Rozdział 13

Przykładowe projekty zaliczeniowe

W tej części skryptu przedstawimy przykłady projektów na zaliczenia zajęć z laboratorium komputerowego z matematyki obliczeniowej. Projekty można potraktować jako trudniejsze zadania laboratoryjne.

Niektóre projekty są proste jeśli chodzi o stronę programistyczną. W przypadku większości takich projektów główną częścią jest przetestowanie danej metody. Inne projekty mogą polegać na zaimplementowaniu bardziej skomplikowanej metody z wykorzystaniem różnych funkcji octave'a.

Interpolacja wielomianowa. Algorytm różnic dzielonych.

1. Zaprogramuj w octave funkcje obliczającą wartości wielomianu zadanego w bazie Newtona dla danych węzłów zmodyfikowanym algorytmem Hornera dla danej tablicy punktów. Parametrami mają być:

   - x tablica punktów - wektor długości N z węzłami bazy Newtona,
   - a wektor długości $N+1$ ze współczynnikami wielomianu w bazie Newtona,
   - N - stopień wielomianu (można opcjonalnie obliczyć N z wektora współczynników).

   Funkcja zwraca y tablice wartości wielomianu w punktach x.

2. Zaprogramuj funkcję obliczającą różnice dzielone dla danych $N+1$ węzłów i wartości funkcji w tych węzłach algorytmem różnic dzielonych (jak z tabelki różnic dzielonych). Parametry funkcji to:

   - x wektor długości $N+1$ z różnymi węzłami
   - y wektor długości $N+1$ z wartościami funkcji f w węzłach.

   Funkcja powinna zwrócić wektor rd z różnicami dzielonymi $rd(k) = f[x_0, \ldots, x_k]$ dla $k = 0, \ldots, N$.

   Czy można to zrobić za pomocą tylko jednej pętli w octavie?

3. **Testy:** Interpolacja funkcji $f(x) = \sin(x)$ i $g(x) = \sin(4*x)$ na $[-\pi, 2\pi]$ dla węzłów równoodległych i Czebyszewa.

   - Porównanie algorytmu różnic dzielonych z algorytmem z funkcji octave'a **polyfit**.

     Znajdź wielomiany interpolacyjne w węzłach Czebyszewa i równoodległych stopnia N dla $N = 4, 8, 16, 32, 64$ w bazie Newtona za pomocą funkcji z projektu i za pomocą funkcji **polyfit** octave'a w bazie potęgowej. Oblicz dyskretną normę maksimum (na 1000 punktach) między wielomianem w bazie Newtona uzyskanym własną funkcją, a jej wielomianem interpolacyjnym uzyskanym **polyfit**. Czy różnice są pomijalne?
   - Błąd interpolacji

Oblicz dyskretną normę maksimum (na 1000 punktach) między funkcją, a jej wielomianem interpolacyjnym otrzymanym za pomocą funkcji z projektu dla $N = 4, 8, 16, 32, 64$. Narysuj wykresy funkcji i wykres jej wielomianu interpolacyjnego (na jednym rysunku).

Wartość wielomianu w bazie Newtona obliczamy algorytmem Hornera z pierwszego podpunktu.

Równania nieliniowe. Metoda Steffensena

Zaprogramuj funkcję octave'a z ***metodą Steffensena*** zdefiniowaną wzorem

$$x_{n+1} = x_n - \frac{f(x_n)^2}{f(x_n + f(x_n)) - f(x_n)}$$

- która ma znaleźć przybliżenie $f(x^*) = 0$.

Napisz funkcję:

function [x, it] = stef(FCN, x0)

w m-pliku `steff.m` z parametrami:

- FCN - 'wskaźnik' do funkcji (function handle) obliczającej wartość $f(x)$ dla danego argumentu x,
- $x0$ - liczba, przybliżenie startowe.

Funkcja powinna zwracać

- x obliczone przybliżenie pierwiastka,
- *it* - ilość iteracji.

Funkcja powinna się zatrzymać, drukując komunikat o braku zbieżności na ekranie komputera, gdy ilość iteracji przekroczy 100.

W przypadku przekroczenia ilości iteracji funkcja ma zwrócić komunikat o tym na ekran.

Przetestuj metodę Steffensena z wykorzystaniem funkcji steff() na przykładach równań rozpatrywanych w zadaniach w § 8.

W szczególności należy sprawdzić, czy metoda zbiega kwadratowo lokalnie, o ile $f'(x^*) \neq 0$ dla konkretnej funkcji $f(x) = x^3 - 27$.

Równania nieliniowe. Metoda Halleya

Zaprogramuj funkcję octave'a z ***metodą Halleya***, która jest metodą Newtona zastosowaną do funkcji

$$g(x) := \frac{f(x)}{\sqrt{|f'(x)|}},$$

która ma znaleźć przybliżenie $f(x^*) = 0$.

Wyprowadź wzór na kolejną iterację metody Halleya, w którym będziemy potrzebować odwołania tylko do wartości f, f' i f''.

Napisz funkcję z metodą Halleya [1]:

function [x, it]= halley (FCN,DFCN, D2FCN, x0)

w m-pliku `halley.m` z parametrami:

- FCN - 'wskaźnik' do funkcji (function handle) obliczającej wartość $f(x)$ dla danego argumentu x,
- DFCN - 'wskaźnik' do funkcji (function handle) obliczającej wartość pochodnej $f'(x)$ dla danego argumentu x,
- D2FCN - 'wskaźnik' do funkcji (function handle) obliczającej wartość drugiej pochodnej $f''(x)$ dla danego argumentu x,
- $x0$ - liczba, przybliżenie startowe.

Funkcja powinna zwracać

- x obliczone przybliżenie pierwiastka,
- it - ilość iteracji.

Jako warunek stopu należy przyjąć warunek spełnienie, któregoś z warunków $|f(x_n)| < 10^{-7}$ lub $|x_{n+1} - x_n| < 10^{-10}$.

Funkcja powinna się zatrzymać, drukując komunikat o braku zbieżności na ekranie komputera, gdy ilość iteracji przekroczy 100.

W przypadku przekroczenia ilości iteracji funkcja ma zwrócić komunikat o tym na ekran.

Przetestuj metodę Halleya z wykorzystaniem funkcji halley() na przykładach równań rozpatrywanych w zadaniach w § 8.

W szczególności należy sprawdzić, czy metoda zbiega kubicznie lokalnie, o ile $f'(x^*) \neq 0$ dla $f(x) = x^2 - 4$ dla $x^* = 2$.

[1] Tak, tego Halleya od komety Halleya.

Równania nieliniowe. Rozwikływanie funkcji.

Rozpatrzmy daną krzywą określoną równaniem $f(x, y) = 0$, gdzie funkcja $f(x, y)$ jest określona na prostokącie $[a, b] \times [c, d]$.

Chcemy znaleźć przybliżone wartości funkcji $y(x)$ zadanej w sposób uwikłany przez

$$f(x, y(x)) = 0$$

w punktach $x_k = a + k * h$ dla $h = (b - a)/N$, znając dobre przybliżenie $y(x_0) = y(a) = y0$. Tutaj N - to ustalona liczba naturalna.

- Napisz funkcję (w m-pliku) octave'a, która dla danych parametrów
  - FCN wskaźnika do funkcji dwóch argumentów $f(x, y)$,
  - ustalonych a, b - końców odcinka
  - $y0$ będącego przybliżeniem y_0
  - N ilości punktów, w których chcemy znaleźć przybliżenie $y_k \approx y(x_k)$ - ten parametr może być opcjonalny. Jeśli nie zostanie podany to powinien przyjmować domyślną wartość sto.

  Funkcja powinna zwrócić jako wektor $y_k = y(x_k)$ dla $k = 0, \ldots, N$.

  W każdym kroku trzeba rozwiązać, używając funkcji octave'a fsolve (), równanie nieliniowe:

  $$g(y_k) = f(x_k, y_k) = 0$$

  biorąc za startowe przybliżenie y_{k-1} (obliczone w poprzednim kroku dla $k - 1$).

  Jeśli się okaże, że fsolve() nie potrafi rozwikłać funkcji powinien na ekranie pojawić się komunikat o błędzie.

- Testować na dwóch przykładach:
  - Fragment elipsy:

    $$f(x, y) = 2 * x * x + y * y - 4.$$

    Interesuje nas tu $y(x)$ na $[-1.5, 1.5]$ na siatce 101 punktowej. Za $y0$ możemy wziąć $y0 = 1$
  - $g(x, y) = x^3 + y^3 - 4$ na $[0, 1] \times [0, 1]$.

Metoda Householdera. LZNK

Załóżmy, że w wyniku doświadczenia otrzymujemy m punktów (x_k, y_k), które powinny leżeć na jednej prostej, ale w wyniku błędów pomiaru leżą tylko blisko prostej. Chcemy wyznaczyć prostą $y = ax + b$, która leży najbliżej tych punktów w sensie najmniejszych kwadratów, tzn. takie a, b, że suma

$$\sum_{k=1}^{m} |ax_k + b - y_k|^2$$

jest minimalna. To zadanie możemy przedstawić jako odpowiednie regularne LZNK. Jeśli istnieją dwa punkty o różnych odciętych, to LZNK ma jednoznaczne rozwiązanie.

Celem projektu jest zaprogramowanie funkcji rozwiązującej problem znalezienia współczynników prostej $y = ax + b$ najlepiej przybliżającej dane punkty (x_k, y_k) $k = 1, \ldots, m$ za pomocą rozkładu QR macierzy przy pomocy metody Householdera. Tzn. szukamy (a, b) takich, że

$$\sum_{k=1}^{m} |ax_k + b - y_k|^2 = \min_{\hat{a},\hat{b}} \sum_{k=1}^{m} |\hat{a}x_k + \hat{b} - y_k|^2.$$

Czyli: w funkcji rozwiązujemy LZNK $A * [a; b] \approx \vec{y}$ z macierzą

$$A = [\vec{x}, \vec{1}]$$

dla wektorów

$$\vec{1} = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix},$$

$$\vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix}$$

i wektora prawej strony

$$\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}.$$

Jako parametry wejściowe funkcji traktujemy wektory $\vec{x}, \vec{y}$ długości m, funkcja powinna zwracać:

- wektor $[a; b]$ z rozwiązaniem tego LZNK,
- macierz A z LZNK
- macierz górnotrójkątną R wymiaru 2×2 z rozkładu QR macierzy A metodą Householdera, tzn. $A = Q * [R; 0]$,
- dwukolumnową macierz $B = [\vec{h}_1, \vec{h}_2]$ wymiaru $m \times 2$ - w której odpowiednie kolumny to wektory Householdera $\vec{h}_k$ $k = 1, 2$ dla macierzy Householdera H_k takich, że $H_1 * H_2 = Q$.

Jeśli kolumny macierzy A okażą się zależne liniowo - funkcja ma zwrócić odpowiedni komunikat na ekran.

Testy:

1. Przetestuj dla dwóch różnych punktów z różnymi x_k - czy znajdzie prostą przechodzącą przez te punkty.
2. Przetestuj dla punktów leżących na prostej: tzn. wygeneruj kilka różnych losowych punktów x_k i przyjmij $y_k = 2 * x_k - 10$. Następnie sprawdź, czy funkcja zwróci $a = 2, b = -10$.
3. Przetestuj dla punktów leżących blisko danej prostej: tzn. przyjmij np. $x_k = k/m$ dla $k = 1, \ldots, m$ dla różnych $m > 1$ z $y_k = x_k + 2 + \epsilon_k$ dla ϵ_k losowego z przedziału $[-10^{-3}, 10^{-3}]$

   (funkcja octave'a **rand**() zwraca losowe punkty z przedziału $[0, 1]$).
4. Sprawdź, czy rzeczywiście dla otrzymanych wektorów Householdera i macierzy R, A zachodzi

$$A = H_1 * H_2 * [R; 0],$$

   np. dla przykładów z poprzednich podpunktów. W tym celu można stworzyć macierze H_k i $[R; 0]$ i te trzy macierze wymnożyć.