Programowanie i metody numeryczne

Aleksandra Fijałkowska

8 czerwca 2018

Zadanie 4. (1 p)
Znaleźć epsilon maszynowy dla zmiennych typu float i double. Wynik porównać z wartością otrzymaną ze statycznej metody epsilon w szablonie std::numeric_limits.

Zadanie 5. (4 p)
a). Napisz funkcję liczącą pochodną dwupunktową metodą centralną:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h} \tag{1}$$

Sprawdzić działanie programu dla dla $f = sin(x)$ dla $x = \frac{2\pi}{3}$ oraz $f = x^3$ dla $x = 2,5$, wykorzystaj zmienne typu float.
Wskazówka: (przyda się w kolejnej części zadania)
Zaprojektuj program tak, aby każda część algorytmu znalazła się w osobnej funkcji np.:

```
float getFunValue(float x);\\
float getFunDerivative(float x);\\
float getFunDerivativeCentralMeth(float x, float h);\\
float getDerivativeError(float x, float h);\\
```

b). Zmodyfikuj program dla funkcji $f = x^3$ tak, aby w wygodny sposób umożliwiał sprawdzenie algorytmu dla zmiennych typu float i double. Funkcje przyjmujące zmienne typu float przepisz na szablony funkcji.
Wskazówka:
Przykładowy szablon funkcji, która zwraca kwadrat przyjętego argumentu:

```
template<typename T> T square(T x)
{
    return x*x;
}
```

c). Wyrysuj błąd metody w funkcji parametru h w skali log-log dla zmiennych typu float i double.

Zadanie 6. (2 p)
Napisz program sumujący liczby od 1 do 10000 zmiennymi int, float i double. Sprawdzić wynik sumując od przodu, od tyłu i metodą Kahana $https://en.wikipedia.org/wiki/Kahan\_summation\_algorithm3$.

Zadanie 7. (2 p)
Napisz program alokujący dynamicznie tablice liczb całkowitych o wczytanym rozmiarze. Następnie program powinien wyzerować podaną liczbę elementów z początku tablicy lub nie, w zależności od wyboru użytkownika. Po tym użytkownik zdecyduje, czy zwolnić zaalokowaną pamięć i następnie kiedy zakończyć program.

Zadanie 8. (3 p)
Napisz procedurę wczytującą liczby typu float z wejścia standardowego. Po wpisaniu pustej linii procedura powininna wyświetlić wczytane liczby na ekran w odwrotnej kolejności.
Wskazówka:
Najłatwiejszą metodą rozwiązania tego zadania jest skorzystania z wektora (za 2 punkty). Rozmiar wektora może być łatwo powiększany metodą push_ back. Zachęcam do zaimplementowania trudniejszego algorytmu, bazującego na tablicy (za 3 punkty). Po przekroczeniu początkowo zadanego rozmiaru tablicy powinna ona dwukrotnie zwiększać swój rozmiar.

Zadanie 9. (1 p)
Zapisz procedurę z zadania 2 przy pomocy szablonu tak, by mogła przyjmować tablicę dowolnego typu.

Zadanie 10. (3 p)
Zaimplementuj strukturę reprezentującą macierz dowolnego wymiaru (niekoniecznie kwadratową). W polach struktura powinna mieć: liczbę kolumn, liczbę wierszy oraz dynamicznie alokowaną tablicę elementów macierzowych (można użyc vector<float>).
Zaimplementuj metody: konstruktor przyjmujący liczbę wierszy i kolumn oraz dane macierzy, konstruktor tworzący macierz stałych wartości o zadanym rozmiarze, konstruktor kopiujący, metodę wyświetlającą macierz, operację transpozycji oraz operację mnożenia macierzy (metoda musi sprawdzać, czy rozmiar mnożonych macierzy się zgadza) oraz mnożenia macierzy przez liczbę. Przykładowy nagłówek struktury może wyglądać tak:

```
class Matrix
{
public:
    Matrix(int rowNr, int colNr, std::vector<float>& value);
    Matrix(int rowNr, int colNr, float constVal = 0.);
    Matrix(const Matrix &right);
    ~Matrix();
    float getElement(int rowNr, int colNr);
    void setElement(int rowNr, int colNr, float value);
    void print();
    void scale(float x);
    Matrix multiply(Matrix& B);
    Matrix getTranspose();
    int getNrOfColumns();
```

```
    int getNrOfRows();
private:
    int rowNrs;
    int colNrs;
    std::vector<float> values;
};
```

Sprawdź działanie Twojej struktury na prostych przykładach. Sprawdzenie powinno zawierać wykorzystanie każdego z konstruktorów, wyświetlanie elementów, mnożenie dwóch macierzy, transpozycja i mnożenie macierzy przez liczbę.

Wskazówka:

```
float Matrix::getElement(int rowNr, int colNr)
{
    if(rowNr > this->rowNrs || colNr > this->colNrs)
        throw std::invalid_argument("Matrix::getElement(int rowNr, int colNr):
                             input parameters exceed the size of the matrix" );
    return values.at(rowNr*colNrs + colNr);
}

void Matrix::setElement(int rowNr, int colNr, float value)
{
    if(rowNr > this->rowNrs || colNr > this->colNrs)
        throw std::invalid_argument("Matrix::getElement(int rowNr, int colNr):
                             input parameters exceed the size of the matrix" );
    values.at(rowNr*colNrs + colNr) = value;
}
```

Zadanie 11. (2 p)
Korzystając ze struktury macierzy z zadania 1 zaimplementuj metodę potęgową ($https : //en.wikipedia.org/wiki/Power\_iteration$) liczenia największej co do wartości bezwzględnej wartości własnej. Zauważ, że iloczyn skalarny wektorów możemy uzyskać mnożąc wektorowo wketor transponowany przez wektor właściwy. W ten sam sposób można wyznaczyć normę tablicy. Jako punkt startowy metody wygenerować wektor o zrandomizowanych elementach (nagłówek <random>). Iterować aż do uzyskania zadanego stopnia zbieżności wartości własnej. Procedura powinna mieć sygnaturę float poweriter(const matrix & A, float epsilon).

Procedurę sprawdz dla macierzy $A = \begin{bmatrix} 5 & -2 & -2 \\ -3 & 5 & 0 \\ 23 & -19 & -6 \end{bmatrix}$

Zadanie 12. (2 p)
Używając procedury z zadania 11 policz rozkład dominujących wartości własnych macierzy losowych postaci $\delta_{ij} + A_{ij}$, gdzie elementy A_{ij} są zmiennymi losowymi o rozkładzie normalnym. Wyniki przedstawić na histogramie dla macierzy 5x5 i 50x50.

Wskazówka:
Zaimplementuj klasę Histogram. Wydziel klasę Matrix i Histogram do osobnych plików (nagłówkowego i pliku ze źródłem), napisz Makefile dla wygodnej kompilacji.

Do rysowania histogramów w programie gnuplot można wykorzystać opcję *with boxes*. Dla przykładowego wyniku zapisanego w pliku "wynik.txt" w postaci dwóch kolumn: środek binu liczba zliczeń oraz przykładowej szerokości binu = 0.15 histogram można wyrysować następująco:

```
binwidth=0.15
bin(x,width)=width*floor(x/width)
plot 'wynik.txt' using (bin($1,binwidth)):2 smooth freq with boxes
```

Zadanie 13. (0 p)
Wyznacz wartość liczby π metodą Monte Carlo.

Zadanie 14. (2 p)
Napisz szablon funkcji, która przyjmie jako argumenty funkcję f o jednym argumencie i tablicę elementów tego samego typu x, i zwróci tablicę elementów $f(x_i)$. Zaprezentuj działanie procedury dla dwóch różnych przykładów.

Zadanie 15. (6 p)
Znajdź miejsca zerowe funkcji:
1. $y = x^5 - 10x^2 + 10x - 2$, $x \in [0, 0.5]$
2. $y = 1.1\cos(x^3) - \cos(x^3 + 0.1)$, $x \in [-1, 1]$

Porównaj algorytmy: bisekcji, Newtona, siecznych wykreślając dokładność (logarytm modułu różnicy) względem liczby iteracji.

Zadanie 16. (2 p)
Znajdź miejsca zerowe funkcji $y = x^5 - 10x^2 + 10x - 2$, $x \in [0, 0.5]$ metodą szukania punktu stałego - rozważ następujące wzory iteracyjne:
1. $x_{n+1} = \frac{x_n^5 - 10x_n^2 - 2}{-10}$
1. $x_{n+1} = \sqrt{\frac{x_n^5 + 10x_n - 2}{10}}$
1. $x_{n+1} = (10x_n^2 - 10x_n + 2)^{1/5}$
1. $x_{n+1} = \dfrac{2}{x_n^4 - 10x_n + 10}$

Zadanie 17. (2 p)
Napisz procedurę, która przybliży wartość całki

$$\int\limits_a^b f(x)dx$$

metodami trapezów i Simpsona dla zadanej liczby węzłów n. Porównaj i skomentuj wyniki dla $f(x) = \sum_{i=0}^{m} \frac{x^i}{i!}$, $m = 0, 1, 2, 3, 4$ $n = 2...8$, $a = 0$, $b = 1$.

Zadanie 18. (3 p)
Napisz procedurę, które przybliża wartość całki

$$\int\limits_{|x|<1} exp(-\tfrac{1+\prod_{i=1}^{D} x_i}{1-x\cdot x})d^D x$$

metodą Monte Carlo dla $D = 1...8$. Metoda ta polega na obliczeniu sumy

$$\tfrac{V}{N}\sum_{n=1}^{N} f(x_n)$$

gdzie x_n jest zmienną losową wybieraną z rozkładu jednorodnego, a V jest objętością zbioru, z którego dobieramy x_n. Do policzenia całki przyjąć $x_n \in [-1, 1]^D$, wtedy $V = 2^D$. Wykreśl wartość przybliżenia w raz z odchyleniem standardowym $\sigma_N = \sqrt{\frac{\sum_{i=1}^{N}(f(x_i)-<f>_N)^2}{N-1}}$ jako słupki błędów, względem wymiaru D i liczby punktów N.

Zadanie 19. (3 p)
Rwónania różniczkowe:
Zaimplementuj metody całkowania równań różniczkowych zwyczajnych: Eulera, punktu pośredniego, Rungego-Kutty 4 rzędu dla problemu rozpadu promieniotwórczego. Wartość początkową N oraz stałą rozpadu λ wczytaj ze strumienia wejsciowego.
Wyrysuj trajektorie przykładowych układów dla różnych wartości kroku całkowania i skomentuj wyniki.

Zadanie 20. (2 p)
Powtórz procedurę z oprzedniego zadania dla problemu oscylatora harmonicznego.
Wkazówka: Równanie oscylatora jest równaniem różniczkowym drugiego rzędu, przedstaw je jako układ równań rzędu pierwszego.

Zadanie 21. (3 p)
Rozwiąż układ równań różniczkowych zwanych Układem Lorentza:

$$\begin{aligned} x' &= \sigma(y-x) \\ y' &= x(\rho - z) - y \\ z' &= xy - \beta z \end{aligned}$$

gdzie $\sigma = 10$, $\beta = 8/3$, $\rho = 28$.

Rozwiązując układ równań zastosuj niejawną metodę Adamsa-Moultona, opisywaną równaniem:

$$y_{n+1} = y_n + \frac{h}{24}(9y'_{n+1} + 19y'_n - 5y'_{n-1} + y'_{n-2})$$

dla której predyktor wyznacz metodą Adamsa-Bashfortha:

$$y_{n+1} = y_n + \frac{h}{24}(55y'_n - 59y'_{n-1} + 37y'_{n-2} - 9y'_{n-3})$$

Otrzymane wyniki warto zwizualizować. W tym celu możesz skorzystać ze skryptu w języku Python:

```
import pylab
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt

mpl.rcParams['legend.fontsize'] = 10
x=np.fromfile("wynik.txt", sep=" ")
x = np.reshape(x, (-1,4))

print x[:, 0]
print x[:, 1]
print x[:, 2]

fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot(x[:, 0], x[:, 1], x[:, 2], label='Lorenz system')
ax.legend()

plt.show()
```

Aby skorzystać se skryptu wyniki obliczeń muszą zostać zapisane w pliku wynik.txt w kolejności $x(t)$ $y(t)$ $z(t)$ t.

Przykładowe punkty początkowe:

```
x_0 = 0
y_0 = 0.5
z_0 = 1
for t = 0
```

```
x_1 = 0.05
y_1 = 0.495
z_1 = 0.973333
for t = 0.01

x_2 = 0.0945
y_2 = 0.503563
z_2 = 0.947625
for t = 0.02

x_3 = 0.135406
y_3 = 0.524092
z_3 = 0.922831
for t = 0.03
```

Zadanie 21. (2 p)
Napisz prodecurę liczącą dyskretną transformatę Fouriera dla sygnału zespolonego o długości N.

Przetestuj procedurę dla funkcji $f(t) = e^{-2\pi ift}$ dla $f = \frac{1}{16}$ próbkowany w 16 punktach z krokiem 0,5.

Zadanie 22. (2 p)
Napisać program liczący transformatę Fouriera losowego wektora zespolonego bezpośrednio z definicji oraz metodą FFT, korzystając z biblioteki FFTW ($http://www.fftw.org$). Sprawdzić, że obie metody dają jednakowy wynik. Zmierzyć czas obliczeń obiema metodami dla wektorów długości 16 i 16384.

Zadanie 22. (3 p)
Korzystając z biblioteki FFTW napisać program generujący ciąg N liczb losowych a następnie wygładzający go przez policzenie transformaty Fouriera, wyzerowanie wszystkich współczynników poza pewną liczbą M współczynników o niskich częstościach i przeprowadzenie transformacji odwrotnej. Wyrysować ciąg przed i po wygładzeniu dla N = 1024 i M = 4, 16, 64, 256.