Obliczenia naukowe

Sprawozdanie nr 4

Barbara Banaszak
236514

09-12-2018

1 Zadanie 1

1.1 Opis problemu

W zadaniu należy zaimplementować funkcję obliczającą wartości ilorazy różnicowych, bez wykorzystania do tego celu tablicy dwuwymiarowej. Ilorazy różnicowe możemy obliczać stosując następujący wzór rekurencyjny:

1. dla $i = 0 \quad f[x_0] = f(x_0)$
2. dla $i = 1 \quad f[x_0, x_1] = \frac{f(x_1)-f(x_0)}{x_1-x_0}$
3. dla $i = n \quad f[x_0, ..., x_n] = \frac{f(x_1,...,x_1)-f(x_0,...,x_{n-1})}{x_n-x_0}$

Z tej zależności skorzystamy implementując algorytm. Należy zauważyć jednak, że zaimplementowanie powyższej rekursji wprost, wymagałoby umieszczenia wyników w dwuwymiarowej tablicy o rozmiarze nxn, gdzie pierwszy wiersz to wartości ilorazy róznicowych, które ma zwracać funkcja:

$$\begin{bmatrix} f[x_0] & f[x_0, x_1] & \dots & f[x_0, ..., x_n] \\ f[x_1] & f[x_1, x_2] & \dots & ? \\ \vdots & \vdots & \ddots & \vdots \\ f[x_n] & ? & \dots & ? \end{bmatrix}$$

Właśnie takiego zapisu należy uniknąć. Zauważmy, że do policzenia kolejnych ilorazy różnicowych potrzebujemy tylko dwóch "poprzednich"(aktualizujemy tablicę od dołu do góry i od lewej do prawej) oraz że wartości z miejsc wypełnionych ? nie będą nam potrzebne. Licząc w ten sposób wystarczy nam jednowymiarowa tablica, którą w każdym kroku będziemy aktualizować o jedno miejsce mniej, co widać w pokazanym niżej algorytmie.

1.2 Rozwiązanie - funkcja 'ilorazyRoznicowe'

INPUT: x (wektor długości $n+1$, zawierający węzły $x_0, ..., x_n$), f (wektor długości $n+1$, zawierający wartości iterpolowanej funckji w punktach x_i)
OUTPUT: fx (wektor długości $n+1$, zawierający obliczone ilorazy różnicowe: $f[x_1] = f(x_0), ..., f[x_{n+1}] = f(x_0, ..., x_n)$)

$N \leftarrow length(x)$
for $k = 1$ **to** N **do**
  $fx[k] \leftarrow f[k]$
end for
for $k = 2$ **to** N **do**
  for $i = N$ **to** k **do**
    $fx[i] \leftarrow \frac{fx[i]-fx[i-1]}{x[i]-x[i-k+1]}$
  end for
end for
return fx

2 Zadanie 2

2.1 Opis problemu

Zadanie polega na zaimplementowaniu funkcji obliczającej wartość wielomianu interpolacyjnego stopnia n, w postaci Newtona $N_n(x)$, w punkcie t za pomocą uogólnionego algorytmu Hornera, ze złożonością czasową $O(n)$. W zadaniu skorzystamy z poniższej postaci wzoru interpolacyjnego Newtona:

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

oraz z tego, że wartości $N_n(x)$ można obliczać za pomocą wzorów (co jest jednocześnie wykorzystaniem uogólnionego algorytmu Hornera):

$$w_n(x) = f[x_0, ..., x_n]$$

$$w_k(x) = f[x_0, ..., x_k] + (x - x_k)w_{k+1}(x) \quad dla \quad k = (n-1, ..., 0)$$

$$N_n(x) = w_0(x)$$

2.2 Rozwiązanie - funkcja 'warNewton'

INPUT: x (wektor długości $n+1$, zawierający węzły $x_0, ..., x_n$), fx (wektor długości $n+1$, zawierający ilorazy różnicowe), t (punkt, w który należy obliczyć wartość wielomianu)
OUTPUT: nt (wartość wieloianu w punkcie t)

$N \leftarrow length(x)$
$nt \leftarrow fx[N]$
for $k = 1$ **to** N **do**
  $nt \leftarrow fx[k] + (t - x[k]) * nt$
end for
return nt

3 Zadanie 3

3.1 Opis problemu

W zadaniu należy zaimplementować funkcję, która na podstawie współczynników wielomianu interpolacyjnego w postaci Newtona obliczy, w czasie $O(n^2)$, współczynniki $(a_0, ..., a_n)$ jego postaci naturalnej $a_n x^n + a_{n-1}x^{n-1} + ... + a_0$. W zadaniu tym podobnie jak w poprzednim wykorzystany zostanie uogólniony algorytm Hornera. W wielomianie interpolacyjnym współczynnik c_n przy najwyższej potędze jest taki sam jak współczynnik a_n przy najwyższej potędze tego wielomianu w postaci naturalnej. W kolejnych krokach wyliczamy a_i, na podstawie wyliczonych wcześniej wartości stojących przy danej potędze.

3.2 Rozwiązanie - funkcja 'naturalna'

INPUT: x (wektor długości $n+1$, zawierający węzły $x_0, ..., x_n$), fx (wektor długości $n+1$, zawierający ilorazy różnicowe)
OUTPUT: a (wektor długości $n+1$ zawierający współczynniki wielomianu w postaci naturalnej)

$N \leftarrow length(x)$
$a[N] \leftarrow fx[N]$
for $k = N-1$ **to** 1 **do**
  $a[k] \leftarrow fx[k] - a[k+1] * x[k]$
  for $i = k+1$ **to** $N-1$ **do**
    $a[i] \leftarrow a[i] - a[i+1] * x[k]$
  end for
end for
return a

4 Zadanie 4

4.1 Opis problemu

Zadanie polega na zaimplementowaniu funkcji, która zinterpoluje zadaną funkcję f na podanym przedziale oraz poda interpretację graficzną danej funkcji oraz wielomianu ją interpolującego.

4.2 Rozwiązanie - funkcja 'rysujNnfx'

INPUT: f (dana anonimowa funkcja), a, b (krańce przedziału interpolacji), n (stopień wielomianu interpolującego)
OUTPUT: wykresy funkcji oraz wielomianu interpolującego

W celu rozwiązania zadania wyznaczamy n równoodległych punktów należących do funkcji f, które posłużą do stworzenia wielomianu interpolującego f o stopniu n. Tworzymy wielomian interpolujący fx przy użyciu zaimplementowanej wcześniej funkcji **ilorazyRoznicowe()**. Wyznaczamy $n * coeff$ (w celu uzyskania dokładniejszych wykresów funkcji) równoodległych punktów na danym przedziale, które posłużą za dziedzinę funkcji. Poczym wyznaczamy dla danych punktów wartości funkcji f oraz wielomianu interpolującego (z wykorzystaniem funkcji **warNewton()**. Do narysowania wykresów funkcji posłuży biblioteka języka Julia- Plots.

5 Zadanie 5

5.1 Opis problemu

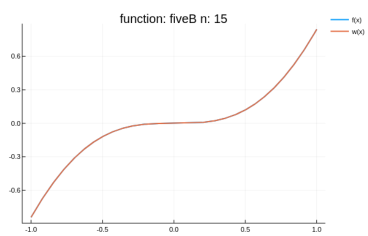
Zadanie polega na przetestowaniu funkcji **rysujNnFx()** (z zadania 4) na przykładach:

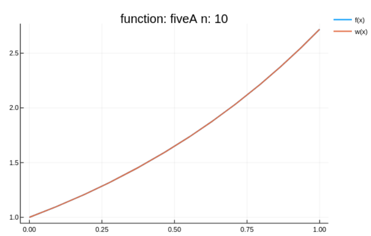
1. $f = e^x, [0, 1], n = 5, 10, 15$
2. $f = x^2 sin(x), [-1, 1], n = 5, 10, 15$

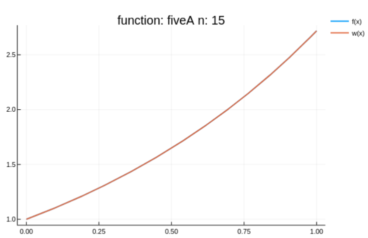
5.2 Rozwiązanie

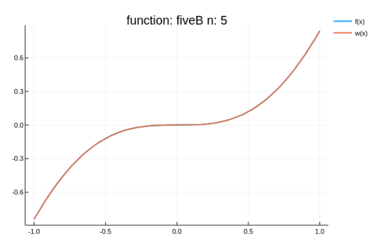
Do rozwiązania zadania użyta została funkcja **rysujNnFx()**, z modułu Interpolation.

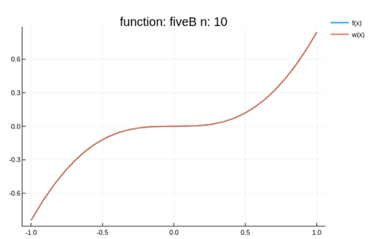
5.3 Wyniki

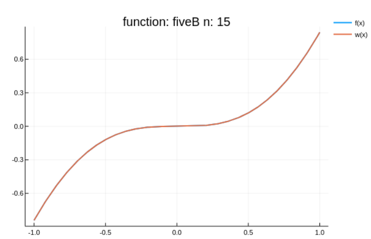












5.4 Wnioski

Dla obu funkcji: e^x(oznaczonej jako fiveA) i $x^2 sin(x)$(oznaczonej jako fiveB) wielomiany interpolacyjne ($w(x)$) są bardzo bliskie interpolowanym funkcjom($f(x)$), patrząć na narysowane wykresy widać, że $w(x)$ właściwie idealnie nakłada się na $f(x)$. Zastosowanie równoodległych węzłów przy interpolacji dało bardzo dobre wyniki i powstałe wielomiany interpolacyjne idealnie odwzorowały przebieg danych funkcji, niezależnie od zadanego stopnia wielomianu.

6 Zadanie 6

6.1 Opis problemu

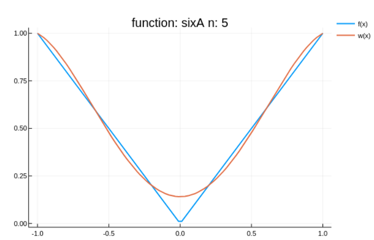
Zadanie polega na przetestowaniu funkcji **rysujNnFx()** (z zadania 4) na przykładach:

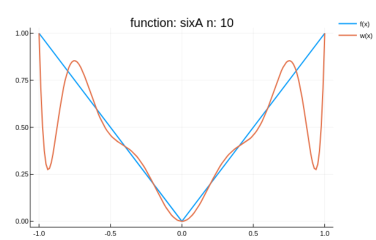
1. $f = |x|, [-, 1], n = 5, 10, 15$
2. $f = \frac{1}{1+x^2}, [-5, 5], n = 5, 10, 15$

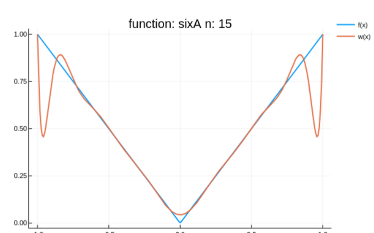
6.2 Rozwiązanie

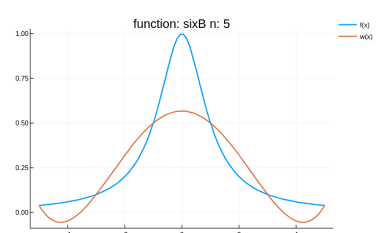
Do rozwiązania zadania użyta została funkcja **rysujNnFx()**, z modułu Interpolation.

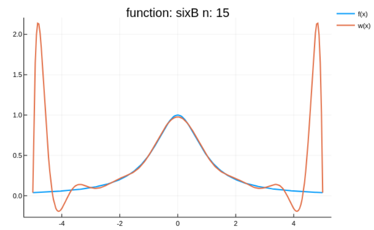
6.3 Wyniki

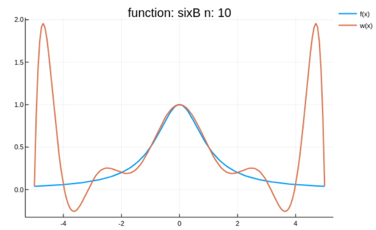












6.4 Wnioski

Łatwo zauważyć, że w przypadku funkcji $|x|$(sixA) oraz $\frac{1}{1+x^2}$(sixB), wielomiany interpolacyjne różnią się od rzeczywistego przebiegu funkcji i dodatkowo im więcej węzłów weźmiemy do utworzenia wielomianu interpolacyjnego tym gorsze wyniki otrzymujemy. W przypadku funkcji $|x|$ jest to spowodowane przede wszytskim tym, że nie jest ona różniczkowalna. Natomiast funkcja $\frac{1}{1+x^2}$ na przedziale $[-5, 5]$ jest przykładem zjawiska Rungiego - pogorszenia jakości interpolacji pomimo zastosowania większej ilości węzłów. Zjawisko to jest związane z pewną osobliwością na osi urojonej w pobliżu przedziału $[-5, 5]$ dla funkcji sixB. Istotne dla wystąpienia zjawiska Rungiego jest także, to że do stworzenia wielomianu interpolacyjnego użyliśmy równoodległych węzłów, a aby tego zjawiska unikać stosuje się interpolację z węzłami gęściej rozmieszczonymi na krańcach przedziałów.