przybliżoną wartość całki oznaczonej, wyznaczoną metodą trapezów.

Do obliczania całek niewłaściwych z definicji wykorzystujemy podstawowe wzory na całkowanie oraz pojęcie granicy `limit(s,a,inf)`

```
>> syms x y a b
>> I=int(1/x^4,x,3,a)
I =
   1/81 - 1/(3*a^3)
>> I1=limit(I,a,inf)  % całka niewłaściwa na [3,\infty] z funkcji 1/x^4
I1 =
   1/81
```

```
>> J=int(1/(3*x-5)^(1/3),a,-1)
J =
   2*(-1)^(2/3) - (3*a - 5)^(2/3)/2
>> J1=limit(J,a,-inf)  % całka niewłaściwa na [-\infty,-1]
J1 =                   % z funkcji 1/(3*x-5)^(1/3) - jest rozbieżna
    Inf - Inf*1i
```

Możemy również zbadać zbieżność całek niewłaściwych 2-go rodzaju poprzez obliczanie granicy prawo lub lewostronnych poleceniem `limit(F,x,a,'right')` lub `limit(F,x,a,'left')`:

```
>> I=int(1/sqrt(x),a,1) % całka niewłaściwa na [0,1]
I = 2 - 2*a^(1/2) % z funkcji 1/x
>> I1=limit(I,a,0,'right')  % granica prawostronna 0+
I1 = 2
```

Istnieją jeszcze inne użyteczne funkcje, np.: `dblquad`, która oblicza całki iterowane podwójne.

M. Różniczkowanie

Matlab pozwala na obliczanie pochodnej dowolnego rzędu. Instrukcje, które służą do wyznaczania pochodnych dla wyrażeń symbolicznych (*Symbolic Math Toolbox*) są następujące:

- `diff(S,'v')` - wyznaczenie pochodnej dla wyrażenia symbolicznego `S`, ze względu na zmienną `v`.
- `diff(S,n)` - wyznaczenie n-tej pochodnej dla wyrażenia symbolicznego `S`.
- `diff(S,'v',n)` - wyznaczenie n-tej pochodnej dla wyrażenia symbolicznego `S`, ze względu na zmienną `v`.

```
>> syms x
>> s1=2*sin(x^3)*cos(x);
>> diff(s1,'x',1)
ans = 6*x^2*cos(x^3)*cos(x) - 2*sin(x^3)*sin(x)
```

Odwołując się do podstawowego znaczenia pochodnej - traktując pochodną jako szybkość zmian określonej wielkości, można wyznaczyć pochodną (numeryczną) na postawie danych empirycznych wg. wzoru:

```
>> diff(y)./diff(x)  %  dy/dx, y jest zależne od x
```

II. ZADANIA (50 PKT; SKALOWANE PRZEZ 1/5)

A. Konwersje liczb. Błędy związane z obliczeniami numerycznymi

Zad. 1 (1.0 pkt) Zmienić zapis z dziesiętnego na ósemkowy i szesnastkowy:
a) 16 , b) 157 , c) 2044.

Zad. 2 (1.0 pkt) Wartość popełnianego błędu zaokrąglenia jest limitowana dostępną dla danej maszyny wartością precyzji ε. Stała ε maszyny jest to najmniejsza liczba nieujemna ε, taka że $1 + \varepsilon \neq 1$. Im mniejsza wartość ε tym większa względna precyzja obliczeń. Należy napisać własną funkcję wyznaczającą precyzję maszynową wykonywanych obliczeń.

Zad. 3 (3.0 pkt) Rozwinięcie funkcji $\sin(x)$ w szereg Maclaurina ma postać:

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

Dla małych wartości zmiennej x ($x < 1$) można aproksymować wartość funkcji $\sin(x) \approx x$ (obcięcie rozwinięcia w szereg do pierwszego wyrazu). 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.

B. Metody numerycznego rozwiązywania równań liniowych

Zad. 4 (4 pkt) Dla poniższych układów równań wyznaczyć wskaźniki uwarunkowania korzystając z różnych norm macierzowych (funkcje `norm` lub `cond`). Podane układy rozwiązać metodą eliminacji Gaussa (napisać własny skrypt **MojGauss.m**) i porównać z rozwiązaniem dokładnym (zwrócić uwagę na przyczyny ewentualnych rozbieżności):

a) $\begin{cases} x + 2y = 10 \\ 1.1x + 2y = 10.4 \end{cases}$

b) $\begin{cases} 2x + 5.999999y = 8.000001 \\ 2x + 6y = 8 \end{cases}$

c) $\begin{cases} 5x + 3y + 4z = 18 \\ 3x + z = 7 \\ 6x + 3y + 6z = 27 \end{cases}$

Zad. 6 (3 pkt) Napisać prosty skrypt, sprawdzający czy dana macierz jest dodatnio określona. Następnie pokazać, że macierz $\mathbf{A}$ jest dodatnio określona i rozwiązać układ liniowych równań $\mathbf{Ax} = \mathbf{b}$ metodą Choleskiego-Banachiewicza:

$$\mathbf{A} = \begin{pmatrix} 4 & 8 & -4 \\ 8 & 17 & -1 \\ -4 & -1 & 57 \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} -12 \\ -17 \\ 65 \end{pmatrix}$$

Zad. 7 (4 pkt) Za pomocą metody iteracyjnej Jacobiego lub metody iteracyjnej Gaussa-Seidela (napisać własny skrypt **MojJacobi.m** lub **MojGaussSeidel.m**) rozwiązać następujące równania (sprawdzić za każdym razem czy metoda będzie zbieżna):

a) $\begin{cases} 4x - y + z = 7 \\ 4x - 8y + z = -21 \\ -2x + y + 5z = 15 \end{cases}$

b) $\begin{cases} -2x + y + 5z = 15 \\ 4x - 8y + z = -21 \\ 4x - y + z = 7 \end{cases}$

Zbadać zależność szybkości zbieżności i poprawności rozwiązania od przybliżenia początkowego (przyjąć dokładności rozwiązania $\epsilon = 10^{-2}, 10^{-4}, 10^{-10}$). Sporządzić wykres relacji między błędem a liczbą iteracji oraz dokonać porównania skuteczności wykorzystywanych metod.

C. Metody numerycznego rozwiązywania równań nieliniowych

Zad. 8 (4.0 pkt) Napisać funkcję `MojaSieczna.m` realizującą metodę siecznych oraz `MojNewton.m` realizującą metodę Newtona. Następnie korzystając ze swoich funkcji wyznaczyć dwoma metodami rozwiązania dla równania:

$$x = e^{-x}, \quad x \in [0, 2].$$

Założyć dokładność rozwiązań na poziomie $\epsilon = 10^{-6}$. Porównać skuteczność metod (np. wykres błędu w zależności od liczby iteracji).

D. Interpolacja

Zad. 9 (3 pkt) Omówić zjawisko Rungego na przykładzie funkcji $f(x) = |x|$, $x \in [-1, 1]$. Rozważ węzły (wybrać kilka wartości $n = 6, 7, \ldots, 20$) równoodległe i węzły Czebyszewa. Sporządzić wykres zależności błędu bezwzględnego od n.

Zad. 10 (3 pkt) Badania naukowców wykazały, ze pewien rzadki gatunek ptaka *Ledwolotus dziwolongus* pokonuje corocznie takie same trasy, których kształt można opisać za pomocą wykresu wielomianu. W wyniku chwilowej utraty zasilania w centrum badawczym znaczna część danych dotyczących tego badania została utracona i pozostało jedynie kilka par punktów krzywej: $(0, -4)$, $(1, 3)$, $(3, 5)$, $(2, 2)$. Odtworzyć trasę lotów *ledwolotusa*, znajdując wielomian możliwie najniższego stopnia, którego wykres przechodzi przez powyższe pary punktów. Napisać skrypt Matlaba realizujący interpolację Lagrange'a (**MojaInterpLagrange.m**).

Zad. 11 (3 pkt) Pewnego dnia, w godzinach od 7:50 do 10:20 na Wydziale Mechatroniki odbywał się egzamin z przedmiotu „Metody Numeryczne". Ponieważ atmosfera, jak na każdym egzaminie, była "gorąca", zdecydowano się zmierzyć temperaturę panującą w sali w trakcie oraz po egzaminie. Pomiary były dokonywane co pełną godzinę: o godzinie 8 było 20 stopni, o 9 - 24 stopnie, o 10 - 26 stopni, a o 11 temperatura spadła z powrotem do 20 stopni Celsjusza. Ponadto, okazało się, że zmianę temperatury w sali można opisać funkcją czasu $T(t)$, która jest wielomianem trzeciego stopnia. Dokonać interpolacji metodą Newtona (**MojaInterpNewton.m**) tej funkcji i odpowiedzieć na pytanie: jaka temperatura panowała w sali o godzinie 10:30?

E. Aproksymacja

Zad. 12 (2 pkt) Dokonać aproksymacji średniokwadratowej funkcji

$$f(x) = \frac{x}{x^2 + 2}$$

wielomianem 2-go stopnia w przedziale $[-1, 1]$ z krokiem 0.01. Narysować wykres danej funkcji i funkcji przybliżającej w jednym układzie współrzędnych natomiast wykres błędu aproksymacji w drugim. Wyznaczyć maksymalną wartość bezwzględnego błędu aproksymacji w rozpatrywanym przedziale.

Zad. 13 (3 pkt) Dopasować model eksponencjalny $y(x) = a_0 e^{a_1 x}$ do danych:

x_i	0.4	0.8	1.2	1.6	2.0	2.3
y_i	750	1000	1400	2000	2700	3750

Następnie rozwiązać zadanie przez sprowadzenie do problemu regresji liniowej. Porównać rezultaty.

F. Całkowanie

Napisać własne funkcje obliczające całki z wykorzystaniem kwadratur Newtona-Cotesa

a) funkcję `MojProstokat(fun,a,b,npanel)` - do obliczania całki metodą prostokątów, gdzie `fun` jest całkowaną funkcją (podana w oddzielnym m-file'u), `a` i `b` to początek i koniec przedziału całkowania, `npanel` to liczba podprzedziałów na którą dzielimy przedział `[a,b]`.

b) funkcję `MojTrapez(fun,a,b,npanel)` - do obliczania całki metodą trapezów, gdzie `fun` jest całkowaną funkcją (podana w oddzielnym m-file'u), `a` i `b` to początek i koniec przedziału całkowania, `npanel` to liczba podprzedziałów na którą dzielimy przedział `[a,b]`.

c) funkcję `MojaParabola(fun,a,b,npanel)` - do obliczania całki metodą parabol, gdzie `fun` jest całkowaną funkcją (podana w oddzielnym m-file'u), `a` i `b` to początek i koniec przedziału całkowania, `npanel` to liczba podprzedziałów na którą dzielimy przedział `[a,b]`.

c) funkcję `MojeTrzyOsme(fun,a,b,npanel)` - do obliczania całki metodą $\frac{3}{8}$ Newtona, gdzie `fun` jest całkowaną funkcją (podana w oddzielnym m-file'u), `a` i `b` to początek i koniec przedziału całkowania, `npanel` to liczba podprzedziałów na którą dzielimy przedział `[a,b]`.

Zad. 14 (6 pkt) Obliczyć wartość całki

$$I = \int_0^5 xe^{-x}dx$$

metodą prostokątów, trapezów i parabol dla liczby węzłów wynoszącej kolejno 2, 4, 8, 16, 32. Dla każdej liczby węzłów obliczyć długość przedziałów h. Dodatkowo podać błąd każdej metody w zależności od liczby węzłów (sporządzić wspólny wykres): $E(f) = |I - \int_a^b f(x)dx|$. Dokładną wartość całki I policzyć za pomocą *Symbolic Math Toolbox*. Zbadać zależność błędu wyniku od stopnia złożoności kwadratury.

Zad. 15 (3 pkt) Zastosować metodę $\frac{3}{8}$ Newtona do obliczenia wartości całki

$$Q = \frac{2}{\sqrt{\pi}} \int_0^1 e^{-x^2}dx$$

dla liczby węzłów wynoszącej kolejno 3, 6, 9, 12. Dla każdej liczby węzłów obliczyć długość przedziałów h. Dodatkowo podać błąd metody w zależności od liczby węzłów (sporządzić wykres): $E(f) = |Q - \int_a^b f(x)dx|$. Dokładną wartość całki Q policzyć za pomocą *Symbolic Math Toolbox*.

Zad. 16 (2 pkt) Wyznacz całkę

$$U = \int_{-1}^1 \frac{x^2}{\sqrt{(1+x)(1-x)}}dx$$

za pomocą kwadratury Gaussa-Czebyszewa dla 61 węzłów. Porównaj otrzymany wynik z wynikiem otrzymanym za pomocą *Symbolic Math Toolbox*. Przyjąć arytmetykę `double`.

Zad. 17 (2 pkt) Stosując kwadraturę Gaussa-Hermite'a dla pięciu węzłów zbadaj zbieżność całki

$$V = \int_{-\infty}^{-\infty} x^2 e^{-x^2} dx.$$

Porównaj otrzymany wynik z wynikiem otrzymanym za pomocą *Symbolic Math Toolbox*. Przyjąć arytmetykę `double`.

G. Różniczkowanie

Zad. 18 (3 pkt) Napisać skrypt realizujący numeryczne obliczanie pochodnej funkcji

$$y = 2 + 2\sin^2(x) + \cos^2(x).$$

Obliczenia wykonać w zakresie $[0, 5]$. Przyjąć stały krok różniczkowania $h = 0.01$. Porównać przebieg pochodnej y wyznaczonej numerycznie z przebiegiem pochodnej y wyznaczonej analitycznie (*Symbolic Math Toolbox*).

Pliki z rozwiązaniami (MEN_Proj_Ind_Jan_Kowalski.pdf) oraz funkcjami matlabowymi należy przesłać jako niespakowane załączniki jednym listem elektronicznym o temacie **MEN PROJEKT IND** na adres:

zofia.grudziak@pw.edu.pl

W treści listu należy podać swoje imię i nazwisko.