

Sprawozdanie

Obliczenia naukowe - lista 5

Kamil Król

244949

Opis problemu oraz struktury danych

Celem listy jest rozwiązanie układu równań liniowych: $\mathbf{A}\mathbf{x} = \mathbf{b}$, dla danej macierzy $\mathbf{A} \in \mathbb{R}^{n \times n}$ i wektora prawych stron $\mathbf{b} \in \mathbb{R}^n$, gdzie $n \geq 4$.

Zgodnie z treścią listy macierz \mathbf{A} jest rzadką macierzą (tj. mającą dużo elementów zerowych) blokową o następującej strukturze:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{C}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{B}_2 & \mathbf{A}_2 & \mathbf{C}_2 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_3 & \mathbf{A}_3 & \mathbf{C}_3 & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{B}_{v-2} & \mathbf{A}_{v-2} & \mathbf{C}_{v-2} & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{B}_{v-1} & \mathbf{A}_{v-1} & \mathbf{C}_{v-1} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_v & \mathbf{A}_v \end{pmatrix}, \quad (1)$$

$v = n/\ell$, zakładając że n jest podzielne przez ℓ , gdzie $\ell \geq 2$ jest rozmiarem wszystkich kwadratowych macierzy wewnętrznych (bloków) $\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i, \mathbf{0}$.

Macierze oznaczone przez $\mathbf{0}$ są macierzami zerowymi tzn. $\mathbf{0} \in \mathbb{R}^{\ell \times \ell}$. Macierze $\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i$, są następującej postaci:

- (i) $\mathbf{A}_i \in \mathbb{R}^{\ell \times \ell}$, $i = 1, \dots, v$ – macierze gęste,
- (ii) $\mathbf{B}_i \in \mathbb{R}^{\ell \times \ell}$, $i = 2, \dots, v$ – macierze mające tylko dwie ostatnie kolumny niezerowe:

$$\mathbf{B}_i = \begin{pmatrix} 0 & \cdots & 0 & b_{1\ell-1}^i & b_{1\ell}^i \\ 0 & \cdots & 0 & b_{2\ell-1}^i & b_{2\ell}^i \\ \vdots & & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & b_{\ell\ell-1}^i & b_{\ell\ell}^i \end{pmatrix}, \quad (2)$$

- (iii) $\mathbf{C}_i \in \mathbb{R}^{\ell \times \ell}$, $i = 1, \dots, v-1$ – macierze diagonalne:

$$\mathbf{C}_i = \begin{pmatrix} c_1^i & 0 & 0 & \cdots & 0 \\ 0 & c_2^i & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & c_{\ell-1}^i & 0 \\ 0 & \cdots & 0 & 0 & c_\ell^i \end{pmatrix}. \quad (3)$$

Dodatkową informacją o danych jest ich duży rozmiar. W konsekwencji wyklucza to pamiętanie macierzy \mathbf{A} jako tablicę o wymiarach $n \times n$ oraz użycie standardowych (bibliotecznych) algorytmów dla macierzy gęstych tj. takich, gdzie nie zakłada się dużej liczby elementów zerowych. Zatem dodatkowym wymaganiem co do zadania jest specjalna struktura pamiętająca efektywnie macierz \mathbf{A} . Taka która pamięta tylko elementy niezerowe macierzy \mathbf{A} . W tym celu w moich programach użyłem biblioteki *SparseArrays*, która pamięta tylko elementy niezerowe. W naszych rozważaniach założymy, że możemy uzyskać dostęp do elementów macierzy w czasie stałym – $O(1)$. W rzeczywistości tak nie jest.

Ponadto konieczna jest modyfikacja standardowych algorytmów tak, aby uwzględniały one specyficzną postać macierzy \mathbf{A} , tj. jej rzadkość, regularność występowania elementów zerowych i niezerowych.

Zaimplementowane przeze mnie algorytmy (napisane w języku Julia i opisane poniżej) działają na macierzy transponowanej, ze względu na optymalizację tzn. dostęp do elementów po kolumnach jest szybszy niż po wierszach. W dalszych rozważaniach pominiemy ten fakt gdyż transpozycja nie zmienia wewnętrznej struktury macierzy \mathbf{A} oraz nie zmienia ogólnej idei algorytmu. Odnośnik do dokumentacji: `juliaLang - SparseArrays`.

Zadanie 1

Celem tego zadania było napisanie funkcji obliczającej wartość wielomianu interpolacyjnego stopnia n w postaci Newtona $N_n(x)$ w punkcie $x = t$ za pomocą uogólnionego algorytmu Hornera, która działa w czasie liniowym ($O(n)$).

Dane:

- `x` – wektor długości $n + 1$ zawierający węzły x_0, \dots, x_n ,
- `fx` – wektor długości $n + 1$ zawierający ilorazy różnicowe,
- `t` – punkt, w którym należy obliczyć wartość wielomianu.

Oczekiwany wynik:

- `nt` – wartość wielomianu w punkcie t

Opis:

Wzór na wielomian interpolacyjny Newtona N_n pokazuje w jaki sposób zależy on od funkcji f . Wzór ten można przedstawić używając ilorazów różnicowych:

$$N_n(x) = \sum_{i=0}^n f[x_0, x_1, \dots, x_i] \prod_{j=0}^{i-1} (x - x_j).$$

Z numerycznego punktu widzenia takie przedstawienie wielomianu interpolacyjnego jest bardzo atrakcyjne. Zauważmy, że w sytuacji kiedy chcielibyśmy dodać nowe węzły (x_i, y_i) możemy to zrobić korzystając ze wcześniej policzonych $d_k = f[x_0, x_1, \dots, x_k]$. Kluczowa jest tu własność ilorazów różnicowych mówiąca, że wartość ilorazu nie zależy od kolejności węzłów. Kolejną zaletą takiego zapisu jest to, że wartość tego wielomianu możemy łatwo obliczyć korzystając z uogólnionego algorytmu Hornera. Sposób w jaki można to zrobić przedstawiono poniżej.

$$\begin{aligned} w_n(x) &:= f[x_0, x_1, \dots, x_n] \\ w_k(x) &:= w_{k+1}(x - x_k) + f[x_0, x_1, \dots, x_k] \quad (k = n - 1, n - 2, \dots, 0) \\ N_n(x) &= w_0(x) \end{aligned}$$

Pseudokod algorytmu

Algorytm 1: Obliczanie wartości wielomianu interpolacyjnego w punkcie t .

```
function warNewton( $x, fx, t$ )  
     $n \leftarrow \text{length}(fx)$   
     $nt \leftarrow fx[n]$   
    for  $i \leftarrow n - 1$  downto 1 do  
         $nt \leftarrow fx[i] + (t - x[i]) * nt$   
    return  $nt$ 
```

Zadanie 3

Celem tego zadania było napisanie funkcji obliczającej współczynniki a_0, \dots, a_n postaci naturalnej wielomianu interpolacyjnego dla zadanych współczynników $d_0 = f[x_0], d_1 = f[x_0, x_1], \dots, d_n = f[x_0, \dots, x_n]$ tego wielomianu w postaci Newtona oraz węzłów x_0, \dots, x_n . Ponadto funkcja miała działać w czasie $O(n^2)$.

Dane:

\mathbf{x} – wektor długości $n + 1$ zawierający węzły x_0, \dots, x_n ,

\mathbf{fx} – wektor długości $n + 1$ zawierający ilorazy różnicowe.

Oczekiwany wynik:

\mathbf{a} – wektor długości $n + 1$ zawierający obliczone współczynniki postaci naturalnej.

Opis:

Przypomnijmy, że wartości d_0, d_1, \dots, d_n są współczynnikami wielomianu interpolacyjnego w postaci Newtona. Punktem wyjściowym to wyprowadzenia algorytmu będzie uogólniony algorytm Hornera. Najpierw jednak zapiszmy wielomian interpolacyjny w postaci Newtona:

$$p(x) = d_0 + (x - x_0)(d_1 + (x - x_1)(d_2 + \dots + (x - x_{n-2})(\overbrace{d_{n-1} + (x - x_{n-1})d_n}^{W_{n-1}})) \dots)$$

$\underbrace{\hspace{15em}}_{W_1}$

$\underbrace{\hspace{20em}}_{W_0}$

Idea jest bardzo podobna do wyprowadzania uogólnionego algorytmu Hornera w poprzednim zadaniu. Teraz przyjrzyjmy się zaznaczonym wielomianom W_k . Zauważmy też, że ich stopnie rosną (patrzac od góry do dołu).

$$\begin{aligned} W_n(x) &= d_n \\ W_{n-1}(x) &= d_{n-1} + (x - x_{n-1})W_n \\ &\dots = \dots \\ W_k(x) &= d_k + (x - x_k)W_{k+1} \text{ dla } 0 \leq k < n \\ &\dots = \dots \\ W_0(x) &= p_0(x) \end{aligned}$$

Dodatkowo mamy też, że $\deg(W_k) = \deg(W_{k+1}) + 1$. Obie te obserwacje są kluczowe dla wyprowadzenia algorytmu. Przypomnijmy, że wartości d_0, d_1, \dots, d_n oraz x_0, \dots, x_n są danymi, a więc są znane. Zastanówmy się jak obliczyć współczynniki wielomianu W_{n-1} w postaci naturalnej. Mamy $\deg(W_{n-1}) = \deg(W_n) + 1 = 0 + 1 = 1$, a zatem W_{n-1} możemy ogólnie zapisać jako $W_{n-1}(x) = a_0x^0 + a_1x^1$. Z drugiej strony patrząc na tabelę wyżej możemy go zapisać jako

$$W_{n-1}(x) = d_{n-1} + (x - x_{n-1})W_n = d_{n-1} + (x - x_{n-1})d_n = d_{n-1} + xd_n - x_{n-1}d_n = (d_{n-1} - d_nx_{n-1})x^0 + (d_n)x^1$$

Otrzymaliśmy współczynniki naturalne wielomianu W_{n-1} . Konkretniej mamy, że $a_0 = d_{n-1} - d_nx_{n-1}$ i $a_1 = d_n$. Zróbmy to samo dla W_{n-2} .

$$\begin{aligned} W_{n-2}(x) &= d_{n-2} + (x - x_{n-2})W_{n-1} = d_{n-2} + (x - x_{n-2})(a_0x^0 + a_1x^1) = \\ &= (d_{n-2} - x_{n-2}a_0)x^0 + (a_0 - a_1x_{n-2})x^1 + a_1x^2 \end{aligned}$$

Obliczyliśmy współczynniki W_{n-2} w postaci naturalnej. Jeśli ten wielomian zapiszemy jako $W_{n-2} = b_0x^0 + b_1x^1 + b_2x^2$ to współczynniki będą następujące: $b_0 = d_{n-2} - x_{n-2}a_0$, $b_1 = a_0 - a_1x_{n-2}$, $b_2 = a_1$. Widzimy zatem, że współczynniki przy najwyższej potędze wielomianów W_{n-1} i W_{n-2} są sobie równe. Ogólnie mamy, że współczynniki przy najwyższej potędze dla wielomianów W_k i W_{k-1} są sobie równe. Inna kluczowa obserwacja to fakt, że licząc współczynniki naturalne wielomianu W_{n-2} korzystaliśmy tylko z danych i ze współczynników naturalnych wielomianu W_{n-1} . Podobnie obliczając współczynniki wielomianu W_{n-1} korzystaliśmy tylko z danych i współczynników wielomianu W_n . Widzimy zatem, że współczynniki naturalne wielomianu W_k jesteśmy w stanie obliczyć znając współczynniki wielomianu W_{k+1} . Oznacza to, że możemy to zrobić używając jednej tablicy. Ponadto jesteśmy w stanie zrobić to w czasie $O(n)$. W szczególności możemy obliczyć współczynniki wielomianu $W_0 = p_0$ w postaci naturalnej licząc kolejno współczynniki wielomianów $W_n, W_{n-1}, \dots, W_1, W_0$ każdy w czasie $O(n)$ co daje łączny czas $O(n^2)$. Teraz dla podsumowania pseudokod.

Pseudokod algorytmu

Algorytm 2: Obliczanie współczynników naturalnych wielomianu interpolacyjnego.

```

function naturalna( $x, fx$ )
     $n \leftarrow \text{length}(fx) - 1$ 
     $a[n] \leftarrow fx[n]$ 
    for  $i \leftarrow n - 1$  downto 0 do
         $a[i] \leftarrow fx[i] - a[i + 1] * x[i]$ 
        for  $j \leftarrow i + 1$  to  $n - 1$  do
             $a[j] \leftarrow a[j] - a[j + 1] * x[i]$ 
    return  $a$ 

```

Zadanie 4

Celem zadania było napisanie funkcji interpolującej zadaną funkcję $f(x)$ w przedziale $[a, b]$ za pomocą wielomianu interpolacyjnego stopnia n w postaci Newtona, a także rysującej wykresy funkcji f oraz otrzymanego wielomianu interpolacyjnego. W interpolacji funkcji należało użyć węzłów równoodległych.

Dane:

f – zadana funkcja,

a, b – przedział interpolacji,

n – stopień wielomianu interpolacyjnego.

Oczekiwany wynik:

– wykres funkcji f oraz wielomianu interpolacyjnego w przedziale $[a, b]$.

Opis:

Na początku wyznaczyłem węzły interpolacyjne x_1, \dots, x_{n+1} w taki sposób aby odległość między nimi wynosiła $\frac{b-a}{n}$. Następnie obliczyłem wartości funkcji w tych punktach tj. $f(x_1), \dots, f(x_{n+1})$. W celu obliczenia ilorazów różnicowych posłużyłem się funkcją z zadania pierwszego – `ilorazyRoznicowe`. Następnie użyłem funkcji z zadania drugiego tj. `warNewton` do obliczenia wartości wielomianu interpolacyjnego w potrzebnych punktach. W celu uzyskania dokładniejszego wykresu, punkty dla których rysowałem wykres musiałem zagęścić. Zrobiłem to mnożąc dane n razy obrany przeze mnie parametr gęstości równy 40. Dzięki temu uzyskałem dokładniejsze wykresy.

Zadanie 5

Celem zadania było przetestowanie funkcji `rysujNnfx(f,a,b,n)` (z zadania 4) na następujących przykładach:

(a) $f(x) = e^x$, $[a, b] = [0, 1]$, $n \in \{5, 10, 15\}$,

(b) $f(x) = x^2 \sin x$, $[a, b] = [-1, 1]$, $n \in \{5, 10, 15\}$.

Poniżej narysowane wykresy dla obu funkcji. Na przykładach tych funkcji widać, że wybranie równoodległych węzłów dało bardzo dokładne przybliżenia funkcji. Dla żadnego z wykresów nie zaobserwowano rozbieżności. Kolejna rzecz warta zaobserwowania to fakt, że dla wszystkich wartości n funkcje były bardzo dobrze przybliżone.

Wykres funkcji e^x i jej wielomianu interpolacyjnego dla danego stopnia n

Wykres funkcji $x^2 \sin x$ i jej wielomianu interpolacyjnego dla danego stopnia n

Zadanie 6

Celem zadania było przetestowanie funkcji `rysujNnfx(f,a,b,n)` (z zadania 4) na następujących przykładach:

(a) $f(x) = |x|$, $[a, b] = [-1, 1]$, $n \in \{5, 10, 15\}$,

(b) $f(x) = \frac{1}{1+x^2}$, $[a, b] = [-5, 5]$, $n \in \{5, 10, 15\}$.

Wykresy otrzymane za pomocą metody `rysujNnfx(f,a,b,n)` prezentują poniższe wykresy.

Wykres funkcji $|x|$ i jej wielomianu interpolacyjnego dla danego stopnia n

Wykres funkcji $\frac{1}{1+x^2}$ i jej wielomianu interpolacyjnego dla danego stopnia n

Widać, że dla funkcji $|x|$ pojawiają się większe odchylenia niż dla funkcji z zadania poprzedniego. Widać też, że wraz ze zwiększaniem stopnia wielomianu odchylenia/błędy, w szczególności na końcach przedziału, znacznie wzrastają. Dla $n = 20$ te odchylenia są już na tyle duże, że wykres traci na czytelności. Dla drugiej funkcji sytuacja wygląda tak samo – wraz ze zwiększaniem stopnia wielomianu błąd rośnie, a błędy najbardziej rosną na końcach przedziału. Jest to sprzeczne z intuicją, ponieważ zwiększając stopień wielomianu interpolacyjnego oczekujemy lepszego przybliżenia. Zaobserwowane zjawisko nosi nazwę zjawiska Rungego. (Sama funkcja $\frac{1}{1+x^2}$ nazywana jest funkcją Rungego). Polega ono na tym, że dla pewnych funkcji błąd wielomianu interpolacyjnego wyliczonego za pomocą równoodległych węzłów dąży do nieskończoności wraz ze wzrostem stopnia tego wielomianu interpolacyjnego.

Pojawia się pytanie czy można poprawić dokładność wielomianów interpolacyjnych dla takich funkcji. Z pomocą przychodzą nam węzły Czebyszewa będące pierwiastkami wielomianów Czebyszewa pierwszego rodzaju. Ich użycie zwiększa liczbę węzłów w miejscach, które są trudniejsze do przybliżenia, czyli między innymi na końcach przedziału. Skutkuje to otrzymaniem dokładniejszego przybliżenia. Poniżej znajdują się wykresy z wielomianami interpolacyjnymi stworzonymi na podstawie węzłów Czebyszewa.

Wykres funkcji $|x|$ i jej wielomianu interpolacyjnego skonstruowanego przy pomocy węzłów Czebyszewa

Wykres funkcji $\frac{1}{1+x^2}$ i jej wielomianu interpolacyjnego stworzonego przy pomocy węzłów Czebyszewa