Sprawozdanie

Nie interesuj się

15 kwietnia 2020

Ilorazy Różnicowe

Opis :

Celem jest zaimplementowanie funkcji *'ilorazyRoznicowe'* obliczającej ilorazy różnicowe w oparciu o podane węzły i odpowiadające im wartości funkcji. Dodatkowym wymaganiem był algorytm zoptymalizowany dla pamięci.

Iloraz różnicowy N-tego rzędu oblicza się za pomocą wzoru rekurencyjnego

dla k = 0:

$$f[x_i] = f(x_i), \quad (0 \leq i \leq N)$$

dla k = 1:

$$f[x_i, x_j] = \frac{f(x_j) - f(x_i)}{x_j - x_i}$$

dla k > 1:

$$f[x_i, x_{i+1}, ..., x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, ..., x_{i+k}) - f(x_i, x_{i+1}, ..., x_{i+k-1}]}{x_{i+k} - x_i}, \quad (0 \leq i \leq i+k \leq N)$$

Działanie algorytmu :

Dzięki znajomości węzłów x_i oraz wartości funkcji $f(x_i)$ jesteśmy w stanie stworzyć tablicę ilorazów różnicowych wyższych rzędów:

$$przyjmujemy: \quad c_{ik} = f[x_i, x_{i+1}, ..., x_{i+k}], \quad wtedy:$$

$c_{0,0}$	$c_{0,1}$	$c_{0,2}$	...	$c_{0,k-1}$	$c_{0,k}$
$c_{1,0}$	$c_{1,1}$	$c_{1,2}$	...	$c_{1,k-1}$	
...	...	...	...		
$c_{k-1,0}$	$c_{k-1,1}$				
$c_{k,0}$					

Ogólny wzór na $c_{i,k}$ jest więc następujący:

$$c_{i,k} = \frac{c_{i+1,k-1} - c_{i,k-1}}{x_{i+k} - x_i}$$

Do stworzenia naiwnego algorytmu opartego na ww. tabeli potrzebna jest tablica dwuwymiarowa, lecz jak możemy zauważyć, jest ona nieoptymalna z punktu widzenia pamięci - używamy tylko połowy zadeklarowanej tablicy. Możemy jednakże skorzystać także z jednowymiarowej tablicy fx:

1. Na początku wartościami $fx[i]$ są $c_{i,0} = f(x_i)$ a potem $c_{i-1,1}, c_{i-2,2}, ..., c_{0,i}$ dla każdego kolejnego wyrazu z $fx[i]$
2. Z każdym przejściem, tworzą się kolejne kolumny z tablicy ilorazów, od dołu do góry, co zapewnia że tablica będzie zawsze zawierać ilorazy potrzebne do dalszych obliczeń.

Pseudokod:

Dane:

- x - wektor długości n+1 zawierający węzły

- f - wektor długości n+1 zawierający wartości funkcji w węzłach

Wyniki:

- fx - wektor długości n+1 zawierający obliczone ilorazy różnicowe

---

Data: x, f
Result: fx

for $i \leftarrow 1$ ***to*** $length(f)$ **do**
  $fx[i] \leftarrow f[i]$
end
for $i \leftarrow 1$ ***to*** $length(f)$ **do**
  for $j \leftarrow length(f)$ ***downto*** i **do**
    $fx[j] \leftarrow \frac{fx[j]-fx[j-1]}{x[j]-x[j-i]}$
  end
  return fx
end

Wielomian Interpolacyjny

Opis :

Celem jest zaimplementowanie funkcji *'warNewton'* obliczającej wartość wielomianu interpolacyjnego stopnia n, w postaci Newtona $N_n(x)$ w punkcie $x = t$ za pomocą uogólnionego wzoru Hornera. Funkcja ma działać w czasie liniowym ($O(n)$).

Działanie algorytmu :

Wzór na wielomian Newtona przedstawiony używając ilorazów różnicowych:

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

Zaletą takiego przedstawienia wzoru jest fakt, że dodanie kolejnych punktów x_i, y_i nie narusza w żaden sposób wcześniej obliczonych współczynników $c_k = f[x_0, x_1, ..., x_k]$ (metoda jest dobrze uwarunkowana pod względem numerycznym). Dla tak wyrażonego wielomianu możemy również w łatwy sposób obliczyć jego wartość stosując uogólniony algorytm Hornera:

1. $\omega_n(x) \leftarrow f[x_0, x_1, ..., x_n]$
2. $\omega_k(x) \leftarrow \omega_{k+1}(x - x_k) + f[x_0, x_1, ..., x_l] \;\; (k = n-1, n-2, ..., 0)$
3. $N_n(x) = \omega_0(x)$

Pseudokod:

Dane:

- x - wektor długości n+1 zawierający węzły
- fx - wektor długości n+1 zawierający wartości funkcji w węzłach
- t - punkt w którym będziemy liczyć wartość wielomianu

Wyniki:

- nt - wartość wielomianu w punkcie t

Data: x, fx, t
Result: nt

$n \leftarrow length(fx)$
$nt \leftarrow fx[n]$
for $i \leftarrow n-1$ ***downto*** 1 **do**
  $nt \leftarrow fx[i] + (t - x[i]) * nt$
end
return nt

Postać naturalna

Opis:

Celem jest zaimplementowanie algorytmu (funkcja *'naturalna'*) obliczającego współczynniki $a_0, a_1, ..., a_n$ postaci naturalnej wielomianu interpolacyjnego dla zadanych współczynników $c_0 = f[x_0], c_1 = f[x_0, x_1], ..., c_n = f[x_0, ..., x_n]$ tego wielomianu w postaci Newtona oraz węzłów $x_0, ..., x_n$, w czasie $(O(n^2))$.

Działanie algorytmu:

Aby znaleźć współczynniki wielomianu interpolacyjnego w postaci naturalnej, zastosowałem uogólniony algorytm Hornera z zadania poprzedniego. Zasada działania opracowanego na tej postawie algorytmu jest następująca:

1. W wielomianie interpolacyjnym n-tego stopnia współczynnik a_n przy najwyższej potędze x jest równy c_n. Wynika z tego że ω_n jest również równy a_n.
2. Mając $\omega_n = a_n$, w kolejnych krokach algorytmu tworzone są wartości a_i bazujące na a_{i+1}.
3. Aby znaleźć zależności między kolejnymi a_i, algorytm przechodzi po każdym ω_i od $i = n$ do $i = 0$, zmieniając tworzone współczynniki postaci naturalnej, tak by doprowadzić do postaci naturalnej.

Pseudokod:

Dane:

- x - wektor długości n+1 zawierający węzły
- fx - wektor długości n+1 zawierający wartości funkcji w węzłach

Wyniki:

- a - wektor dlugośc n+1, zawierający obliczone współczynniki postaci naturalnej

---

Data: x, fx
Result: a

$n \leftarrow length(fx)$
$a[n] \leftarrow fx[n]$
for $i \leftarrow n-1$ ***downto*** 0 **do**
  $a[i] \leftarrow fx[i] - x[i] * a[i+1]$
  for $j \leftarrow i+1$ ***to*** $n-1$ **do**
    $a[j] \leftarrow a[j] - x[j] * a[j+1]$
  end
end
return a

Interpolacja funkcji oraz wykres

Opis:

Celem jest zaimplementowanie funkcji *'rysujNnfx'* intepolującej zadaną funkcję f w przedziale $[a, b]$ za pomocą wielomianu intepolacyjnego n w postaci Newtona, oraz rysującej wykresy funkcji f i otrzymanego wielomianu. Przy interpolacji należało użyć węzłów równoodległych.

Działanie :

1. Wyznaczane są węzły interpolacji $(x_1, x_2, ..., x_{n+1})$, które rozmieszczone są równomiernie w odległości $\frac{b-a}{n}$ (równoodległe), oraz wyznaczana jest wartość funkcji f we wskazanych węzłach.
2. Za pomocą funkcji *ilorazyRoznicowe* obliczane są ilorazy różnicowe dla wskazanych wcześniej węzłów.
3. Dla zwiększenia dokładności wykresów, zarówno funkcja f jak i wielomian są próbkowane w $30 * (n + 1)$ równoodległych punktach. Wartości wielomianu w tych punktach są obliczone za pomocą funkcji *warNewton*.
4. Uzyskane w ten sposób wartości są wystarczające do stworzenia wykresu funkcji f oraz jej wielomianu interpolacyjnego. Do rysowania wykresów użyte zostały bibliteki *Plots* oraz *SymPy*.

Pseudokod:

Dane:

- f - zadana funkcja
- a - początek przedziału interpolacji
- b - koniec przedziału interpolacji
- n - stopień wielomianu

Wyniki:

- plik.png - wykres funkcji i jej wielomianu interpolacyjnego

---

Data: f, a, b, n
Result: a

$x, y, fx \leftarrow array[n + 1]$
$density \leftarrow 30$
$w\_x, w\_y, w\_n \leftarrow array[(n + 1) * density]$
$max\_w \leftarrow n + 1$
$h \leftarrow \frac{(b-a)}{n}$
$k\_h \leftarrow 0.0$ //kolejne wartości h
for $i \leftarrow 1$ ***to*** max_w **do**
  $x[i] \leftarrow a + k\_h$
  $y[i] \leftarrow f(x[i])$
  $k\_k \leftarrow k\_h + h$
end
$fx \leftarrow ilorazyRoznicowe(x, y)$
$k\_h \leftarrow 0.0$
$max\_w \leftarrow max\_w * density$
for $i \leftarrow 1$ ***to*** max_w **do**
  $w\_x[i] \leftarrow a + k\_h$
  $w\_y[i] \leftarrow f(w\_x[i])$
  $w\_n[i] \leftarrow warNewton(x, fx, w\_x[i])$
  $k\_h \leftarrow k\_h + h$
end
$draw(w\_x, w\_y, w\_n)$

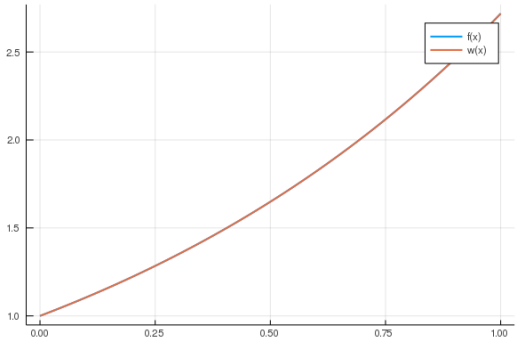
Interpolacja funkcji e^x oraz $x^2 * sinx$

Opis:

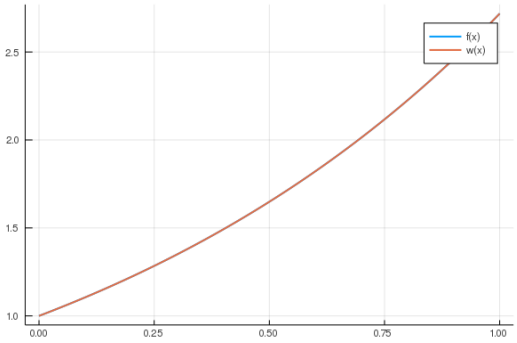
Zastosowanie funkcji *rysujNnfx* na następujących przykładach:

- $f(x) = e^x$, $[a, b] = [0, 1]$, $n \in \{5, 10, 15\}$
- $f(x) = x^2 * sinx$, $[a, b] = [-1, 1]$, $n \in \{5, 10, 15\}$

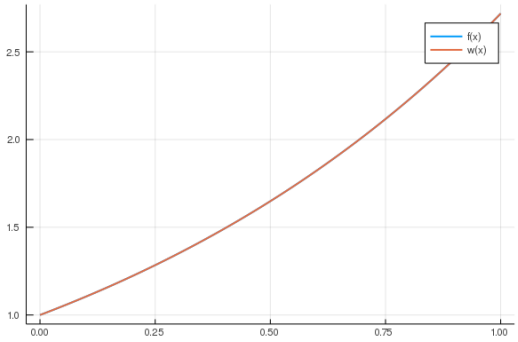
Wyniki:



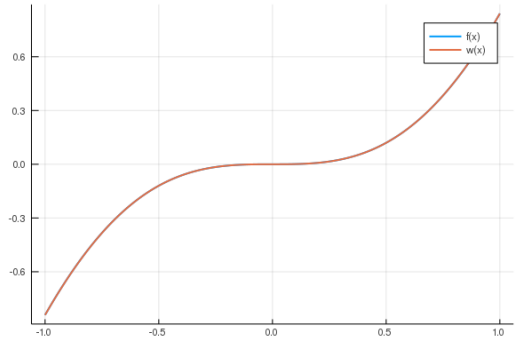
$f(x) = e^x$, n = 5



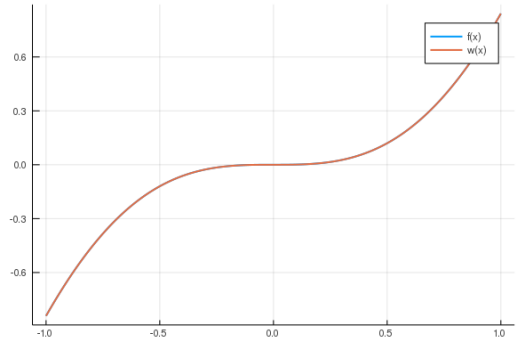
$f(x) = e^x$, n = 10



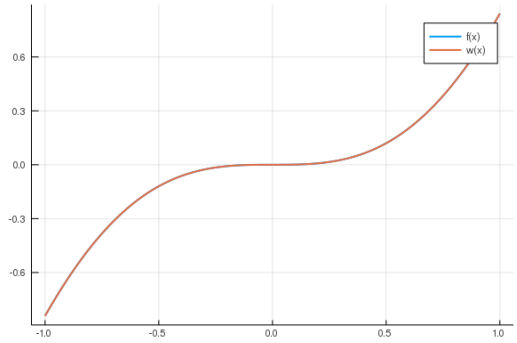
$f(x) = e^x$, n = 15



$f(x) = x^2 * sinx$, n = 5



$f(x) = x^2 * sinx$, n = 10



$f(x) = x^2 * sinx$, n = 15

Wnioski:

Dla obu funkcji, na zadanych przedziałach wielomiany interpolacyjne są bardzo bliskie interpolowanym funkcjom, na żadnym z wykresów nie widać rozbieżności. W tym przypadku zastosowanie równoodległych węzłów interpolacji dało bardzo dobre przybliżenia funkcji interpolowanych.

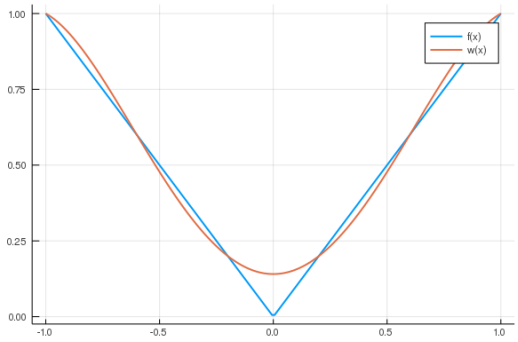
Interpolacja funkcji $|x|$ oraz $\frac{1}{1+x^2}$

Opis:

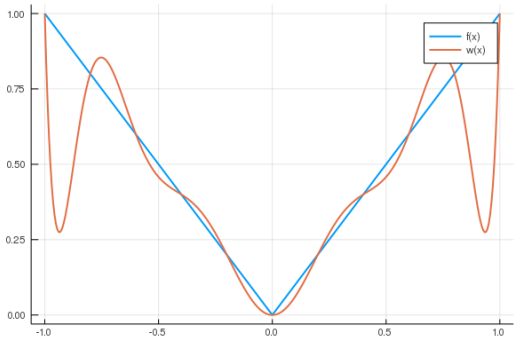
Zastosowanie funkcji *rysujNnfx* na następujących przykładach:

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

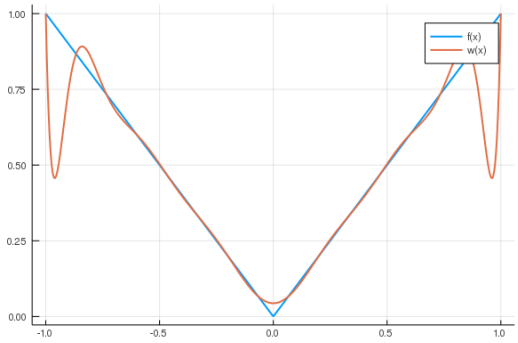
Wyniki:



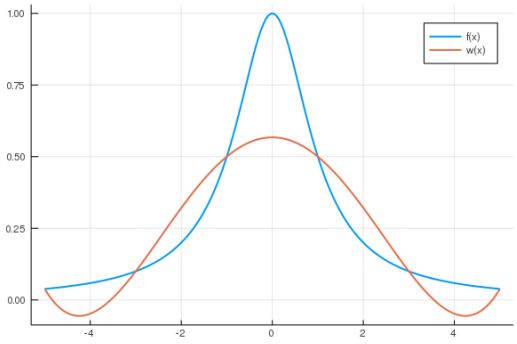
$f(x) = |x|$, n = 5



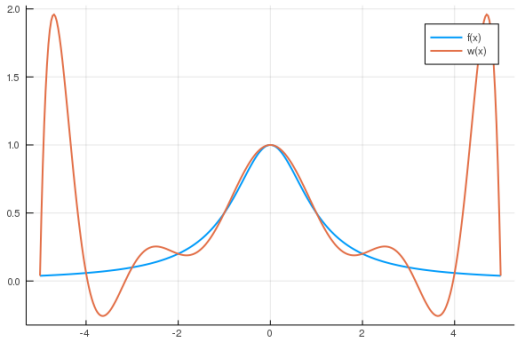
$f(x) = |x|$, n = 10



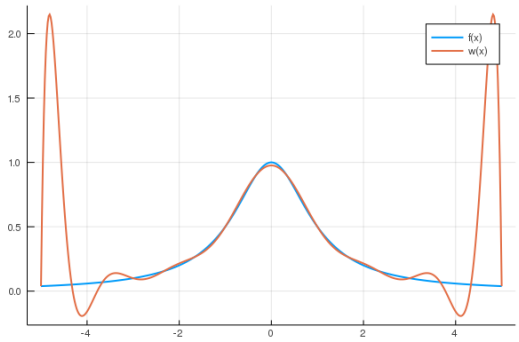
$f(x) = |x|$, n = 15



$f(x) = \frac{1}{1+x^2}$, n = 5



$f(x) = \frac{1}{1+x^2}$, n = 10



$f(x) = \frac{1}{1+x^2}$, n = 15

Wnioski:

Dla obu przypadków obserwujemy wyraźne rozbieżności, szczególnie dla końców przedziału.

- Funkcja $|x|$ nie jest różniczkowalna, co w największym stopniu dopwiada za odchylenia w tym przypadku.
- Dla funkcji $\frac{1}{1+x^2}$ obserwujemy *efekt Runge'go* - pogorszenie jakości interpolacji wielomianowej, mimo zwiększenia ilości węzłów. Jest to typowe zjawisko dla przypadku gdy węzły wielomianu są równoodległe, a sam wielomian jest wysokiego stopnia, co powoduje znaczne odchylenia wartości wielomianu od wartości funkcji na końcach badanym przedziale. Efekt występuje również, gdy badana funkcja jest nieciągła, lub odbiega znacząco do funkcji gładkiej.

Głónym czynnikiem, który wpływa na niską dokładność powyższych przybliżeń, jest fakt równoodległości węzłów, przez co w miejscu o trudniejszej interpolacji przypada ich stosunkowo niewiele. Uniwersalnym sposobem na poprawę interpolacji w ww. przypadkach jest stosowanie wielomianów opartych na węzłach Czebyszewa, które dzięki oparciu na węzłach mocno zagęszczonych przy końcach przedziałów mają znacznie mniejsze oscylacje. W celu najdokładniejszego przybliżenia funkcji, należy dobrać węzły w sposób otymalny (na przykład stosując zera wielomianu Czebyszewa).