Zadania i scenariusze zajęć z laboratorium komputerowego do wykładu z Matematyki Obliczeniowej

Leszek Marcinkowski

12 grudnia 2011

Streszczenie

W skrypcie przedstawimy zestawy zadań do odbywającego się co dwa tygodnie laboratorium komputerowego do semestralnego wykładu z Matematyki Obliczeniowej. Zakładamy, że zadania zostaną rozwiązane przy wykorzystaniu pakietu obliczeń numerycznych octave albo pakietu matlab.

Spis treści

Rozdział 1

Wstęp

Celem zadań komputerowych przeprowadzanych w laboratorium komputerowym jest przetestowanie numerycznych metod omawianych w czasie wykładu i ćwiczeń z Matematyki Obliczeniowej.

Zadania komputerowe przedstawione w tym zbiorze polegają na implementacji metod numerycznego rozwiązywania zadań z wykorzystaniem pakietu octave, czyli środowiska obliczeń numerycznych-naukowych i przetestowaniu funkcji octave'a, które są w stanie rozwiązać zadania omawiane w trakcie kursu Matematyki Obliczeniowej.

Część funkcji octave'a wywołuje odpowiednią bibliotekę numeryczną, w której jest zaimplementowana odnośna - zazwyczaj zaawansowana - metoda. Ale część funkcji octave'a jest zaimplementowana wprost w samym octave'ie. Często nie wiemy jakiej metody numerycznej używa octave. Szczególnie, że kolejne wersje tego pakietu używają innych bibliotek, mimo że nazwa funkcji - np. rozwiązującej równanie nieliniowe - jest wciąż taka sama.

Pakiet octave jest zarówno środowiskiem obliczeń numerycznych, jak i językiem programowania. Umożliwia on proste rozwiązywanie podstawowych zadań numerycznych jak: numeryczne obliczenie całki, rozwiązanie zadań liniowych lub nieliniowych, równań różniczkowych zwyczajnych itp. Można go używać z linii komend, pisać własne skrypty, czy m-pliki (czyli pliki z implementacjami własnych funkcji octave'a).

Pakiet octave jest programem ogólnodostępnym jako wolne oprogramowanie. Rozprowadzany jest na zasadach licencji GNU GPL. Warto dodać, że pakiet octave jest odpowiednikiem środowiska matlab, które jest programem komercyjnym, szeroko stosowanym do obliczeń numerycznych.

Zadania z tego zbioru są sformułowane tak, że bez kłopotów mogą być wykonane zarówno w octave, jak i w środowisku matlaba.

Zaletą octave'a jest to, że jest on dostępny bez opłat. Octave jest dystrybuowany w wersjach binarnych zarówno pod różne dystrybucje Linuxa, jak i

w wersji binarnej pod system Windows.

Poniżej załączamy link do stron octave'a:

1. Główna strona octave'a
2. Rozbudowany podręcznik do octave'a - w języku angielskim dostępny on-line.
3. Strona z linkami do plików z octavem
4. Strona octave-forge'a - czyli rozszerzeń octave'a

Podstawowym podręcznikiem do kursu z matematyki obliczeniowej jest książka [11]. Innymi wartymi polecenia podręcznikami w języku polskim są np. [8], [3], [17] i [1], [16]. W języku angielskim opublikowano wiele podręczników do matematyki obliczeniowej, inaczej nazywanej też metodami numerycznymi, czy analizą numeryczną. Niektóre z nich zawierają zadania komputerowe. Warto polecić książki: [15], [14]. Obie zawierają więcej materiału niż standardowy kurs matematyki obliczeniowej na wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego. Innymi wartymi polecenia podręcznikami i monografiami dotyczącymi metod numerycznych są: [5], [6], [2], [12], [13], [9], [18], [10], [7] i inne. Niektóre z tych podręczników obejmują tylko część materiału z wykładu matematyki obliczeniowej lub wykraczają poza ten materiał.

Jeśli chodzi o pakiet octave - polecamy przeczytać manual, tzn. [4] dostępny w dziale dokumentacji na stronach octave'a on-line pod adresem: `http://www.gnu.org/software/octave/`.

Zadania w tym zbiorze są podzielone na rozdziały. Standardowe zajęcia z laboratorium komputerowego do Matematyki Obliczeniowej polegają zazwyczaj na rozwiązaniu kilku zadań z zadanego tematu. Pozostałe zadania mogą zostać ewentualnie zadane do rozwiązania samodzielnego.

Zajęcia z laboratorium do Matematyki Obliczeniowej na wydziale Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego odbywają się co dwa tygodnie, więc ilość rozdziałów w tym skrypcie przewyższa standardową liczbę zajęć. Prowadzący zajęcia może wybrać, które rozdziały omówić na zajęciach. Zadania w poszczególnych rozdziałach są tak skomponowane, że standardowy scenariusz zajęć dotyczących tematu danego rozdziału powinien obejmować kilka pierwszych zadań. Prowadzący zajęcia może dokonać samodzielnego wyboru zadań, pomijając niektóre z nich.

Scenariusze i zadania komputerowe

Rozdział 2

Wstępne zapoznanie się z octavem

W przypadku przeznaczenia dwóch laboratoriów scenariusz każdych zajęć polega na rozwiązaniu możliwie dużej ilości kolejnych zadań z tego rozdziału. W przypadku jednych zajęć należy dokonać wyboru rozwiązując tylko kilka najważniejszych pozycji z listy zadań.

Zadania w tym rozdziale ilustrują podstawowe operacje, struktury i własności octave'a. Przedstawimy w zadaniach octave jako kalkulator naukowy. Omówimy też operator dwukropek służący np. tworzeniu indeksów. Przetestujemy, jak tworzyć macierze, wektory; jak zapisywać zmienne do plików (czytać z plików) w formatach: tekstowym i binarnym. Sprawdzimy tworzenie macierzy z podmacierzy, wycinanie podmacierzy i inne podstawowe operacje na macierzach - mnożenie, dodawanie, transponowanie, funkcje matematyczne od macierzy, normy wektorów/macierzy.

Zadania obejmują również tworzenie wykresów funkcji matematycznych przy pomocy funkcji octave'a **plot**(). Sprawdzimy też tworzenie i używanie skryptów i funkcji (m-pliki) w octavie, oraz podstawowe instrukcje warunkowe i pętle:

- **if else endif**;
- **switch case** endswitch
- **while**( ) **do endwhile**;
- **do** .. **until**( );
- **for** .. **endfor**.

Zadania obejmą też wskaźniki do funkcji (function handle) i operator @ - zwracający wskaźnik do funkcji.

Zadanie 1 **Octave jako kalkulator**. Otwórz sesję octave'a. Zapoznaj się z pomocą do funkcji **sqrt**() oraz **sin**(). Policz w octave'ie, ile wynosi pierwiastek z 5 oraz policz wartość funkcji sin na tym pierwiastku.

Zadanie 2 **Operacje macierzowe. Operator dwukropek.**

- Utwórz wektor z liczbami od 1 do 20 oraz wektor ze wszystkimi liczbami parzystymi od -6 do 4.
- Utwórz dowolne macierze 3x4 A i 3x5 B, a następnie macierz 3x8 C, której pierwsze 3 kolumny to A, a kolejne to B.
- Z macierzy C 'wytnij' podmacierz D składającą się z 1 głównego minora tzn. 3x3 od C(1,1) do C(3,3).
- Zamień kolejność kolumn D.
- Zamień kolejność wierszy D.
- Wytnij dolnotrójkątną, a potem górnotrójkątną część macierzy D
- Wstaw D z powrotem do C jako główny minor.
- Policz $\sin(D) = (\sin(D_{ij})$ od D.
- Zapisz D do pliku (binarnego i ASCII) - zamień element D(1,1) na -100 i wczytaj nową macierz do octave'a.

Zadanie 3 **Funkcje matematyczne od macierzy. Normy macierzy i wektorów**

Policz dyskretną normę maksimum od $(\sin(x))^2$ na $[0, 1]$ (wektorowo-czyli bez użycia pętli).

Zadanie 4 **Wykresy funkcji matematycznych.**

- Narysuj wykres funkcji $\sin(x)$ na odcinku $[0, 4]$.
- Narysuj wykres funkcji $\sin(x)$ na odcinku $[0, 4]$ - wykres powinien być podpisany, narysowany za pomocą gwiazdek w kolorze czerwonym.
- Narysuj w jednym oknie podpisane wykresy funkcji $\sin(x)$ i $\log(x)$ na odcinku $[1, 6]$ podpisane odpowiednio.

Zadanie 5 Znajdź przybliżone maksimum i minimum funkcji

$$f(x) = x * (3 + 2 * \cos(x))$$

na odcinku $[-1, 5]$ bez użycia pętli, oraz przybliżenia punktów ekstremalnych.

Powtórz to zadanie dla jakiegoś wielomianu stopnia dwa i trzy np. $x^3 + x^2 - x - 4$.

Zadanie 6 **Funkcje w octave'ie, m-pliki, czyli tzw. pliki funkcyjne.**

Utwórz funkcję w m-pliku obliczającą dla zadanego x wartość funkcji

$$f(x) = (\sin(x))^2.$$

Zadanie 7 **Zmienne globalne**

Utwórz m-plik z funkcją octave'a obliczającą wartość funkcji matematycznej z parametrem: $\sin(a * x)$ - parametr przekaż jako zmienną globalną.

Zadanie 8 **Implementacja wektorowa funkcji w octave'ie.**

Utwórz m-plik z funkcją obliczający wartość funkcji $f(x) = 1 + (\cos(x))^2$ dla argumentu będącego macierzą, tzn. jeśli $X = (x_{i,j})_{i,j}$ macierz wymiaru $M \times N$, to funkcja powinna zwrócić Y macierz wymiaru $M \times N$ taką, że

$$Y = (y_{i,j})_{i,j}, \qquad y_{i,j} = f(x_{i,j}).$$

Narysuj wykres f na odcinku $[-1, 5]$ z wykorzystaniem tylko jednego wywołania tak zaimplementowanej funkcji octave'a.

Zadanie 9 **Funkcje anonimowe.**

Utwórz funkcję uchwyt do funkcji anonimowej, która dla danego argumentu x zwraca wartość równą $(\sin(x))^2$.

Zadanie 10 **Pętle, instrukcje warunkowe, instrukcja printf().**

- Zapoznaj się z pomocą octave'a do pętli **while**, pętli **for**, instrukcji warunkowej **if** oraz funkcji drukującej napisy na ekranie **printf**().
- Przy pomocy pętli

  for (k = . . .)

  endfor

while(warunek stopu)

endwhile

oblicz sumę

$$S_N = 1 + ... + N$$

dla $N = 100$.

- Użyj instrukcji warunkowej

  if()

  endif

  by sprawdzić, czy otrzymane S_N zostało obliczone poprawnie, tzn. czy otrzymaliśmy $0.5 * N * (N + 1)$

- wyprowadź na ekran komunikat używając

  printf()

Rozdział 3

Obliczenia w arytmetyce zmiennopozycyjnej

Zadania z tego rozdziału powinny wykazać pewne charakterystyczne własności arytmetyki zmiennopozycyjnej.

W octavie wykorzystywane są domyślnie liczby zmiennopozycyjne o podwójnej precyzji, jakkolwiek w najnowszych wersjach octave'a można również sztucznie wymusić używanie zmiennych w precyzji pojedynczej przy pomocy funkcji single(a) zwracającej zmienną pojedynczej precyzji z tą samą wartością.

Jedną z ważnych własności arytmetyki zmiennopozycyjnej, która wynika z jej konstrukcji, jest to, że odejmowanie dwóch wartości o tym samym znaku o małej różnicy może skutkować dużą utratą dokładności względnej.

Zadanie 1 **Funkcje single(a) i eps.** Wywołaj pomoc do tych funkcji w octave'ie. Sprawdź, czy prawdą jest, że $1 + eps$ obliczone w arytmetyce podwójnej precyzji w octave'ie jest większe od jeden. Przyjmij, że $a = single(eps)$ i sprawdź, czy ponownie $1 + a$ jest większe od jeden.

Zadanie 2 **Epsilon maszynowy w arytmetyce podwójnej precyzji**

Wyznacz samodzielnie epsilon maszynowy - czyli najmniejszą liczbę w arytmetyce zmiennopozycyjnej taką, że po dodaniu jej do jeden otrzymujemy liczbę większą od jeden. Będziemy szukali liczby postaci 2^{-t} dla t - ilości bitów mantysy. Porównaj z **eps** komendą octave'a. Zadanie można też wykonać w C/C++. Czy otrzymane wyniki są takie same jak te otrzymane w octavie (dla liczb typu double)?

Zadanie 3 **Epsilon maszynowy w arytmetyce pojedynczej precyzji**
Powtórz poprzednie zadanie dla arytmetyki pojedynczej precyzji. Funkcja octave'a single (x) tworzy zmienne takiego typu. Wykorzystując tę funkcję ponownie wyznacz epsilon maszynowy jako liczbę postaci 2^{-t}, ale dla liczb w pojedynczej precyzji.

Zadanie 4 Narysuj wykres funkcji $f(x) = (x + a) - a$ na $[0, 1]$ dla różnych wartości $a = 10^k$ dla $k = 1, 2, .., 20$. Tutaj ważne jest aby obliczać wartość $f(x)$ dokładnie ze wzorów: $b = (x - a)$, $f(x) = b - a$, choć matematycznie $f(x) = x$.

Zadanie 5 Policz

$$f(x) = x - \sqrt{1 + x * x}$$

algorytmem wprost wynikającym z tego wzoru, a następnie z wykorzystaniem równoważnego wzoru

$$f(x) = \frac{-1}{x + \sqrt{1 + x * x}}$$

tzn. proszę zastosować:

Algorytm 1

$$a = \sqrt{1 + x^2} \quad w_1 = x - a$$

oraz

Algorytm 2

$$a = \sqrt{1 + x^2} \quad w_2 = \frac{-1}{x + a}$$

dla $x = 10^k$ i $k = 4, \ldots, 10$. Czy widać różnicę w wyniku?

Powtórz zadanie w arytmetyce pojedynczej precyzji, tzn. z wykorzystaniem funkcji octave'a single (x).

Zadanie 6 **Wykres wielomianu na dwa sposoby**

Oblicz wartości wielomianu

$$(x - 2)^4 = x^4 - ..... + 16$$

na siatce równomiernej 1000 punktowej na $[2 - a, 2 + a]$ dla $a = 10^{-3}$ za pomocą dwóch algorytmów:

Algorytm 1

$$a = (x - 2), \quad f_1(x) = a^4;$$

Algorytm 2

$$f_2(x) = x * x * x * x - \ldots + 16.$$

Matematycznie $f_1 \equiv f_2$, ale wyniki obliczone w arytmetyce zmiennopozycyjnej mogą się różnić.

Narysuj wykresy obu funkcji i policz błąd $\|f_1(x) - f_2(x)\|_\infty$, czyli $\max_k |f_1(x(k) - f_2(x(k))|$. Tu $x(k) = 2 - a + k * h$ dla $h = a/500$. Wektor x można utworzyć w octavie przy pomocy funkcji octave'a **linspace**().

Zadanie 7 Powtórz poprzednie zadanie dla arytmetyki pojedynczej precyzji, tzn. powtórz obliczenia wielomianu $(x-2)^4 = x^4 - ..... + 16$ na odcinku $[2-a, 2+a]$ dla $a = 10^{-3}, 10^{-2}, 10^{-4}$ dla zmiennych w pojedynczej precyzji uzyskanych za pomocą funkcji octave'a: single (x). Narysuj wykresy wielomianu obliczanego obydwoma algorytmami.

Zadanie 8 Przybliżenie $\exp(x)$ z rozwinięcia w szereg $\sum_{k=0}^{\infty} x^k/k!$. Za odpowiednią aproksymacje $\exp(x)$ bierzemy najpierw sto pierwszych elementów szeregu czyli przybliżamy $\exp(x)$ przez

$$F_N(x) = \sum_{k=0}^{N} \frac{x^k}{k!},$$

dla $N = 100$, a potem przybliżamy przez tysiąc elementów szeregu, czyli ustalamy $N = 1000$.

Sprawdź błąd względny $|F_N(x) - \exp(x)|/|\exp(x)|$ dla x od -100 do 100 (np. dla liczb różniących się o dziesięć, czyli $-100 + k * 10$ dla $k = 0, \ldots, 20$) dla obu wartości N.

Czy błędy dla liczb ujemnych i dodatnich są tego samego rzędu?

Jak zmodyfikować powyższą metodę przybliżonego obliczania funkcji eksponencjalnej $\exp(x)$ dla $x << 0$ tak aby błąd względny był na tym samym poziomie co dla $x > 0$?

Zadanie 9 Policz całki $I_n = \int_0^1 x^n/(5+x)dx$ $n = 0, .., 20$ dwoma algorytmami ze wzoru

$$I_n + 5 * I_{n-1} = 1/n$$

Pierwszy algorytm przyjmuje $I_0 = \log(6/5)$ i oblicza z powyższego wzoru kolejne

$$I_n = 1/n - 5 * I_{n-1}$$

dla $n = 1, 2, 3, \ldots$.

Drugi algorytm wykorzystuje fakt, że

$$\frac{1}{(n+1)6} \leq I_n \leq \frac{1}{(n+1)5} \tag{3.1}$$

W tym algorytmie przyjmujemy za I_{30} jakąkolwiek wartość z tego przedziału np. $I_{30} = 1/180$ i iterujemy w tył, tzn. dla $n = 30, 29, \ldots, 20, \ldots, 0$ obliczamy

$$I_{n-1} = \frac{1}{5}(1/n - I_n).$$

Porównaj wyniki obu algorytmów dla $0 \leq n \leq 20$ oraz sprawdź czy wyniki otrzymane w octave'ie spełniają oszacowanie (3.1) dla $n = 1, \ldots, 20$.

Dlaczego jeden z algorytmów działa zdecydowanie lepiej w arytmetyce zmiennopozycyjnej?

Jako dodatkowe zadanie teoretyczne pozostawimy uzasadnienie wzoru rekurencyjnego i oszacowania (3.1) wykorzystywanych w algorytmach.

Zadanie 10 Zastosuj algorytm bisekcji (algorytm połowienia odcinka) dla funkcji $(x-2)^3$ liczonej z wzoru na rozwinięcie dwumianu $x^3 - ... - 8$ startując z $a = 2 - 10^{-3}$ a $b = 2 + 10^{-3}$. Jako warunek zakończenia działania algorytmu przyjmujemy, że błąd jest mniejszy od 10^{-20} (za przybliżenie rozwiązania przyjmujemy środek danego odcinka w metodzie bisekcji, czyli warunkiem zakończenia iteracji jest to, że długości odcinka, w którym jest rozwiązanie powinna być mniejsza od $2 * 10^{-20}$).

Czy algorytm zwraca przybliżenie liczby dwa jedynego pierwiastka tego wielomianu?

Narysuj wykresy tej funkcji obliczone z obu wzorów. tzn. $(x-2)^3$ i $x^3 - ... - 8$ na odcinkach $2 + [-h, 2 * h]$ dla $h = 10^{-3}, 10^{-4}, 10^{-5}$. Czy z wykresów wynika, że ten wielomian ma tylko jedno miejsce zerowe w otoczeniu dwa?

Rozdział 4

Interpolacja Lagrange'a, bazy wielomianów

W tym rozdziale zajmiemy się interpolacją wielomianową. Zadanie interpolacji wielomianowej polega na znalezieniu wielomianu stopnia nie większego od n, spełniającego $n+1$ warunków interpolacyjnych.

W octave'ie istnieje funkcja znajdująca współczynniki wielomianu interpolacyjnego Lagrange'a. Jest to funkcja **polyfit**(), czyli funkcja obliczająca współczynniki wielomianu interpolacyjnego w bazie potęgowej dla zadanych wartości i węzłów. Z kolei funkcja **polyval**() jest funkcją obliczającą wartość wielomianu zadanego poprzez współczynniki w bazie potęgowej w jednym lub równocześnie wielu punktach - czyli wektorowo. Co ważne, obie funkcje są ze sobą zgodne. Trzeba uważać na kolejność współczynników; octave numeruje współczynniki w bazie potęgowej

$$(x^j)_{j=0}^n$$

przestrzeni wielomianów stopnia nie większego od n w odwrotnej kolejności tzn.

$$(x^n, x^{n-1}, \ldots, 1)$$

czyli np. wektor współczynników $(3,2,1)$ odpowiada wielomianowi $3x^2+2x+1$. Oczywiście stopnień wielomianu, a dokładniej: dla jakiego n rozpatrujemy bazę potęgową dla tego wektora, jest długością wektora współczynników pomniejszoną o jeden.

Zadanie 1 Zapoznaj się z pomocą octave'a do funkcji octave'a **polyval**(). Oblicz wartość wielomianu $x^{50}-1$ dla $x=-1,0,1$.

Zadanie 2 Korzystając z funkcji **polyval**() narysuj wykres wielomianu x^3+x-2 bez wykorzystania pętli - czyli wektorowo.

Zadanie 3 Test funkcji **polyfit**().

Zapoznaj się z pomocą octave'a do funkcji octave'a **polyfit**().

Wykorzystując funkcję **polyfit**() znajdź wielomian interpolacyjny L_nF dla funkcji $F(x) = \sin(x)$ dla węzłów $-1, 0, 1$. Policz wartości różnicy $F - L_nF$ w węzłach oraz korzystając z funkcji **plot**() narysuj wykresy F i L_nF na jednym rysunku.

Wykonaj powtórnie to zadanie ale dla węzłów $-1, 0, 1, 10$.

Oblicz wartości wielomianu bez użycia pętli z wykorzystaniem funkcji **polyval**().

Zadanie 4 Interpolacja Lagrange'a - zbieżność ciągu wielomianów interpolacyjnych dla węzłów równoodległych i węzłów Czebyszewa:

- Wykorzystując funkcję **polyfit**() znajdź wielomiany interpolacyjne L_Nf dla funkcji $f = \sin()$ dla N węzłów równoodległych na $[0, 2 * \pi]$ dla $N = 4, 8, 16, 32, 64$.
- Oblicz dyskretną normę maksimum różnicy $f - L_Nf$ na siatce tysiąca równoodległych punktów na tym odcinku, tzn. $e_N = \max|f(x_k) - L_Nf(x_k)|$, gdzie x_k to punkty siatki. Policz stosunek e_N/e_{2N} dla $N < 64$.

  Czy błędy maleją do zera? Jak zachowuje się e_N/e_{2N}?

  Siatkę tysiąca równoodległych punktów na odcinku $[a, b]$ najprościej utworzyć z wykorzystaniem funkcji octave'a: **linspace**(a,b,1000).
- Narysuj na ekranie wykresy $\sin(x)$ i tych wielomianów dla różnych N - używając funkcji **polyval**() i **plot**().

Zadanie 5 Powtórz zadanie 4 dla tej samej funkcji i tego samego odcinka dla węzłów Czebyszewa. Węzły Czebyszewa to pierwiastki wielomianu:

$$T_{n+1}(t) = \cos((n+1)\arccos(t))$$

na $[-1, 1]$ odpowiednio przesunięte i przeskalowane.

Przetestuj, czy błędy e_N dla węzłów Czebyszewa są mniejsze niż dla węzłów równoodległych.

Zadanie 6 Napisz funkcję znajdującą współczynniki w bazie potęgowej wielomianu interpolacyjnego zadanego stopnia dla węzłów równoodległych oraz węzłów Czebyszewa dla danej funkcji, odcinka $[a, b]$, tzn. napisz funkcję octave'a (w m-pliku):

function [LN,eN]=LagrInterp(FCN,a,b,N,**type**=0)

która dla parametrów:

- zadanego wskaźnika funkcyjnego FCN (ang. *function handle*) do funkcji jednego argumentu: **function** y=f(x),
- a, b - końców odcinka $[a, b]$,
- N - stopnia wielomianu interpolacyjnego
- **type** - typu węzłów 0 - równoodległych, 1 - Czebyszewa

zwróci wektor LN - współczynniki $L_N f$ wielomianu interpolującego funkcję w tych węzłach oraz eN przybliżenie dyskretnej normy maksimum różnicy $L_N f - f$ na tym odcinku.

Przybliżenie normy maksimum liczymy na dyskretnej siatce zawierającej tysiąc punktów.

Zadanie 7 Powtórz zadanie 4 dla obu typów węzłów, tzn. powtórz znajdowanie wielomianów interpolacyjnych na węzłach równoodległych i węzłach Czebyszewa dla funkcji

$$f(x) = \log(1 + x)$$

na odcinkach

- $[0, 1]$,
- $[0, 10]$.

Czy dla tej funkcji i obu odcinków błędy w normach maksimum maleją wraz ze wzrostem N? Porównaj wyniki otrzymane w tym zadaniu w obliczeniach z oszacowaniami teoretycznymi błędu interpolacji Lagrange'a.

Zadanie 8 Interpolacja Lagrange'a - przykład Rungego. Powtórz zadanie 4 dla obu typów węzłów, tzn. znajdowanie wielomianów interpolacyjnych na węzłach równoodległych i węzłach Czebyszewa, ale dla funkcji:

$$f(x) = 1/(1 + x * x)$$

na $[-5, 5]$.

Czy obliczone wyniki wskazują na to, że obliczone ciągi wielomianów interpolacyjnych zbiegają do f jednostajnie dla obu typów węzłów?

Zadanie 9 Napisz funkcję octave'a obliczającą wartość wielomianu zadanego w bazie potęgowej tzn.

$$w(x) = \sum_{k=0}^{n} a_k x^k$$

algorytmem Hornera. Parametrami funkcji będą

- wektor współczynników a
- macierz wartości x.
- N - stopnień wielomianu (to może być parametr opcjonalny, domyślnie przyjmujący wartość równą długości wektora a minus jeden)

Funkcja ma zwrócić wartości wielomianu dla wartości w x.

Przetestuj funkcję dla wielomianów $1 + x^2$ oraz $1 - 2x + x^2$ dla węzłów równoodległych na $[-1, 2]$, tzn. policz wartości wielomianów dla kilku wartości oraz narysuj wykresy tych wielomianów z wykorzystaniem tej funkcji.

Zadanie 10 Napisz funkcję octave'a znajdującą dla danego wielomianu stopnia n:

$$w(x) = \sum_{k=0}^{n} a_k x^k$$

oraz danej liczby q współczynniki $(b_k)_{k=0}^{n}$ wielomianu

$$p(x) = \sum_{k=0}^{n-1} b_k x^k$$

oraz wartość r takie, że

$$w(x) = (x - q) * p(x) + r,$$

obliczone z wykorzystaniem algorytmu Hornera.

Zadanie 11 Napisz funkcję octave'a znajdującą dla danego wielomianu stopnia n:

$$w(x) = \sum_{k=0}^{n} a_k x^k$$

oraz liczby q wartość $w(q)$ i pochodnej $w'(q)$, obliczone z wykorzystaniem algorytmu Hornera.

Zadanie 12 Algorytm Hornera w bazie Newtona. Różnice dzielone.

Zaprogramuj w octavie funkcję ze zmodyfikowanym algorytmem Hornera zwracającą wartość wielomianu zadanego w bazie Newtona dla danych węzłów. Parametrami będą

- x punkt, w którym obliczamy wielomian (ewentualnie tablica punktów, ale wtedy funkcja też musi zwrócić wektor z wartościami wielomianu w tych punktach),
- N - stopnień wielomianu,
- wektor długości $N+1$ ze współczynnikami wielomianu w bazie Newtona.

Przetestuj na kilku prostych przykładach: dla węzłów $-1, 0, 1$ i wielomian $w(x) = x^2$, który w bazie Newtona związanej z tymi węzłami ma następującą postać: $x^2 = (x+1)x - (x+1) + 1$.

Zadanie 13 Napisz funkcję octave'a, która dla danego wielomianu $w(x)$, którego współczynniki w bazie potęgowej znamy, oblicza współczynniki tego wielomianu w bazie Newtona dla zadanych węzłów podanych w wektorze y. Tzn. parametrami funkcji będą:

- wektor $a = (a_k)_k$ taki, że
$$w(x) = \sum_{k=0}^{n} a_k x^k$$
- $y = (y_k)_k$ wektor współczynników bazy Newtona.

Funkcja powinna zwrócić wektor współczynników b_k takich, że
$$w(x) = \sum_{k=0}^{n} b_k w_k,$$
gdzie
$$w_k(x) = \Pi_{j=0}^{k-1}(x - y_j).$$

Zadanie 14 Napisz funkcję, która dla danego wielomianu $w(x)$, którego współczynniki w bazie Newtona $(w_k)_{k=0}^n$ znamy, oblicza współczynniki tego wielomianu w bazie potęgowej $(1, x, x^2, \ldots, x^n)$. Tzn. parametrami funkcji jest wektor współczynników $b = (b_k)_k$ takich, że
$$w(x) = \sum_{k=0}^{n} b_k w_k$$

i wektor $y = (y_k)_k$ węzłów bazy Newtona $(w_k)_{k=0}^n$ dla

$$w_k(x) = \Pi_{j=0}^{k-1}(x - y_j).$$

Funkcja ma zwrócić wektor współczynników $a = (a_k)_k$ takich, że

$$w(x) = \sum_{k=0}^{n} a_k x^k.$$

Zadanie 15 Sprawdź eksperymentalnie ile wynosi dla różnych wartości $N = 4, 8, 16, 32, 64, \ldots$ przybliżenie:

$$A_{r,N} = \sum_{k=0}^{N} \|l_k\|_{\infty,[-1,1]}$$

dla $\{l_k\}_{k=0}^N$ bazy Lagrange'a dla węzłów równoodległych na $[-1, 1]$, tzn. dla $x_k = -1 + k * h$ dla $h = 2/N$.

Normę maksimum funkcji $\|f\|_{\infty,[a,b]}$ liczymy w sposób przybliżony obliczając dyskretną normę maksimum na $\max\{1000, 100 * N\}$ równoodległych punktach z odcinka $[a, b]$.

Zadanie 16 Sprawdź eksperymentalnie ile wynosi dla różnych N np.

$$N = 4, 8, 16, 32, 64, \ldots$$

przybliżenie:

$$A_{c,N} = \sum_{k=0}^{N} \|l_k\|_{\infty,[-1,1]}$$

dla $\{l_k\}_{k=0}^N$ bazy Lagrange'a dla węzłów Czebyszewa na $[-1, 1]$, tzn. dla x_k zer wielomianu $T_{N+1}(x) = \cos((N + 1)\arccos(x))$.

Normę maksimum funkcji $\|f\|_{\infty,[a,b]}$ liczymy w sposób przybliżony obliczając dyskretną normę maksimum na $\max\{1000, 100 * N\}$ równoodległych punktach z odcinka $[a, b]$.

Zadanie 17 Powtórz dwa poprzednie zadania dla węzłów równoodległych i węzłów Czebyszewa na odcinku $[0, 10]$ i dla odpowiedniej normy maksimum na tym odcinku.

Zadanie 18 Policz iloraz

$$\frac{\|L_N f\|_{\infty,[-5,5]}}{\sum_{k=0}^{N} \|l_k\|_{\infty,[-5,5]}}$$

dla $f = 1/(1+x^2)$ i $L_N f$ wielomianu interpolującego f w węzłach równoodległych na $[-5, 5]$ dla $N = 10, 20, 40, 80$. Tutaj $\{l_k\}_{k=0}^N$ baza Lagrange'a dla tych węzłów.

Normę maksimum funkcji $\|f\|_{\infty,[a,b]}$ liczymy w sposób przybliżony obliczając dyskretną normę maksimum na $\max\{1000, 100 * N\}$ równoodległych punktach z odcinka $[a, b]$.

Zadanie 19 Powtórz poprzednie zadanie dla tych samych funkcji i odcinka dla węzłów Czebyszewa zamiast węzłów równoodległych.

Rozdział 5

Splajny kubiczne i liniowe. Interpolacja splajnowa

W tym rozdziale zajmiemy się interpolacją splajnową, czyli interpolowaniem danej funkcji za pomocą splajnów - inaczej funkcji giętych.

Skupimy się na splajnach kubicznych, czyli funkcjach, które są klasy C^2 na odcinku $[a, b]$ i dla danego podziału tego odcinka:

$$a = x_0 < x_1 \ldots < x_N = b$$

na pododcinki. Te funkcje obcięte do każdego pododcinka $[x_i, x_{i+1}]$ są wielomianami kubicznymi.

Zadanie interpolacji splajnami kubicznymi polega na znalezieniu spaljnu kubicznego s spełniającego:

$$\begin{aligned} s(x_0) &= y_0 \\ s(x_1) &= y_1 \\ &\vdots \\ s(x_N) &= y_N \end{aligned}$$

dla zadanych wartości y_k. Okazuje się, że tak postawione zadanie nie jest jednoznaczne; trzeba dodać dwa dodatkowe warunki na s. Zazwyczaj są to odpowiednie warunki brzegowe, tzn. związane z wartościami s, pierwszych lub drugich pochodnych s w końcach odcinka.

Zadanie 1 Funkcje octave'a **spline**() and ppval.

Zapoznaj się z pomocą do tych funkcji (**help spline** i **help** ppval).

Wykorzystując te funkcje narysuj wykres splajnu kubicznego s_1 na podziale równomiernym odcinka $[-3, 3]$ z węzłami $\{x_k = k\}$

dla $k = -3, -2, \ldots, 3$ przyjmującego wartości $s_1(x_k) = (-1)^k$ w tych węzłach.

Następnie znajdź współczynniki splajnu kubicznego s_2 na tym samym podziale odcinka i przyjmującego te same wartości w węzłach co s_1, ale który dodatkowo przyjmuje wartości pochodnych w końcowych węzłach równe zero, tzn. wywołaj funkcje **spline**() podając dwie wartości więcej.

Następnie narysuj wykresy splajnów s_1 i s_2 na tym samym rysunku.

Czy otrzymaliśmy te same splajny?

Policz przybliżoną normę maksimum różnicy $s_1 - s_2$ na odcinku $[-3, 3]$.

Zadanie 2 Splajn kubiczny bazowy.

Dla danych węzłów równoodległych $\{k\}_{k=-5,-4,\ldots,5}$ na $[-5, 5]$ narysuj wykres splajnu kubicznego typu not-a-knot (czyli splajnu, którego współczynniki zwróci funkcja **spline**() przy najprostszym wywołaniu przez podanie wektora węzłów i wektora wartości w tych węzłach, por. **help spline**) takiego, że $s(0) = 1$ i $s(k) = 0$ dla węzłów $k \neq 0$.

Określ na podstawie wykresu nośnik tego splajnu.

Zadanie 3 Splajn kubiczny o minimalnym nośniku.

Dla danych węzłów równoodległych $\{k\}_{k=-5,-4,\ldots,5}$ na $[-5, 5]$ narysuj wykres splajnu kubicznego takiego, że $s(-1) = s(1) = 1, s(0) = 4$ i $s(k) = 0$ dla węzłów $k \notin \{-1, 0, 1\}$ oraz ma pochodne równe zero w węzłach skrajnych, tzn. : -5 i 5. Czy poza $[-2, 2]$ ten splajn jest równy zero?

Policz przybliżone normy maksimum na $[-5, -2]$ i $[2, 5]$ dla tego splajnu.

Zadanie 4 Testowanie eksperymentalne rzędu zbieżności splajnu interpolacyjnego kubicznego z hermitowskimi warunkami brzegowymi.

Korzystając z funkcji octave'a **spline**() znajdź współczynniki interpolacyjnego splajnu kubicznego hermitowskiego S_N na N węzłach równoodległych dla funkcji $f(x) = \sin(x)$ na odcinku $[-\pi, 2*\pi]$ dla $N = 2^k N_0$ dla $N_0 = 5$ i $k = 1, 2, 3, 4, 5$.

Następnie

- narysuj wykresy funkcji $f(x)$ i splajnów S_N dla różnych N.
- oblicz dyskretną normę maksimum na siatce równomiernej złożonej z tysiąca punktów na tym odcinku, tzn. $e_N = \max_k |\sin(x_k) - S_N(x_k)|$ dla x_k punktów siatki.
- policz równocześnie współczynnik $\frac{e_N}{e_{2N}}$. Czy prawdą jest, że

$$\frac{e_N}{e_{2N}} \approx 2^p$$

dla jakiegoś p całkowitego np. $p = 4, 8$ lub 16?

Zadanie 5 Testowanie eksperymentalne rzędu zbieżności splajnu interpolacyjnego kubicznego bez warunków brzegowych (splajn typu *not a knot*).

Powtórz zadanie 4, ale dla splajnów interpolacyjnych otrzymanych przez **spline**() bez podawania wartości pochodnych w skrajnych węzłach. Czy współczynniki $\frac{e_N}{e_{2N}}$ są te same? Tzn. czy szybkość zbieżności $\|\sin(x) - S_N\|_\infty$ jest taka sama?

Zadanie 6 Testowanie eksperymentalne rzędu zbieżności splajnu interpolacyjnego kubicznego naturalnego (warunek brzegowy - zerowanie się drugich pochodnych w końcach odcinka).

Powtórz zadanie 4, ale dla splajnów interpolacyjnych naturalnych. Tu trzeba wykorzystać funkcję z octave-forge (czyli rozszerzenia pakietu octave)
pp=csape(x,y,'variational')
- ostatni argument określa to, że splajn będzie naturalny.

Podajemy link do strony www z pomocą do funkcji csape():
`http://octave.sourceforge.net/splines/function/csape.html`

Zadanie 7 Przykład Rungego, czyli $f(x) = 1/(1 + x * x)$ i odcinek $[-5, 5]$, a zbieżność interpolacji splajnami kubicznymi.

Przetestuj jak w poprzednich zadaniach, czy splajny interpolacyjne kubiczne z podanymi warunkami na pochodne w końcach odcinka zbiegają w normie supremum do f, tzn. korzystając z funkcji octave'a **spline**() znajdź współczynniki splajnu interpolacyjnego kubicznego S_N na N węzłach równoodległych dla f na odcinku $[-5, 5]$ dla $N = 2^k N_0$ dla $N_0 = 5$ i $k = 1, 2, 3, 4, 5$ oraz narysuj wykresy f i tych splajnów dla różnych N.

Następnie oblicz dyskretną normę maksimum na siatce złożonej z tysiąca punktów na tym odcinku, tzn. $e_N = \max|\sin(x_k) - S_N \sin(x_k)|$ dla $x_k = -5 + k * 0.01$ z $k = 0, \ldots, 1000$.

Policz równocześnie współczynnik $\frac{e_N}{e_{2N}}$. Czy $\frac{e_N}{e_{2N}} \approx 2^p$ dla jakiegoś p całkowitego?

Zadanie 8 Funkcja octave'a mkpp() . Zapoznaj się z tą funkcją (**help** mkpp()). Utwórz przy pomocy mkpp() splajn kubiczny s na podziale $-1 \leq 0 \leq 1$ odcinka $[-1, 1]$ taki, że s jest wielomianem trzeciego stopnia na całym odcinku $[-1, 1]$ np. $s(x) = x^2$ lub $(x+1)^3$.

Zadanie 9 Funkcja octave'a umkpp().

Zapoznaj się z tą funkcją (**help** umkpp()).

Utwórz przy pomocy **spline**() splajn kubiczny s na podziale $-1 \leq 0 \leq 1$ odcinka $[-1, 1]$ taki, że s interpoluje wielomian trzeciego stopnia na całym odcinku $[-1, 1]$ np. $f(x) = (x+1)^3$.

Następnie sprawdź współczynniki s w bazie $\{(x - x_k)^j\}_{j=0,1,2,3}$ za pomocą umkpp() na obu przedziałach, tzn. dla $x_0 = -1$ na przedziale $[-1, 0]$ i $x_1 = 0$ na $[0, 1]$.

Zadanie 10 Znajdź za pomocą funkcji octave'a umkpp() współczynniki splajnu z zadania 3 na wszystkich pododcinkach $[k, k+1]$ w bazie $\{(x - k)^j\}_{j=0,\ldots,3}$ dla $k = -2, \ldots, 1$ czyli tam, gdzie ten B-splajn ma nośnik.

Porównaj z wynikami otrzymanymi teoretycznie.

Zadanie 11 Interpolacja splajnami liniowymi.

Dla danego równomiernego podziału odcinka $[-\pi, 2*\pi]$ na N pododcinków utwórz za pomocą mkpp() strukturę splajnu liniowego s_n interpolującego funkcję $\sin(x)$ w węzłach dla $N = 3, 6, 9, 18$.

- Narysuj wykresy funkcji $\sin(x)$ oraz tych splajnów liniowych na jednym wykresie
- Policz przybliżone normy maksimum (na 1000 punktach z tego odcinka) błędu $e_N = \|\sin - s_n\|_\infty$.
- Policz współczynnik $\frac{e_N}{e_{2N}}$. Czy widać, że

$$\frac{e_N}{e_{2N}} \approx 2^p$$

dla jakiegoś p całkowitego np. $p = 2, 4$ lub 8?

W tym zadaniu można wykorzystać funkcję z następnego zadania, tj. zadania 12.

Zadanie 12 Napisz w octavie funkcję **function** pp=linspline(x,y), która dla

- x wektora $N+1$ różnych węzłów uszeregowanych ($a = x_0 < x_1 \ldots < x_N = b$)
- y - wektora $N+1$ wartości funkcji $y = f(x)$

zwróci w strukturze *pp* współczynniki splajnu liniowego s dla podziału zadanego węzłami x_k.

Strukturę należy utworzyć funkcją octave'a mkpp() w taki sposób, aby można było obliczyć wartość tego splajnu w punkcie (tablicy punktów) za pomocą funkcji octave'a ppval().

Zadanie 13 Testowanie rzędu zbieżności interpolacji splajnami liniowymi w zależności od gładkości funkcji.

Dla funkcji

$$f_j(x) = (x)_+^j = \begin{cases} 0 & x < 0 \\ x^j & x > 0 \end{cases} \qquad j = 1, 2, 3$$

oraz dla podziału odcinka $[a, b]$ z $a = -\pi$ i $b = 3$ na węzły równoodległe

$$\{x_k = -\pi + k * h\}$$

dla $h = (b-a)/N$ i $N = 4, 8, 16, 32, 64$.

Przetestuj rząd zbieżności splajnu liniowego interpolacyjnego.

Bardziej szczegółowo:

- przy pomocy funkcji octave'a z zadania 12 znajdź współczynniki odpowiedniego splajnu liniowego $s_N f_j$.
- policz przybliżoną normę maksimum błędu (na 1000 równomiernych punktach), tzn. przybliżenie

$$e_N = \|s_N f_j - f_j\|_{\infty,[a,b]}.$$

- policz współczynniki $\frac{e_N}{e_{2N}}$. Czy widać, że

$$\frac{e_N}{e_{2N}} \approx 2^p$$

dla jakiegoś p całkowitego np. $p = 2, 4$ lub 8? Czy widać różnicę dla różnych j?

Rozdział 6

Wielomiany Czebyszewa

Wielomiany Czebyszewa definiujemy rekurencyjnie: $T_{-1} \equiv 0$, $T_0 \equiv 1$ oraz

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \tag{6.1}$$

albo ze wzoru:

$$T_n(x) = \cos(n * \arccos(x)) \qquad x \in [-1, 1]. \tag{6.2}$$

W tym rozdziale przetestujemy podstawowe własności tych wielomianów.

Zadanie 1 Napisz rekurencyjną funkcję octave'a obliczającą wartość wielomianu zadanego poprzez współczynniki w bazie Czebyszewa wprost ze wzoru rekurencyjnego (6.1), tzn. parametrami funkcji będą:

- x argument dla którego chcemy obliczyć wartość wielomianu
- a - wektor zawierający współczynniki $a_0, \ldots, a_n$ takie, że $w(x) = \sum_{k=0}^{n} a_k T_k(x)$.
- n - stopień wielomianu (parametr opcjonalny, można tę wartość uzyskać z wektora a)

Funkcja octave'a ma zwrócić wartość $w(x)$.

Przetestuj tę funkcję dla $w(x) = T_{15}(x)$, tzn. narysuj dwa wykresy $T_{15}(x)$ raz korzystając z tej funkcji oraz drugi raz korzystając wprost ze wzoru (6.2).

Zadanie 2 Napisz nierekurencyjną funkcję octave'a

```
function [Y]=Czebyszew(X,a,n)
```

obliczającą wartość wielomianu zadanego poprzez współczynniki w bazie Czebyszewa z wzoru (6.1), tzn. parametry wejściowe funkcji to:

- macierz $X = (x_{ij})_{ij}$, wymiaru $k \times l$
- wektor współczynników

  $$a = [a_0, \ldots, a_n]$$

  takich, że

  $$w(x) = \sum_{k=0}^{n} a_k T_k(x).$$

- n stopień wielomianu; ten parametr może być opcjonalny, jeśli go nie podamy to funkcja powinna przyjąć, że n to wymiar wektora współczynników minus jeden.

Funkcja octave'a powinna zwrócić macierz $Y = (y_{ij})_{ij}$, wymiaru $k \times l$ z wartościami $y_{ij} = w(x_{ij})$.

Funkcja ma korzystać z wzoru rekurencyjnego (6.1), ale jako funkcja octave'a ma być **nierekurencyjna**. Koszt algorytmu powinien wynosić $k * l * O(n)$.

Narysuj dwa wykres $T_3(x)$ - jeden korzystając z tej funkcji oraz drugi - ze wzoru (6.2).

Zadanie 3 Sprawdź obliczeniowo, czy powyższe wzory na wielomiany Czebyszewa: wzór definiujący wielomiany Czebyszewa rekurencyjnie (6.1) i (6.2), są zgodne na odcinku $[-1, 1]$. Tzn. dla dużej ilości np. 10000 losowych punktów na $[-1, 1]$ policz wartości T_{32} ze wzoru

$$T_{32}(x) = \cos(32 * \arccos(x))$$

oraz ze wzoru rekurencyjnego (6.1).

Policz maksimum wartości absolutnych z różnicy między wartościami wielomianu obliczonymi obiema metodami. Porównaj czas obliczania wartości tego wielomianu obydwoma wzorami.

Powtórz testy ale dla wielomianów innych stopni, np. T_7, T_{50}.

Zadanie 4 Narysuj korzystając z wzoru

$$T_n(x) = \cos(n * \arccos(x))$$

wykres wielomianów T_n dla $n = 0, 1, 2, 3, 4, 5$. Policz dyskretną normę maksimum na siatce. Czy wynosi jeden? Policz na wykresie zera oraz ekstrema każdego z wielomianów.

Zadanie 5 Wyznacz zera i ekstrema T_{10} i T_{15} ze wzoru $T_n(x) = \cos(n * \arccos(x))$. Sprawdź wartości tych wielomianów w jego zerach i ekstremach.

Zadanie 6 Sprawdzenie własności optymalności zer wielomianu Czebyszewa jako węzłów interpolacji Lagrange'a.

Chcemy eksperymentalnie sprawdzić, czy wielomian $2^{-n}T_{N+1}(x)$ ma minimalną normę supremum na $[-1, 1]$ wśród wszystkich wielomianów postaci $\Pi_{k=0}^{N}(x - x_k)$.

Policz dla $N + 1$ różnych losowych węzłów x_k, $k = 0, \ldots, N$ z odcinka $[-1, 1]$ dla $N = 2, 4, 8, 16, 32$ i kilkuset różnych losowych zestawów przybliżoną normę supremum wielomianu $\Pi_{k=0}^{n}(x-x_k)$.

Przeprowadź testy również dla węzłów Czebyszewa (czyli zer T_{N+1}) oraz węzłów równoodległych dla tego samego N.

Zadanie 7 Sprawdzenie własności ekstremalnej wielomianów Czebyszewa.

Z teorii wiadomo, że wielomian $2^{-n}T_{N+1}(x)$ ma minimalną normę supremum na $[-1, 1]$ wśród wszystkich wielomianów postaci

$$w(x) = x^{n+1} + \sum_{k=0}^{n} a_k x^k.$$

Chcemy sprawdzić eksperymentalnie, czy ta własność się potwierdzi.

Policz dla $N + 1$ losowych współczynników a_k dla $k = 0, \ldots, N$ przybliżoną normę supremum wielomianu:

$$w(x) = x^{n+1} + \sum_{k=0}^{n} a_k x^k.$$

Przetestuj dla $N = 2, 4, 8, 16, 32$ i co najmniej kilku tysięcy różnych losowych zestawów węzłów. Węzły losujemy funkcją **randn**() zwracającą liczby losowe o rozkładzie normalnym.

Zadanie 8 Sprawdzenie własności ekstremalnej wielomianów Czebyszewa przyjmujących ustaloną wartość poza $[-1, 1]$.

Chcemy sprawdzić, czy dla dowolnego punktu $a = 2$ i $b \neq 1$ wielomian

$$w^*(x) = T_n(x)/T_n(2)$$

ma minimalną normę supremum na $[-1, 1]$ wśród wszystkich wielomianów w stopnia nie większego od $n+1$ przyjmujących wartość b w punkcie a.

Policz dla n losowych współczynników a_k z $k = 1, \ldots, n$ dla $n = 2, 4, 8, 16, 32$ i co najmniej kilku tysięcy różnych losowych zestawów węzłów przybliżoną normę supremum wielomianu

$$w(x) = 1 + \sum_{k=1}^{n} a_k (x-2)^k$$

i porównaj z normą supremum w^* na $[-1, 1]$ równą $1/T_n(2)$ (dlaczego tyle ona wynosi?).

Węzły losujemy funkcją **randn**() zwracającą liczby losowe o rozkładzie normalnym.

Zadanie 9 Funkcja octave'a służąca przybliżonemu obliczaniu całek po odcinkach (zadanie pomocnicze)

Zapoznaj się z funkcją octave'a **quad**().

Policz całkę przy pomocy funkcji **quad**() z $\sin(x)$ na $[-\pi, 2 * \pi]$.

Zadanie 10 (trudne) **Iloczyn skalarny typu $L^2_w(a, b)$.**

Napisz funkcję octave'a:

```
function [nl2]=IlSkL2w(FCN,GCN,a,b,FCNW)

 komendy octave'a

 nl2 = ....

endfunction
```

która oblicza iloczyn skalarny typu $L^2_w(a, b)$ z wagą $w(x)$ tzn.:

$$(f, g)_{L^2_w(a,b)} = \int_a^b f(x)g(x)w(x)\,dx$$

dla danej funkcji wagowej $w(x)$ i funkcji f, g.

Parametry funkcji to:

- wskaźniki do funkcji FCN, GCN do dwóch funkcji octave'a

```
function y=f(x)

  y = ......
endfunction


function y=g(x)

  y = ......
endfunction
```

obliczających odpowiednio wartości funkcji $f(x)$ i $g(x)$ dla $x \in [a, b]$.

- a, b - końce odcinka całkowania $[a, b]$,
- $FCNW$ to wskaźnik do funkcji wagowej

```
function y=w(x)

  y = ......
endfunction
```

Funkcja powinna zwrócić przybliżenie iloczynu skalarnego obliczone za pomocą funkcji octave'a **quad**().

W przypadku wywołania funkcji IlSkL2() tylko z czterema pierwszymi argumentami (tzn. bez podania wskaźnika do wagi) funkcja powinna zwrócić iloczyn skalarny z wagą $w \equiv 1$.

Zadanie 11 Ortogonalność wielomianów Czebyszewa w $L^2_{\frac{1}{\sqrt{1-x^2}}}(-1, 1)$.

Sprawdź eksperymentalnie w octavie, np. za pomocą funkcji **quad**() lub własnej funkcji z poprzedniego zadania, czy wielomiany Czebyszewa tworzą układ ortogonalny w $L^2_{\frac{1}{\sqrt{1-x^2}}}(-1, 1)$, tzn. czy prawdą jest, że

$$\int_{-1}^{1} T_n(x)T_m(x)\frac{1}{\sqrt{1-x^2}}\,dx = \begin{cases} 0 & m \neq n, \\ \frac{\pi}{2} & m = n > 0, \\ \pi & m = n = 0. \end{cases}$$

Sprawdź powyższe zależności dla różnych m, n różnej wielkości np. dla $m = 0, 1, 2, 3, 100, 1000$ i $n = 0, 3, 10, 500, 1004$.

Rozdział 7

Kwadratury

Zadania z tego rozdziału służą przetestowaniu najprostszych kwadratur numerycznych, czyli metod przybliżonego obliczania całek po odcinkach.

W tym rozdziale zapoznamy się także z funkcją octave'a **quad**() służącą przybliżonemu obliczaniu całek jednowymiarowych postaci

$$\int_a^b f(t)\,dt \qquad -\infty \leq a < b \leq +\infty,$$

Możemy więc znajdować przybliżenia całek po całej prostej rzeczywistej czy półprostych.

Zadanie 1 Zapoznaj się z pomocą do funkcji octave'a **quad**().

Policz przy pomocy **quad**() całkę $\int_a^b f(t)\,dt$ dla następujących funkcji i odcinków:

- x^2 na $[-1, 1]$
- x^9 na $[0, 1]$
- $\sin(x)$ na $[0, \pi]$
- $\cos(100 * x)$ na $[0, \pi]$
- $\cos(100 * x) * \cos(1000 * x)$ na $[0, \pi]$

Czy wyniki są zgodne z teorią?

Zadanie 2 Złożona kwadratura trapezów.

(a) Zaprogramuj w octavie funkcję
function c=kwadtrapez(FCN,a,b,n)

obliczającą złożoną kwadraturę trapezów:

$$T_n f = h\left(0.5[f(a) + f(b)] + \sum_{k=1}^{n-1} f(a + k * h)\right)$$

dla $h = (b - a)/n$.

Parametry funkcji:

- FCN - wskaźnik do funkcji octave'a **function** y=f(x) obliczającej wartość funkcji podcałkowej
- a - lewy koniec odcinka
- b - prawy koniec odcinka
- n - ilość obliczeń wartości funkcji podcałkowej w kwadraturze trapezów minus jeden (tak, jak we wzorze powyżej)

Funkcja zwraca przybliżoną wartość całki obliczoną za pomocą powyższego wzoru, tzn. złożonej kwadratury trapezów.

Funkcja powinna działać również jeśli ją wywołamy tylko z trzema parametrami. Wtedy n powinno domyślnie przyjąć wartość sto.

(b) Przetestuj funkcję octave'a z poprzedniego podpunktu dla $N_k = 2^k N_0$ $k = 0, 1, 2, \ldots$ z ustalonym $N_0 = 5$ licząc:

$$\frac{E_N}{E_{2N}}$$

dla błędu

$$E_N = |\int_a^b f\, dt - T_N f|$$

dla następujących funkcji, dla których wartości całek znamy:

- x^2 na $[0, 1]$,
- $\sin(x)$ na $[0, \pi]$ i $N_0 = 5$ - funkcja analityczna,
- $\sin(100 * x)$ na $[0, \pi]$ i $N_0 = 5$ - funkcja analityczna, silnie oscylująca - duże wartości drugiej pochodnej,
- $x^{j+0.5}$ na $[0, 1]$ dla $j = 0, 1, 2$ czyli funkcji w $C^j([0, 1])$ i o nieograniczonej w otoczeniu zera $j + 1$ pochodnej.

Porównaj wyniki obliczane kwadraturą trapezów z wynikami funkcji **quad**(). Można sprawdzić dla jakiego n błąd

obliczony złożoną kwadraturą trapezów jest na poziomie błędu funkcji octave **quad**() - i porównać ilość wywołań funkcji f przez obie procedury.

Zadanie 3 Czy wielomiany Czebyszewa tworzą układ funkcji ortogonalnych w L^2 na $[-1,1]$ z wagą $\frac{1}{\sqrt{1-x^2}}$.

Policz za pomocą funkcji octave **quad**():

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x*x}} T_n(x) T_m(x)\, dx$$

dla $m = 2$ i $n = 3$. Czy wynik jest zgodny z teorią?

Zadanie 4 Kwadratura Gaussa-Czebyszewa dla

$$I(f) = \int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} f(x)\, dx$$

czyli

$$GC_{n+1}f = \frac{\pi}{n+1} \sum_{k=0}^{n} f(x_k) = \frac{\pi}{n+1} \sum_{k=0}^{n} f\left(\cos\left(\frac{0.5*\pi + k\pi}{n+1}\right)\right)$$

dla x_k zer T_{n+1} - $n+1$ wielomianu Czebyszewa.

Zaimplementuj funkcję **function** c=gaussczeb(FCN,n) obliczającą wartość kwadratury $GC_n f$. Parametry - wskaźnik do funkcji **function** y=f(x) i ilość punktów kwadratury.

Przetestuj jej działanie:

- policz przybliżenia całek $I(T_k^2)$. Czy $GC_n(T_k^2)$ przybliża $\pi/2$ dla $k > 0$ dla $n = 20, 40, 80$?
- policz przybliżenie całki z wielomianów różnych stopni dla $n = 4, 10, 20$ - czy kwadratura jest dokładna dla wielomianów Czebyszewa stopnia $0 < k < 2n$ (tzn. czy zwraca zero)?
- dla funkcji matematycznej $f(x) = \exp(\arccos(x))$, której całkę z tą wagą możemy obliczyć dokładnie.

  Porównaj wartość kwadratury z wynikiem dokładnym dla $n = 10, 20, 40, 80$, tzn. policz

  $$e_n = |GC_n(f) - I(f)|$$

  przyjmując za $I(f)$ wartość dokładną.

- analogicznie do poprzedniego podpunktu tego zadania, porównaj wynik obliczony za pomocą kwadratury Gaussa - Czebyszewa, tzn. $GC_n(f)$, dla funkcji $f(x) = \exp(\arccos(x))$ z wynikiem obliczonym funkcją **quad**(). Wypisz na ekran $e_n = |GC_n(f) - c|$ dla c wartości uzyskanej za pomocą funkcji octave'a **quad**().
- Przetestuj rząd zbieżności tej kwadratury dla funkcji $f(x) = \exp(\arccos(x))$, jak w przypadku kwadratury trapezów, tzn. policz błędy e_n, analogicznie do poprzednich podpunktów i wypisz e_n/e_{2n}.

Zadanie 5 Złożona kwadratura prostokątów.

(a) Zaprogramuj w octave funkcję
function c=kwadprost(FCN,a,b,n),
która zwraca wartość złożonej kwadratury prostokątów dla funkcji f na odcinku $[a, b]$ na n równoodległych punktach:

$$P_n f = h \sum_{k=1}^{n} f(a + (k - 0.5) * h)), \qquad h = (b - a)/n.$$

Parametry funkcji:

- FCN - wskaźnik do funkcji octave'a postaci

```
function y=f(x)
  y = ......
endfunction
```

obliczającej wartość funkcji, której całkę chcemy obliczyć, tj. $f(x)$, w danym punkcie x.
- a, b - końce odcinka,
- n - ilość węzłów wykorzystywanych przez kwadraturę.

Funkcja zwraca przybliżoną wartość całki obliczoną za pomocą powyższego wzoru, tzn. złożonej kwadratury prostokątów.
Funkcja powinna działać również jeśli ją wywołamy bez ostatniego parametru. Wtedy n powinno domyślnie przyjąć ustaloną wartość, np. dwieście.

(b) Testy dla funkcji, których całki znamy.

Przetestuj tę funkcję octave'a kwadprost() dla $N_k = 2^k N_0$ $k = 0, 1, 2, \ldots$ z ustalonym $N_0 = 5$ licząc:

$$\frac{E_N}{E_{2N}}, \qquad E_N = |\int_a^b f\,dt - P_N f|$$

dla całek z następujących funkcji po odpowiednich odcinkach:

- x^2 na $[0, 1]$
- $\sin(x)$ na $[0, \pi]$ i $N_0 = 5$ - funkcja analityczna,
- $\sin(100*x)$ na $[0, \pi]$ i $N_0 = 5$ - funkcja analityczna silnie oscylująca (duże wartości drugiej pochodnej),
- $x^{j+0.5}$ na $[0, 1]$ dla $j = 0, 1, 2$ czyli funkcji w $C^j([0, 1])$ i o nieograniczonej w otoczeniu zera $j + 1$ pochodnej.

Porównaj wyniki obliczane kwadraturą prostokątów z wynikami funkcji **quad**() .

Zadanie 6 Porównaj wyniki obliczane kwadraturą prostokątów z wynikami funkcji obliczającej całkę za pomocą złożonej kwadratury trapezów.

Porównaj błędy dla wartości N i $f(x)$ z poprzedniego zadania - czyli całek, których wartość teoretycznie znamy - z wynikami dla obu kwadratur:

- dla tej samej ilości wywołań funkcji f, tzn. porównaj $P_N f$ z $T_{N-1} f$,
- porównaj obie kwadratury dla tego samego h, tzn. $T_N f$ z $P_N f$, aby ocenić błąd w terminach parametru $h = (b-a)/N$.

Zadanie 7 Kwadratura Romberga.

(a) Zaprogramuj w octave funkcję

function c=Romberg(FCN,a,b,p,N0)

z kwadraturą Romberga $R_{p,N0}$ obliczającą przybliżenie całki $\int_a^b f(t)\,dt$.

Parametry funkcji:

- FCN wskaźnik do funkcji octave'a **function** y=f(x) obliczającej wartość funkcji podcałkowej

- a lewy koniec odcinka
- b prawy koniec odcinka
- p ilość poziomów w kwadraturze Romberga
- $N0$ startowa ilość punktów w kwadraturze Romberga

Funkcja zwraca przybliżoną wartość całki obliczoną za pomocą kwadratury Romberga.

Funkcja powinna działać również jeśli ją wywołamy tylko z trzema albo czterema parametrami. Parametr p powinien wtedy domyślnie przyjąć wartość 5, a $N0$ wartość sto.

(b) Przetestuj kwadraturę Romberga dla całki $\int_a^b f\,dt$ dla następujących funkcji, odcinków i wartości parametrów $N0$ i p:

- $\sin(x)$ dla $[0, \pi]$
- $\sin(x)$ dla $[0, 100 * \pi]$
- $\sin(10 * x)$ dla $[0, \pi]$
- $x^{j+0.5}$ na $[0, 1]$ dla $j = 0, 1, 2, \ldots, 10$,

dla $N0 = 100$ i dla $p = 2, 3, 4, 5, 6$.

Czy dla $\sqrt{x}$ i rosnącego p błąd maleje?

Czy stosunek E_N/E_{2N} maleje tak samo dla wszystkich funkcji?

Tutaj użyliśmy oznaczenia na błąd $E_N = |\int_a^b f\,dt - R_{p,N0}|$.

Rozdział 8

Rozwiązywanie równań nieliniowych

W poniższych zadaniach przetestujemy jak działają cztery metody rozwiązywania równań nieliniowych skalarnych, tzn. metoda bisekcji, metoda Newtona, metoda siecznych i metoda iteracji prostych oraz dwie metody rozwiązywania układów równań nieliniowych, czyli wielowymiarowa metoda Newtona i metoda iteracji prostych w najprostszej formie.

Zapoznamy się również z funkcjami octave'a **fzero**() oraz fsolve () służącymi do rozwiązywania równań nieliniowych skalarnych i układów równań nieliniowych.

Zadanie 1 Zaimplementuj metodę bisekcji w skrypcie - dla rozwiązania równania $\cos(x) = 0$ na odcinku $[0, 2]$. Przetestuj, czy funkcja znajdzie dobre przybliżenie $\pi/2$.

Zadanie 2 Przetestuj metodę bisekcji z poprzedniego zadania do rozwiązania innych problemów:

$$x^2 - 2 = 0$$
$$\exp(x) = 2$$
$$\cos(10 * x) = 0$$

startując z odcinka $[0, 110]$.

Zadanie 3 Napisz funkcję octave'a:

```
function [y,iter,kod]=bisekcja(FCN,a,b,...
                                    tola,maxit),
```

której parametrami będą

- wskaźnik do funkcji FCN (inaczej: *function handle*),
- a, b - końce przedziału,
- tola - żądana tolerancja bezwzględna - błąd powinien być mniejszy od tola,
- maxit - maksymalna ilość iteracji.

Funkcja ma zwrócić

- y - przybliżone rozwiązanie,
- iter - ilość iteracji,
- kod - kod wyniku:
  - 0 metoda zbiegła,
  - 1 - metoda zatrzymała się z powodu przekroczenia maksymalnej ilości iteracji,
  - 2 - wartości w końcach odcinka funkcji mają ten sam znak.

Funkcja ma działać również jeśli podamy tylko trzy lub cztery pierwsze argumenty - wtedy maksymalna ilość iteracji domyślnie powinna wynosić sto, a tolerancja 10^{-5}.

Zadanie 4 Zapoznaj się z pomocą dla funkcji octave'a **fzero**() - rozwiązującą skalarne równania nieliniowe, przetestuj ją na kilku prostych przykładach skalarnych np.

- $\cos(x) = 0$,
- $x^2 - 2 = 0$,
- $x^{10} - 2 = 0$,
- $x - \sin(x) = 0$

i innych prostych równaniach nieliniowych.

Proszę przetestować tę funkcję dla różnych wartości początkowego przedziału $x_0 = [a, b]$.

Zadanie 5 Zapoznaj się z pomocą dla funkcji octave'a fsolve () - rozwiązującą układy równań nieliniowych. Przetestuj ją na kilku prostych przykładach skalarnych, np. na tych z poprzedniego zadania.

Zadanie 6 Zaimplementuj metodę Newtona w octavie i przetestuj jej zbieżność dla następujących funkcji:

f1(x)= $x * x - 2$ z $x_0 = 2$,

f2(x)= $x * x * x - 27$ z $x_0 = 27$,

f3(x)= $\exp(x) - 2$ z $x_0 = 10, -10, 1e3$,

f4(x)= $\sin(x)$ dla $x_0 = 2$,

f5(x)= $\cos(x)$ z $x_0 = 2$

f6(x)= $(x-2)^k$ x $x_0 = 3$ dla $k = 2, 4, 8, 16$,

f7(x)= $x * x - 2$ z $x_0 = 10^6$, sprawdź, czy metoda Newtona zbiega, a jeśli tak - to czy zbiega ona kwadratowo,

f8(x)= $1/x - a$ dla danych $a = 0.5, 2, 4, 100$ (oczywiście implementując bez dzielenia) jak dobrać x_0?

Dla wszystkich tych funkcji znamy rozwiązania, więc można wyświetlać równocześnie na ekranie błąd

$$e_n = x_n - r$$

(tutaj r - znane rozwiązanie) i badać eksperymentalnie rząd zbieżności, tzn. drukować na ekranie stosunki błędu bieżącego do poprzedniego w odpowiedniej potędze:

$$|e_n|/|e_{n-1}|^p$$

dla $p = 1, 2, 3$.

Proszę powtórzyć testy z innymi różnymi wartościami startowymi x_0.

Zadanie 7 Powtórz zadanie 6 zastępując metodę Newtona przez metodę siecznych.

Przetestuj, czy

$$e_n/(e_{n-1}e_{n-2})$$

asymptotycznie zbiega do stałej, i czy

$$|e_n|/|e_{n-1}|^p$$

dla $p = (1 + \sqrt{5})/2$ zbiega do stałej np. dla $x * x - 2$ z $x_0 = 2$ lub innych równań z poprzedniego zadania.

Za x_1 możemy przyjąć $x_0 + 10^{-7}$.

Zadanie 8 Sprawdź, czy metoda iteracji prostych

$$x_n = \cos(x_{n-1})$$

zbiega do $x^* = \cos(x^*)$.

Zbadaj eksperymentalnie, czy zbieżność jest liniowa, tzn. czy

$$\frac{|x^* - x_n|}{|x^* - x_{n-1}|}$$

zbiega do stałej. Za dobre przybliżenie x^* można przyjąć rozwiązanie równania obliczone za pomocą wywołania funkcji octave'a: **fzero**().

Zadanie 9 Zaimplementuj przybliżoną metodę Newtona, w której pochodną przybliżamy ilorazem różnicowym, tzn.

$$x_{n+1} = x_n - \frac{f(x_n)}{\frac{f(x_n+h)-f(x_n)}{h}} = x_n - \frac{f(x_n) * h}{f(x_n+h) - f(x_n)}$$

dla ustalonego h.

Przetestuj różne h, np. $h = 10^{-4}, 10^{-7}, 10^{-10}$ itp.

Porównać zbieżność z dokładną metodą Newtona (szczególnie ostatnie iteracje) dla funkcji z zadania 6.

Zadanie 10 Rozwikływanie funkcji:

Dla funkcji $y(x)$ zadanej w sposób uwikłany równaniem

$$g(x, y) = 2x^2 + 3y^2 - 3 = 0$$

znajdź przybliżone wartości $y_k = y(x_k)$ dla $x_k = k * h$ dla $k = -N, \ldots, N$ i $h = 1/N$, tzn. spełniające $g(x_k, y_k) = 0$.

Testuj dla różnych wartości N, np. $N = 10, 20, 40, 80, \ldots$.

oblicz y_k rozwiązując równanie

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

korzystając z odpowiedniej funkcji octave'a lub własnej implementacji metody Newtona.

Jak w kolejnych krokach dobierać przybliżenie startowe w metodzie Newtona?

Zadanie 11 Odwracanie funkcji

Rozpatrzmy daną funkcję, np.

$$f(x) = \sin(x) + 2 * x.$$

Znajdź wartości funkcji odwrotnej f^{-1} na odcinku $[0, 5]$ na siatce $k * h$ dla $k = 0, .., 100$. Narysuj wykres funkcji f^{-1}.

Sam wykres można narysować dużo prościej bez wyliczania wartości f^{-1}. Jak to zrobić w octave'ie?

Zadanie 12 Wielowymiarowa metoda Newtona

Zaimplementuj wielowymiarową metodę Newtona w wersji z dokładnym Jakobianem i w wersji, gdy Jakobian przybliżamy różnicami dzielonymi z parametrem h.

Zastosuj tę metodę do rozwiązania układu

$$\begin{aligned} f_1(x_1, x_2) &= x_1 + 2x_2 = 1 \\ f_2(x_1, x_2) &= 3 * x_1^2 + x_2^2 = 1 \end{aligned}$$

dla różnych przybliżeń początkowych.

W przypadku zbieżności policz

$$\frac{\|x - x_n\|_p}{\|x - x_{n-1}\|_p^q}$$

dla $p = 1, 2, \infty$ oraz $q = 1, 2, 3$. Czy ten iloraz zbiega do stałej dla jakiegoś z tych q?

Funkcję $F(x) := (f_1(x), f_2(x))^T$ w octavie można zdefiniować następująco:

```
function y=F(x)
#x- wektor pionowy

  y=[[1  2]*x-1;3.*x(1)*x(1)+x(2)*x(2)-1];

endfunction
```

Zadanie 13 Wielowymiarowa metoda iteracji prostych

Zastosuj metodę iteracji prostych

$$x_n = x_n - aF(x)$$

do układu z zadania 12, tzn. przyjmijmy, że $F(x) := (f_1(x), f_2(x))^T$, a parametr $a \neq 0$ np. $a = 1, -1, 10^{-1}$ itp.

Czy metoda zawsze zbiega, a jeśli tak - to jak szybko? W przypadku zbieżności policz

$$\frac{\|x - x_n\|_p}{\|x - x_{n-1}\|_p}$$

dla $p = 1, 2, \infty$.

Zadanie 14 Do układu równań z zadania 12 zastosuj funkcję fsolve () - porównaj z implementacją metody Newtona w zadaniu 12, która była zastosowana do rozwiązania tego układu.

Czy obie metody zbiegają do tego samego rozwiązania dla tych samych wartości wektorów startowych?

Sprawdź - za pomocą funkcji **tic** i **toc** - czas potrzebny do rozwiązania tego układu równań za pomocą obu metod.

Rozdział 9

Układy równań liniowych - rozkłady typu LU i LL'

W tym rozdziale zapoznamy się z metodami służących do rozwiązywania układów równań liniowych przy pomocy uzyskiwaniu odpowiednich rozkładów macierzy typu LU i LL^T oraz obliczaniu macierzy odwrotnej.

Zapoznamy się z odpowiednimi funkcjami octave'a służącymi znajdowaniu odpowiednich rozkładów, czy rozwiązywaniu układów równań liniowych z pomocą tych rozkładów.

Podstawowe funkcje i operatory octave'a związane z rozkładem LU to:

- [L,U,P]=**lu**(A) - funkcja zwracająca czynniki rozkład LU nieosobliwej macierzy A, czyli $PA = LU$,
- [R]=**chol**(A) - funkcja zwracająca czynnik rozkładu Choleskiego macierzy symetrycznej dodatnio określonej $A = A^T > 0$, czyli $A = R^T R$, (zazwyczaj w literaturze podaje się ten rozkład w terminach czynnika $L = R^T$),
- operator x=A\b - operator rozwiązujący układ równań $Ax = b$ dla A macierzy nieosobliwej i b wektora prawej strony,
- **inv**(A) - funkcja zwracająca macierz odwrotną do A macierzy nieosobliwej,
- **cond**(A) - funkcja zwracająca współczynnik uwarunkowania macierzy A, czyli $\|A\|_2 * \|A^{-1}\|_2$.

Zadanie 1 Operator octave'a \ służący m.in. do rozwiązywaniu układów równań liniowych w octavie.

Przetestuj operator octave'a \ dla układu równań z macierzą $A = [1, 2; 3, 4]$ i $x = [1; 3]$ z f=A∗x. Czy y=A\f jest rozwiązaniem tego układu?

Policz normy residualną $\|Ay - f\|_p$ i normę błędu $\|x - x\|_p$ dla $p = 1, 2$.

Powtórz testy dla macierzy osobliwej $[1, 2; 3, 6]$ i prawie osobliwej $[1, 2; 3, 6 - 10^{-6}]$. Co się wtedy dzieje?

Zadanie 2 Przetestuj solver octave'a, tzn operator backslash dla dwóch prostych układów równań liniowych z macierzami nieosobliwymi: $A_1 = [1, 1; 1, 0.98]$ i $b = [2; 1]$, $A_2 = [1, 1; 1, 0.99]$ i wektorem prawej strony układu równań liniowych $b = [2; 1]$ jak w poprzednim zadaniu. Policz w normie drugiej różnicę rozwiązań, normę Frobeniusa różnicy $A_1 - A_2$ oraz uwarunkowania tych macierzy korzystając z funkcji octave'a **cond**().

Powtórz zadanie dla macierzy: $A_1 = [1, 1; 1, 1 - 2\epsilon]$ i $b = [2; 1]$, $A_2 = [1, 1; 1, 1 - \epsilon]$ dla $\epsilon = 10^{-p}$ z $p = 3, 4, \ldots, 16$.

Zadanie 3 Funkcja **inv**().

Zapoznaj się z pomocą do funkcji: **inv**(). Przetestuj, obliczając macierz odwrotną do $A = [1, 1; 1, -1]$. Czy macierz B obliczona za pomocą tej funkcji rzeczywiście jest macierzą odwrotną?

Policz normy pierwszą i Frobeniusa $\|A * B - I\|$ oraz $\|B * A - I\|$.

Zastosuj funkcje do rozwiązywania układu równań liniowych $Ax = f$ dla znanego x (liczymy $f = A * x$). Policz y jako iloczyn B z f i policz błędy residualny $\|Ay - f\|$ i $\|x - y\|$ w normie pierwszej i drugiej.

Powtórz to zadanie dla macierzy $N \times N$ losowej (funkcja octave'a **rand**() zwraca macierz losową) dla $n = 10, 50, 250, 1250$ ze znanym rozwiązaniem - oszacuj czas przy pomocy funkcji **tic** i **toc**. Porównaj czas i błędy w normie pierwszej dla rozwiązania uzyskanego przy pomocy operatora octave'a \.

Zadanie 4 Funkcje **lu**() i **chol**().

- Zapoznaj się z pomocą do funkcji: **lu**() i **chol**().
- Dla macierzy $A = [1, 1; 1, -1]$ znajdź jej rozkłady: $PA = LU$ i rozkład Choleskiego $A = L_1^T L_1$.

- Sprawdź te rozkłady licząc normy macierzowe pierwszą i Frobeniusa błędów np. $PA - LU$ i $A - L_1^T L_1$.
- Zastosuj te rozkłady do znalezienia rozwiązania układu równań liniowych $Ax = f$ ze znanym rozwiązaniem np. $x = [1; 1]$ i $f = [2; 0]$.
- Policz normy pierwszą i drugą wektorowe pomiędzy x, a wynikiem algorytmu w, polegającym na zastosowaniu odpowiedniego rozkładu, oraz takie same normy residualne, tzn. normy $Aw - f$.

Zadanie 5 W octavie przetestuj eliminacje Gaussa z częściowym wyborem i bez wyboru dla macierzy $A = [e,\ 1; 1,\ 1]$ z $e = eps/10$ (funkcja octave'a **eps** zwraca epsilon maszynowy) i wektorem prawej strony $f = [1; 0]^T$. Trzeba tu zaprogramować samodzielnie eliminację Gaussa bez wyboru elementu głównego, tzn. bez permutacji ani wierszy, ani kolumn dla macierzy 2×2.

Zadanie 6 Testy solvera octave'a dla macierzy Hilberta $H(N)$

Polecenie octave'a **hilb**() tworzy macierz Hilberta.

- Stwórz macierz $H(N)$ dla $N = 10, 20, 30$,
- Dla znanego rozwiązania, np. stałego równego jeden na każdej pozycji, czyli $sol_k = 1$, policz $H(N) * sol = f$,
- Rozwiąż w octavie układ z $H(N)$ i f,
- Policz normy (1,2 itd) $\|H(N)x - f\|$ i $\|x - sol\|$ dla x rozwiązania wyliczonego przez octave,
- Policz uwarunkowanie macierzy Hilberta dla powyższych N.

Zadanie 7 Powtórz zadanie 6 w arytmetyce pojedynczej precyzji.

Należy tu sztucznie wymusić, za pomocą funkcji octave'a single (), aby zmienne były zmiennymi pojedynczej precyzji.

Zadanie 8 Przetestuj funkcję **invhilb**() tworzącą macierz odwrotną do macierzy Hilberta (por. zadanie 6).

- Policz normy: macierzowa normę Frobeniusa i normę macierzową pierwszą różnicy macierzy $E = H(N) * iH(N) - I$ dla $N = 10, 20, 30$, gdzie $iH(N)$ macierz odwrotna do macierzy Hilberta obliczona przez **invhilb**().

- Wykorzystaj $iH(N)$ do rozwiązania układu równań z macierzą Hilberta $H(N)$, tzn.:
  (a) stwórz macierze $H(N)$ oraz $iH(N)$ dla $N = 10, 20, 30$ korzystając z funkcji **hilb**() i **invhilb**(),
  (b) dla znanego rozwiązania sol, policz $H(N) * sol = f$,
  (c) rozwiąż układ $H(N)x = f$ mnożąc f przez $iH(N)$, tzn. $x = iH(N) * f$,
  (d) policz normy $\|H(N)x - f\|_p$ i $\|x - sol\|_p$ dla $p = 1, 2, \infty$.

Zadanie 9 Stwórz w octavie macierz trójdiagonalną $A = (a_{ij})_{i,j=1}^{n}$ wymiaru $n \times n$ z 2 na diagonali i -1 na pod- i nad diagonalą, tzn.

$$a_{i,j} = \begin{cases} 2 & i = j \\ -1 & |i - j| = 1 \\ 0 & |i - j| > 1 \end{cases}, \qquad i, j = 1, \ldots, n.$$

przy pomocy funkcji octave'a **diag**(), jak i wprost w pętli. Porównaj czas używając **tic** i **toc** dla $n = 10, 100, 1000$.

Policz uwarunkowanie macierzy dla różnych N.

Policz macierz odwrotną przy pomocy **inv**() (czy jest trójdiagonalna?).

Zadanie 10 Sprawdź z wykorzystaniem funkcji octave'a **chol**(), czy macierz z zadania 9 jest symetryczna i dodatnio określona.

Zadanie 11 Zaimplementuj rozkład LU dla macierzy trójdiagonalnej $N \times N$ bez wyboru elementu głównego, tzn. stwórz własną funkcję octave'a:

function [x,d,l]=lu3diag(a,b,c,f,N)

Parametry funkcji:

- a, b, c - przekątna, podprzekątna i nadprzekątna macierzy A,
- f - wektor prawej strony,
- N - wymiar zadania - długość przekątnej a.

Funkcja powinna zwrócić x - rozwiązanie $Ax = f$ oraz czynniki rozkładu macierzy $A = LU$, czyli diagonalę macierzy górnotrójkątnej U w wektorze d i poddiagonalę L - macierzy dolnotrójkątnej, trójdiagonalnej z jedynkami na diagonali, czyli wektor l.

Naddiagonala U równa się nadiagonali A, tzn. wektorowi c.

Zadanie 12 Zastosuj funkcję z zadania 11 do uzyskania rozkładu LU macierzy z zadania 9 dla $N = 4, 10, 15$. Porównaj z wynikiem uzyskanym przy pomocy funkcji octave'a **lu**().

Zadanie 13 Zastosuj funkcję z zadania 11 do rozwiązania układu równań z macierzą z zadania 9 dla $N = 10, 100, 1000$, czy 10000:

- Rozwiąż układ dla znanego rozwiązania, np. $x = (1, \ldots, 1)^T$,
- Porównaj czas rozwiązywania własnym algorytmem i algorytmem octave'a, czyli operatorem A\f,
- Policz błąd rezydualny i błąd rzeczywisty w normie drugiej, tzn. $\|x - \tilde{x}\|_2$ i $\|f - A * \tilde{x}\|_2$, gdzie $f = A * x$, a $\tilde{x}$ to rozwiązanie obliczone przez octave'a lub obliczone własnym algorytmem.

Zadanie 14 Zaimplementuj rozkład LU dla macierzy trójdiagonalnej z częściowym wyborem elementu głównego.

Następnie testuj ten rozkład jak w zadaniu 13.

Zastosuj ten rozkład do rozwiązania układu równań z macierzą z poprzedniego zadania A i z macierzą $A - 1.5 * I$.

Zadanie 15 Zaprogramuj rozkład Choleskiego typu dla macierzy $A = A^T > 0$ trójdiagonalnej, tzn. policz L dolnotrójkątną z jedną poddiagonalą (czyli reprezentowaną przez dwa wektory z przekątną i podprzekątną) taką, że $A = L^T L$.

Zastosuj do rozwiązania układu równań z zadania 9.

Testuj analogicznie do zadania 13.

Porównaj macierz L uzyskaną w ten sposób z macierzą uzyskaną przez **chol**() zastosowaną do tej samej macierzy.

Rozdział 10

LZNK. Rozkład QR. Metoda Householdera

W tym rozdziale zajmiemy się liniowym zadaniem najmniejszych kwadratów (LZNK).

Dla danej macierzy A wymiaru $M \times N$ i wektora b wymiaru M chcemy znaleźć wektor x wymiaru N taki, że

$$\|Ax - b\|_2 = \min_y \|Ay - b\|_2.$$

Jeśli A jest macierzą kolumnami regularną (rząd A jest maksymalny równy N), to zadanie to ma jednoznaczne rozwiązanie i nazywamy je regularnym LZNK (RLZNK).

Podstawowym algorytmem służącym jego rozwiązaniu jest metoda Houseloldera, czyli znalezienia rozkładu QR macierzy A, gdzie Q to macierz ortogonalna - iloczyn macierzy Householdera, a R to macierz górnotrójkątna.

W tym rozdziale przetestujemy podstawowy operator octave'a służacy do rozwiązywania dowolnego LZNK, tzn. operator \. Zauważmy, że jeśli $M = N$ i A jest macierzą kolumnami regularną, to A jest nieosobliwa i to regularne LZNK jest równoważne rozwiązaniu układu równań liniowych $Ax = b$.

Przetestujemy w zadaniach funkcję octave'a **qr**() służącą znajdowaniu rozkładu QR macierzy, kilka własności macierzy (przekształceń) Householdera i rozwiążemy kilka konkretnych LZNK.

Zadanie 1 Operator octave'a \ służący m.in. do rozwiązywaniu układów równań liniowych i LZNK w octavie.

- Przetestuj operator octave'a \ rozwiązując RLZNK dla macierzy A ze znanym konkretnym rozwiązaniem x:

$$A = [1, 1; 1, -1; 1, 3], \qquad x = [1; 2]$$

przyjmując, że $f = Ax$.

Czy y=A\f jest rozwiązaniem tego układu?

Policz normy residualną $\|Ay - f\|_2$ i normę błędu $\|x - y\|_2$.

- Przetestuj ten operator dla nieregularnego LZNK dla macierzy A^T z $x = [1; 1; 1]$ i $f = Ax$.

  Czy y=A'\f jest rozwiązaniem tego układu?

  Policz normy residualną $\|A^T y - f\|_2$ i normę błędu $\|x - y\|_2$.

Zadanie 2 Rozkład QR w octave. Funkcje **qr**().

(a) Zapoznaj się z pomocą do funkcji: **qr**().

(b) Dla macierzy $A = [1, 1; 1, -1; 1, 3]$ znajdź jej rozkład $A = QR$ z pomocą funkcji **qr**().

(c) Sprawdź, czy uzyskana Q jest ortogonalna - policz normy Frobeniusa $QQ^T - I$ i $Q^TQ - I$.

(d) Sprawdź ten rozkład licząc normy macierzowe: normę drugą i normę Frobeniusa błędu $A - QR$.

(e) Zastosuj ten rozkład do znalezienia rozwiązania LZNK $Ax = f$ ze znanym rozwiązaniem, np. $x = [1; 0]$ i $f = [1; 1; 1]$.

(f) Policz normę drugą wektorową pomiędzy x, a wynikiem algorytmu w, polegającym na zastosowaniu odpowiedniego rozkładu oraz takie same normy residualne, tzn. normy drugie $Aw - f$ oraz $Rw - Q^T f$.

Zadanie 3 Układ równań normalnych, a rozkład QR.

Rozpatrzmy macierz $A_{2n,k}$ - pod-macierz wymiaru $2n \times k$ macierzy Vandermonde'a $A_{2n,2n}$ dla $2n$ węzłów równoodległych na $[0, 1]$.

LZNK z $A_{2n,k}$ z wektorem prawej strony f równym pierwszej kolumnie tej macierzy (rozwiązanie to pierwszy wersor) rozwiąż trzema sposobami:

(a) używając operator \,

(b) używając rozkład QR uzyskanym funkcją **qr**(),

(c) poprzez rozwiązanie układu równań normalnych:

$$Bx = g$$

dla

$$B = A_{2n,k}^T A_{2n,k}, \qquad g = A_{2n,k}^T f,$$

tzn. tworzymy macierz układu równań normalnych B, wektor prawej strony g układu równań normalnych, a następnie rozwiązujemy układ równań normalnych operatorem \ .

Macierz Vandermonde'a można w octave'ie utworzyć za pomocą funkcji **vander**().

Przeprowadź testy dla $N = 10, 20, 40, 80$ i $k = 2, 4, n$. Porównaj

- czas obliczeń
- błąd - $\|x - y\|_2$
- błąd residualny $\|Ax - f\|_2$

dla x rozwiązania dokładnego LZNK, f wektora prawej strony LZNK, y przybliżenia rozwiązania uzyskanego daną metodą.

Zadanie 4 Rozkład QR a operator \ przy rozwiązywaniu układów równań liniowych,

Rozpatrzmy macierz $A_{n,n}$ Vandermonde'a dla n węzłów równoodległych na $[0, 1]$.

Układ równań liniowych z wektorem prawej strony równym pierwszej kolumnie tej macierzy (rozwiązanie to pierwszy wersor) rozwiąż dwoma sposobami:

(a) operatorem \,

(b) rozkładem QR uzyskanym funkcją **qr**(),

Przetestuj dla $N = 10, 20, 40, 80$. Porównaj

- czas obliczeń,
- błąd $\|x - y\|_2$,
- błąd rezydualny: $\|Ax - f\|_2$,

dla x rozwiązania dokładnego tego układu równań, f wektora prawej strony układu i y przybliżenia rozwiązania uzyskanego daną metodą.

Zadanie 5 Krzywa najlepiej pasująca do danych punktów.

Zastosuj octave'a do znalezienia współczynników a, b krzywej najlepiej pasującej do zadanych punktów: (x_k, y_k), tzn. znajdź takie a, b, że

$$\sum_k |ax_k^2 + by_k^2 - 1|^2 = \min_{c,d} \sum_k |cx_k^2 + dy_k^2 - 1|^2.$$

Za (x_k, y_k) przyjmiemy zaburzone punkty z danej elipsy

$$y_k = \sqrt{1 - 4 * x_k^2} + z_k,$$

gdzie z_k to zaburzenie wylosowane z $[0, 10^{-2}]$ a $x_k = 1/k$ lub $x_k = -1 + h * k$ dla $h = 2/N$ $k = 1, ..., N$.

Czy obliczone a i b jest bliskie 4 i 1?

W jednym oknie zaznacz punkty (x_k, y_k) plusami oraz narysuj fragmenty wykresów obu elips: pierwszej - dla $a = 4, b = 1$ i drugiej elipsy - dopasowanej do zaburzonych punktów.

Powtórz obliczenia dla różnych zaburzeń z_k.

Zadanie 6 Zaprogramuj funkcję octave'a

function y=H(x,w,nw)

która dla danych wektorów $\vec{x}$ i $\vec{w}$ tego samego wymiaru N i skalaru $nw = \|w\|_2$ zwróci wektor $y = H_w\vec{x}$ dla

$$H_w = I - 2 * \frac{1}{nw}\vec{w}\vec{w}^T$$

czyli przekształcenia (macierzy) Householdera.

Skalar może być parametrem opcjonalnym. Jeśli funkcja będzie wywołana z dwoma tylko parametrami, to normę w można obliczyć w tej funkcji.

Przetestuj tę funkcję dla losowych wektorów $\vec{x}$ i $\vec{w}$ i sprawdź, czy

$$\|H_w\vec{x}\|_2 = \|\vec{x}\|_2, \qquad H_w(H_w x) = x.$$

Zadanie 7 Napisz ogólniejszą wersję funkcji z poprzedniego zadania, tzn.: funkcję:

function Y=Hm(X,w,nw) ,

gdzie X macierz $N \times M$ i wtedy zwracany wynik to macierz $Y = H_w * X$. Pozostałe dwa parametry funkcji pozostaną bez zmian.

Czy można zaimplementować taką funkcję bez użycia pętli?

Sprawdź, wykorzystując tę funkcję, czy mnożenie przez macierz Householder nie zmienia norm macierzowych drugiej i Frobeniusa, tzn. czy:

$$\|A\|_2 = \|H_w * A\|_2 = \|A * H_w\|_2$$

i

$$\|A\|_F = \|H_w * A\|_F = \|A * H_w\|_F$$

dla losowej macierzy A i H_w macierzy Householdera dla losowego wektora $w \neq 0$.

Zadanie 8 Znajdź wektor Householdera $\vec{w}$ taki, że odpowiednie przekształcenie Householdera przeprowadza dany wektor $\vec{u} \neq 0$ na kierunek drugiego danego niezerowego wektora $l * \vec{v} \neq 0$ dla $l = \frac{\|v\|_2}{\|u\|_2}$.

Przetestuj dla dowolnych dwóch różnych wektorów o tej samej długości, czy rzeczywiście $H_w\vec{u} = \vec{v}$.

Zadanie 9 Zastosuj metodę Householdera do rozwiązania zadania znalezienia prostej $y = ax+b$ najlepiej przybliżającej N punktów $(x_k, y_k) = (k, 1 + 2 * k + \epsilon_k)$ dla $k = 1, \ldots, N$ gdzie $(\epsilon_1, \ldots, \epsilon_N)$ to losowy wektor za zakresu $[-\epsilon, \epsilon]$, tzn.:

$$\sum_k |ax_k + b - y_k|^2 = \min_{c,d} \sum_k |cx_k + d - y_k|^2.$$

Należy testować dla wartości $\epsilon = [1, 10^{-1}, 10^{-2}, 10^{-3}]$.

Funkcja **rand**(n) generuje wektor losowy o rozkładzie jednostajnym na $[0, 1]$ w octavie.

Porównaj z wynikami otrzymanymi za pomocą standardowej funkcji octave'a, tzn. $\backslash$, oraz przy wykorzystaniu funkcji octave'a **qr**(A).

Zadanie 10 Zaprogramuj metodę Householdera rozwiązywania układu równań liniowych $A\vec{x} = \vec{b}$ dla A macierzy trójdiagonalnej $N \times N$, tzn. napisz funkcję octave'a:

function [x]=hous3diag(a,b,c,f,N)

Parametry funkcji:

- a, b, c przekątna, pod-przekątna i nad-przekątna macierzy A,
- f - wektor prawej strony,
- N - wymiar zadania - długość przekątnej a.

Funkcja zwraca x rozwiązanie $Ax = f$.

Przetestuj działanie funkcji analogicznie do zadania 4 dla macierzy trójdiagonalnej o stałych diagonalach, np. takiej, że elementy na głównej diagonali są równe dwa, a elementy na pod- i nad-diagonalach są równe minus jeden dla $N = 10^p$ z $p = 1, 2, \ldots, 9.$. Za wektor prawej strony f możemy przyjąć pierwszą kolumnę macierzy A.

Rozdział 11

Numeryczne zadanie własne

W tym rozdziale zajmiemy się symetrycznym zadaniem własnym, tzn. zadaniem znajdowania wartości i/lub wektorów własnych dla macierzy symetrycznej $A = A^T$. W zadaniach zbadamy, jak działa podstawowa funkcja octave'a rozwiązująca zadanie własne, tzn. funkcja octave'a **eig**() oraz zbadamy zbieżność różnych wersji metody potęgowej.

Metoda potęgowa jest zdefiniowana następująco: dla dowolnego wektora $\hat{x}_0 = x_0$ o normie drugiej równej jeden (np. losowego), definiujemy ciąg iteracji metody potęgowej następująco:

$$\begin{array}{rcll} \hat{x}_k & = & Ax_{k-1} & \text{(iteracja)} \\ x_k & = & \frac{\hat{x}_k}{\|\hat{x}_k\|_2} & \text{(normowanie)} \qquad k > 0 \\ r_k & = & x_k^T A x_k & \text{(iloraz Rayleigha)} \end{array} \tag{11.1}$$

Zgodnie z teorią, o ile x_0 nie jest ortogonalny do wektora własnego (podprzestrzeni własnej) dla największej wartości własnej, to x_{2k} zbiegnie do tego wektora własnego, a r_k - iloraz Rayleigha - do tej wartości własnej.

Zadanie 1 Zapoznaj się z pomocą do funkcji octave'a **eig**().

Za jej pomocą oblicz wartości własne macierzy

$$\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}.$$

Sprawdź, czy dla λ wartości własnej A prawdą jest, że wyznacznik $A - \lambda * I$ wynosi zero.

Wyznacz macierz V taką, że jej kolumny to wektory własne A.

Przetestuj w normie macierzowej Frobeniusa, czy $\|AV - \Lambda V\|_2$ wynosi zero. Tutaj Λ - to macierz diagonalna z wartościami własnymi A.

Zadanie 2 Zastosuj metodę potęgową (11.1) do zadania własnego z macierzą A z poprzedniego zadania. Za warunek stopu przyjmiemy, że iloraz Rayleigha r_k praktycznie przestanie się zmieniać, co może sugerować, że jest bliski granicy, czyli odpowiedniej wartości własnej, tzn. jeśli

$$|r_{k+1} - r_k| < tol$$

to zatrzymujemy iterację. Przyjmijmy, że $tol = 10^{-12}$.

Przetestuj, czy otrzymane r_k rzeczywiście jest dobrym przybliżeniem największej co do modułu wartości własnej A - czyli dwa, a x_k zbiegło do wektora własnego dla tej wartości własnej o normie drugiej równej jeden, tzn. do $v_1 = (\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})^T$, lub $-v_1$.

Przetestuj algorytm biorąc za x_0:

(a) wektory losowe - otrzymane za pomocą funkcji losowych **rand**() lub **randn**().

(b) wektor własny dla drugiej wartości własnej, tzn. dla minus jeden, czyli unormowany wektor $v_2 = (-2, 1, 1)^T$

(c) wektor prawie ortogonalny do $v_1 = (\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})^T$, np. wektor otrzymany przez zaburzenie ostatniej składowej wektora v_2: $(-2, 1, 1.01)^T$.

Czy we wszystkich tych przypadkach metoda zbiegnie do v_1 lub $-v_1$?

Czy wybór x_0 wpływa na ilość iteracji metody potęgowej?

Zadanie 3 Napisz funkcję octave'a z implementacją metody potęgowej (11.1), tzn. napisz m-plik z funkcją:

```
function [x,r,it]=power(A, tol, x0)
```

która za parametry ma

- A - macierz wymiary $N \times N$, której wektora i wartości własnej szukamy
- *tol* - warunek stopu, domyślnie przyjmujący wartość 10^{-12},
- x_0 -wektor startowy niezerowy, domyślnie losowy,

i która zwraca

- przybliżenie - unormowanego w normie drugiej - wektora własnego x dla największej wartości własnej co do modułu macierzy A,

- r - przybliżenie tej wartości własnej,
- it - ilość iteracji metody potęgowej.

Warunek stopu przyjmiemy jak w poprzednim zadaniu, tzn. że iloraz Rayleigha r_k przestanie się zmieniać. Jeśli

$$|r_{k+1} - r_k| < tol$$

to zatrzymujemy iterację.

Zadanie 4 Utwórz macierz symetryczną (ale nie diagonalną) $N \times N$ o wartościach własnych $1, \ldots, N$. Sprawdź dla $N = 10, 100, 1000$ czy funkcja octave'a **eig**() znajdzie wartości własne tej macierzy.

Zadanie 5 Utwórz macierze symetryczne A_a (niediagonalne) wymiaru 11×11 o wartościach własnych $1, \ldots, 10, a$ dla $a = 11, 101, 1001$ i ze znanymi wektorami własnymi.

Sprawdź, ile iteracji będzie potrzebowała metoda potęgowa dla losowego wektora startowego do znalezienia największej wartości własnej.

Czy potwierdzają się wyniki teoretyczne, że szybkość zbieżności zależy od stosunku największej co do modułu wartości własnej do drugiej co do modułu wartości własnej, czyli w tym zadaniu $a/10$?

Zadanie 6 Dla macierzy z zadania 5 dla $a = 11, 101$ ze znanym wektorem własnym v_1 dla wartości własnej a, przetestuj zbieżność x_k obliczanego przy pomocy metody potęgowej do v_1. W szczególności sprawdź, czy

$$\|v_1 - x_k\|_2 = O(|\lambda_2/\lambda_1|^k)$$

dla $\lambda_2 = 10$ i $\lambda_1 = a$.

Zadanie 7 Dla macierzy z zadania 5 dla $a = 11, 101$ dla wartości własnej a, przetestuj zbieżność r_k ilorazu Rayleigha obliczanego przy pomocy metody potęgowej do λ_1. W szczególności sprawdź, czy

$$|\lambda_1 - r_k| = O(|\lambda_2/\lambda_1|^{2k})$$

dla $\lambda_2 = 10$ i $\lambda_1 = a$.

Zadanie 8 Dla macierzy z zadania 5 dla $a = 11, 101$ ze znanym wektorem własnym v_1 dla wartości własnej 11 przetestuj zbieżność metody potęgowej dla różnych wektorów startowych:

- dla losowego wektora x_0 ortogonalnego do v_1 (jak go można utworzyć?)
- dla wektora prawie ortogonalnego - za x_0 bierzemy $w+10^{-6}*z$, gdzie $w^T v_1 = 0$ i z to wektor losowy o normie drugiej równej jeden.
- dla dowolnego losowego wektora x_0

Proszę powtórzyć testy kilka razy dla każdego podpunktu - czy zdarzyło się, że wyniki były istotnie różne w zależności od wylosowanego wektora?

Zadanie 9 Zaimplementuj funkcję analogiczną do tej z zadania 3 z implementacją odwrotnej metody potęgowej, tzn. metody potęgowej zastosowanej do macierzy odwrotnej A^{-1} zamiast A:

```
function [x,r,it]=invpower(A, tol, x0)
```

Wszystkie parametry i zwracane wartości będą takie same jak w funkcji power() z zadania 3.

Zadanie 10 Utwórz macierz z zadania 5 dla $a = 11, 101, 1001$. Sprawdź, ile iteracji będzie potrzebowała odwrotna metoda potęgowa dla losowego wektora startowego do znalezienia przybliżenia najmniejszej wartości własnej przy ustalonym warunku stopu $|r_k - r_{k+1}| \leq 10^{-12}$. Tutaj r_k - to odpowiedni iloraz Rayleigha.

Zadanie 11 Zaimplementuj funkcję analogiczną do tej z zadania 3 z implementacją odwrotnej metody potęgowej z przesunięciem (*shift*), tzn. metody potęgowej zastosowanej do macierzy $(A - b * I)^{-1}$ dla parametru b:

```
function [x,r,it]=invpowwiel(A, b, tol, x0)
```

Tutaj b będzie dodatkowym parametrem tej funkcji, a pozostałe parametry i zwracane wartości pozostają takie same jak w funkcji z zadania 3.

Zadanie 12 Utwórz macierz z zadania 5 dla $a = 11, 101, 1001$.

Sprawdź, ile iteracji będzie potrzebowała odwrotna metoda potęgowa z przesunięciem dla losowego wektora startowego i różnych wartości parametru $b = 1.3, 1.5, 1.8, 2.1, 2.01$ do znalezienia różnych wartości własnych macierzy A_a.

Zadanie 13 Zaimplementuj funkcję analogiczną do tej z zadania 3 z implementacją metody ilorazów Rayleigha:

```
function [x,r,it]=raylegh(A, b,tol , x0)
```

Metoda polega na tym, że dla danego wektora startowego x_0 o drugiej normie równej jeden, obliczamy iloraz Rayleigha $r_0 = x_0^T A x_0$ i iterujemy:

$$\begin{array}{rcll} x_k & = & \frac{(A-r_{k-1}I)^{-1}x_{k-1}}{\|(A-r_{k-1}I)^{-1}x_{k-1}\|_2} & k>0 \\ r_k & = & x_k^T A x_k & \end{array} \qquad (11.2)$$

Oczywiście w implementacji proszę nie obliczać macierzy odwrotnej do $A - r_k * I$, tylko rozwiązywać układ równań liniowych z tą macierzą.

Wszystkie parametry i zwracane wartości będą takie same jak w funkcji power() z zadania 3.

Zadanie 14 Przetestuj metodę ilorazów Rayleigha na przykładzie macierzy symetrycznej z zadania 5 dla $a = 11, 101, 1001$ biorąc losowy wektor startowy. W kolejnych testach proszę wziąć wektory startowe bliskie wektorom własnym dla wartości własnych 2 i 7.

Zadanie 15 Zaimplementuj funkcję octave'a sprowadzającą macierz symetryczną A wymiaru $N \times N$ do macierzy podobnej trójdiagonalnej: $Q^T B Q = A$ lub $B = Q A Q^T$ za pomocą odpowiednich przekształceń Householdera:

```
function [B,Q]=sym2tridiag(A, typ)
```

która za parametry ma

- A - symetryczną macierz wymiary $N \times N$
- *tol* - typ zwracanej macierzy Q, typ=0 oznacza, że Q jest zwracana jako macierz $N \times N$ - wymnożone macierze Houseloldera (domyślna opcja), typ=1 oznacza, że w macierzy Q zwracamy kolumny odpowiednich macierzy Householdera

i która zwraca

- B - trójdiagonalną macierz symetryczną podobną do A w formacie rzadkim octave'a (**sparse**),

- Q macierz będącą iloczynem przekształceń Householdera lub macierz, w której kolumnach są odpowiednie wektory Householdera w zależności od wartości parametru typ.

Zadanie 16 Przetestuj funkcję z poprzedniego zadania dla jakiejś macierzy symetrycznej A (ani diagonalnej, ani trójdiagonalnej, ani trójkątnej) o znanych wartościach własnych, np. dla macierzy z zadania 5 dla $a = 11, 101, 1001$.

Policz normę Frobeniusa $\|A - Q^T BQ\|_F$ i $\|QAQ^T - B\|_F$.

Sprawdź, czy funkcja octave'a **eig**() dla obu macierzy zwróci te same wartości własne, i czy odpowiednie wektory własne spełniają:

$$Qv_k = w_k, \qquad Q^T w_k = v_k,$$

gdzie w_k to wektor własny macierzy B dla wartości własnej λ_k, a v_k - to wektor własny dla tej samej wartości własnej λ_k dla macierzy A.

Rozdział 12

Algorytm FFT

W tym rozdziale przedstawimy zadania laboratoryjne, w których przetestujemy dyskretną transformatę Fouriera (DFT) oraz szybki algorytm jej obliczania, czyli algorytm szybkiej trasformaty Fouriera; FFT (ang. Fast Fourier Transform)

W opisie matematycznym przyjmujemy, że mamy do czynienia z wektorami okresowymi o okresie długości N indeksowanymi od zera i $x_k = x_{k+N}$ dla k ujemnych.

Przez $\mathcal{F}_N x$ oznaczymy wynik działania operatora DFT na wektorze x, a przez $\mathcal{F}_N^{-1} x$ oznaczymy wynik działania operatora odwrotnego do DFT na wektorze x.

Zadanie 1 Funkcja octave'a z algorytmem FFT **fft** (), która oblicza wartość DFT i funkcja octave'a z algorytmem odwrotnym FFT **ifft** (), która oblicza wartość transformaty odwrotnej do DFT.

Sprawdź, czy rzeczywiście operacje wykonywane przez **fft** () i **ifft** () są do siebie odwrotne, tzn. dla różnych losowych wektorów x o długości $N = 4, 8, 16, 32$ policz $y = \mathcal{F}_N x$ za pomocą **fft** (), a następnie $z = \mathcal{F}_N^{-1} y$ za pomocą **ifft** ().

Policz normy drugie różnicy $z - x$.

Zadanie 2 Skalowanie w funkcjach octave'a **fft** () i **ifft** ().

W literaturze zdarza się definicja DFT bez $1/N$ przed macierzą. Wtedy macierz odwrotną do DFT należy przemnożyć przez N albo obie przez $1/\sqrt{N}$.

Sprawdź, jaka jest stała przed DFT obliczanym przez funkcję

fft (), tzn. wyznacz C_N takie, że

$$(\mathcal{F}_N x)_j = C_N \sum_{k=0}^{N-1} x_k \exp(-2\pi * j * k/N)$$

dla $\mathcal{F}_N x$ obliczanego przez tę funkcję octave'a .

Zadanie 3 Sprawdź, czy DFT obliczona przy pomocy funkcji **fft** () spełnia

$$\|\mathcal{F}_N x\|_2 = C_N \|x\|_2$$

ze stałą C_N dla stu losowych wektorów x, oraz czy analogicznie odwrotna DFT obliczona **ifft** () spełnia:

$$\|\mathcal{F}_N^{-1} y\|_2 = C_N^{-1} \|x\|_2.$$

Zadanie 4 Algorytm szybkiego mnożenia wielomianów.

Napisz funkcję octave'a

function w=polymult(p,q)

szybko mnożącą dwa wielomiany stopnia nie większego od N, dla których znamy ich współczynniki w bazie potęgowej. Funkcja powinna korzystać z DFT i odwrotnej DFT, czyli dyskretnej transformaty Fouriera i odwrotnej dyskretnej transformaty Fouriera.

Parametry funkcji powinny być:

- wektor $p = (p_k)_k$ ze współczynnikami wielomianu $P(x) = \sum_k p_k x^k$,
- wektor $q = (q_k)_k$ ze współczynnikami wielomianu $Q(x) = \sum_k q_k x^k$.

Funkcja powinna zwrócić współczynniki wielomianu $W(x) = P(x) * Q(x) = \sum_k w_k x^k$.

Sprawdź działanie funkcji dla wielomianów $P(x) = 1+x$ i $Q(x) = 2+x$, czyli $(P * Q)(x) = 2 + 3x + x^2$, a potem dla wielomianów dużych stopni, czyli np. dla $Q(x) = x^{50}$ i $P(X) = x^{50} + 1$ funkcja powinna zwrócić współczynniki wielomianu $(P * Q)(x) = x^{100} + x^{50}$.

Porównaj czas obliczeń tej funkcji z wynikiem funkcji octave'a **conv**() dla wielomianów różnych stopni z losowymi współczynnikami.

Zadanie 5 DFT a dyskretny splot dwóch wektorów.

Napisz dwie funkcje obliczające splot dwóch wektorów $z = x \star y$

- wprost z definicji, tzn. $z_k = \sum_{j=0}^{N-1} x_k y_{k-j}$
- z wykorzystaniem DFT - tu musimy skorzystać z tego, że

$$(\mathcal{F}_N(x \star y))_k = \alpha_N (\mathcal{F}_N x)_k * (\mathcal{F}_N y)_k \qquad k = 0, \ldots, N-1$$

dla pewnej stałej α_N. Trzeba tę stałą wyznaczyć teoretycznie albo eksperymentalnie.

Przetestuj obie funkcje dla losowych wektorów i różnych $N = 4, 8, 64, 1024$. Porównaj z wynikami funkcji octave'a **conv**(). Należy zapoznać się z pomocą do tej funkcji.

Zadanie 6 Filtry a DFT.

W niektórych zastosowaniach współczynniki wektora x mogą odpowiadać próbkom sygnału (np. dźwięku). Stosuje się wtedy filtry w postaci splotu x z zadanym wektorem F pełniącym rolę filtru: $x \star F$.

Rozważmy filtr postaci $F = \frac{1}{4} * (1, 1, 1, 1, 0, \ldots, 0)^T$ (pierwsze cztery współrzędne są równe $\frac{1}{4}$, pozostałe - zero).

Zastosuj ten filtr do wektora imitującego próbki sygnału z szumem: $z = x + y$, gdzie $x_k = \sin(k * h)$ dla $h = 2 * \pi / N$ - imituje sygnał bez szumów, a y jest wektorem losowym o wartościach w $[-0.1, 0.1]$ imitującym losowe zaburzenie, czyli tzw. szumy. Przetestuj filtr dla dużych N, np. $N = 1024$ lub 2048, tzn.:

- Narysuj wykres z przed zastosowaniem filtra i po zastosowaniu, tzn. wykresy z i $z \star F$. Można narysować mały fragment wykresu, np dla $100 \leq k \leq 120$. Jaki efekt optyczny widać na wykresach?
- Policz maksimum z modułu różnicy kolejnych elementów obu wektorów, tzn. $\max_{k>0} |z_k - z_{k-1}|$ i $\max_{k>0} |(z \star F)_k - (z \star F)_{k-1}|$. Im ta wartość dla danego wektora z próbkami sygnału jest mniejsza, tym sygnał jest gładszy, czyli możemy uznać go za sygnał z mniejszą ilością szumów.

Projekty zaliczeniowe

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.

Bibliografia

[1] Åke Björck i Germund Dahlquist. *Metody numeryczne*. Państwowe Wydawnictwo Naukowe (PWN), Warszawa, 1983. Przetłumaczone ze szwedzkiego przez Stefana Paszkowskiego.

[2] James W. Demmel. *Applied numerical linear algebra*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.

[3] Maksymilian Dryja, Janina Jankowska, i Michał Jankowski. *Metody numeryczne*, tom 2. Wydawnictwo Naukowo Techniczne (WNT), Warszawa, 1982.

[4] John W. Eaton. *GNU Octave Manual*. Network Theory Limited, 2002.

[5] Gene Golub i James M. Ortega. *Scientific computing*. Academic Press Inc., Boston, MA, 1993. An introduction with parallel computing.

[6] Gene H. Golub i James M. Ortega. *Scientific computing and differential equations*. Academic Press Inc., Boston, MA, 1992. An introduction to numerical methods.

[7] Wolfgang Hackbusch. *Iterative solution of large sparse systems of equations*, tom 95, *Applied Mathematical Sciences*. Springer-Verlag, New York, 1994. Translated and revised from the 1991 German original.

[8] Janina Jankowska i Michał Jankowski. *Metody numeryczne*, tom 1. Wydawnictwo Naukowo Techniczne (WNT), Warszawa, 1981.

[9] C. T. Kelley. *Solving nonlinear equations with Newton's method*. Fundamentals of Algorithms. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2003.

[10] A. Kiełbasiński i H. Schwetlick. *Numeryczna algebra liniowa: wprowadzenie do obliczeń zautomatyzowanych*. Wydawnictwo Naukowo Techniczne (WNT), Warszawa, 1992.

[11] David Kincaid i Ward Cheney. *Analiza numeryczna*. Wydawnictwo Naukowo-Techniczne (WNT), 2006.

[12] J. M. Ortega i W. C. Rheinboldt. *Iterative solution of nonlinear equations in several variables*, tom 30, *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2000. Reprint of the 1970 original.

[13] James M. Ortega. *Numerical analysis*, tom 3, *Classics in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 1990. A second course.

[14] Alfio Quarteroni, Riccardo Sacco, i Fausto Saleri. *Numerical mathematics*, tom 37, *Texts in Applied Mathematics*. Springer-Verlag, New York, 2000.

[15] Alfio Quarteroni i Fausto Saleri. *Scientific computing with MATLAB*, tom 2, *Texts in Computational Science and Engineering*. Springer-Verlag, Berlin, 2003.

[16] Anthony Ralston. *Wstęp do analizy numerycznej*. Państwowe Wydawnictwo Naukowe (PWN), Warszawa, 1983.

[17] J. Stoer i R. Bulirsch. *Wstęp do metod numerycznych. Tom drugi.* Państwowe Wydawnictwo Naukowe (PWN), Warszawa, 1980. Przetłumaczone z niemieckiego przez Małgorzatę Mikulską.

[18] Lloyd N. Trefethen i David Bau, III. *Numerical linear algebra.* Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1997.