Obliczenia naukowe **2017/2018**

Prowadzący: dr hab. Paweł Zieliński czwartek TN, 11:15

Agata Jasionowska 229726

Laboratorium – Lista 3

1. Zadanie 1

1.1. Opis problemu

Zadanie polegało na implementacji funkcji obliczającej ilorazy różnicowe na podstawie podanego wektora węzłów oraz wartości funkcji, jednak bez użycia tablicy dwuwymiarowej.

Iloraz różnicowy k-tego rzędu spełnia poniższą zależność:

$$f[x_0, x_1, \ldots, x_k] = \frac{f[x_1, x_2, \ldots, x_k] - f[x_0, x_1, \ldots, x_{k-1}]}{x_k - x_0},$$

zaś ilorazy rzędu zerowego oraz pierwszego równe są kolejno:

$$f[x_0] = f(x_0), \qquad f[x_0, x_1] = \frac{f(x_1) - f(x_0)}{x_1 - x_0}.$$

1.2. Opis rozwiązania

Znając węzły x_k i wartości funkcji $f(x_k)$ (czyli również ilorazy $f[x_k]$ zerowego rzędu) można za pomocą tego wzoru utworzyć dwuwymiarową tablicę ilorazów różnicowych wyższych rzędów. Jednak algorytm takiej konstrukcji jest nieefektywny, gdyż wystarczy użyć jednowymiarowej tablicy d zmiennych z jednym wskaźnikiem. Początkowymi wartościami zmiennych d_i są odpowiadające im $f[x_i]$, czyli każde d_i obliczane jest ze wzoru:

$$d_i = \frac{d_i - d_{i-1}}{x_i - x_{i-j}},$$

gdzie j oznacza numer kolumny. Każda kolejna wartość uaktualniana jest kolumnami, zaś wewnątrz każdej kolumny - z dołu do góry. Dzięki takiej kolejności obliczeń tablica d zawiera w każdej chwili ilorazy, które będą później potrzebne.

Szczegółowy przebieg metody został przedstawiony w Algorytmie 1.

Algorytm 1: Ilorazy różnicowe

```
Input  :  x  —  wektor długości n + 1 zawierający węzły x_0, ..., x_n,
          f  —  wektor długości n + 1 zawierający wartości interpolowanej funkcji
                w węzłach f(x_0), ..., f(x_n).
Output:   fx —  wektor długości n + 1 zawierający obliczone ilorazy różnicowe.
Function ilorazyRoznicowe(x, f)
    for i ← 1 to n do
        fx[i] ← f[i]
    end
    for j ← 1 to n do
        for i ←n downto i do
            fx[i] ← (fx[i] − fx[i − 1])/(x[i] − x[i − j])
        end
    end
    return fx
```

2. Zadanie 2

2.1. Opis problemu

Zadanie polegało na implementacji funkcji obliczającej wartość wielomianu interpolacyjnego w postaci Newtona w zadanym punkcie za pomocą uogólnionego algorytmu Hornera.

$$p_k(x) = \sum_{i=0}^{k} c_i \prod_{j=0}^{i-1} (x - x_j).$$

Metoda Newtona, pomimo iż jest trudniejsza w implementacji niż np metoda Lagrange'a i wymaga większej ilości oddzielnych "kroków", jest dużo lepiej uwarunkowana numerycznie, przez co warta poznania pod kątem stosowania na maszynach cyfrowych. W metodzie najpierw wyznaczane są odpowiednie ilorazy różnicowe, a dopiero później z ich użyciem obliczana jest interpolowana funkcja.

Następujący wzór podaje postać Newtona wielomianu w:

$$w(x) = \sum_{k=0}^{n} b_k p_k(x),$$

dla punktów rzeczywistych $x_0, x_1, \ldots$, wielomianów $p_0, p_1, \ldots$ oraz $w \in \prod_n$. Uogólniony algorytm Hornera prezentuje się następująco:

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

2.2. Opis rozwiązania

Na podstawie przedstawionego wyżej algorytmu Hornera możliwe jest napisanie funkcji wyznaczającej wartość wielomianu w punkcie w czasie O(n). Szczegóły przedstawiono w poniżej (Algorytm 2).

Algorytm 2: Wielomian interpolacyjny Newtona.

Input : 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.

Output: nt — wartość wielomianu w punkcie t.

```
Function warNewton(x, fx, t)
    n ← length(fx)
    nt ← fx[n]
    for i ← n − 1 downto 1 do
        nt ← fx[i] + (t − x[i]) ∗ nt
    end
    return nt
```

3. Zadanie 3

3.1. Opis problemu

Zadanie polegało na implementacji funkcji obliczającej współczynniki postaci naturalnej wielomianu interpolacyjnego w postaci Newtona.

3.2. Opis rozwiązania

Przebieg tej metody następuje zgodnie z Algorytmem 3.

Algorytm 3: Współczynniki naturalne wielomianu interpolacyjnego.

Input : x — wektor długości $n+1$ zawierający węzły $x_0, \dots, x_n$,
fx — wektor długości $n+1$ zawierający ilorazy różnicowe.

Output: a — wektor długości $n+1$ zawierający obliczone współczynniki postaci naturalnej.

```
Function naturalna(x, fx)
    n ← length(fx)
    a [n ] ← fx[n]
    for i ← n − 1 downto 1 do
        a[i] ← fx[i] − a[i + 1] ∗ x[i]
        for j ← i + 1 downto n − 1 do
            a[j] ← a[j] − a[j + 1] ∗ x[i]
        end
    end
    return a
```

4. Zadanie 4

4.1. Opis problemu

Zadanie polegało na implementacji funkcji, która interpoluje zadaną funkcję $f(x)$ w przedziale $[a, b]$ za pomocą wielomianu interpolacyjnego stopnia n w postaci Newtona, a następnie rysuje ten wielomian oraz $f(x)$.

4.2. Opis rozwiązania

W celu rozwiązania zadania zastosowano metody `ilorazyRoznicowe` oraz `warNewton` utworzone w poprzednich zadaniach. Rysowanie wykresów zrealizowano przy pomocy funkcji pakietu `Gadfly`.

Przebieg tej metody następuje zgodnie z Algorytmem 4.

Algorytm 4: Wielomian interpolacyjny Newtona

Input : f — funkcja $f(x)$ zadane jako anonimowa funkcja,
a,b — przedział interpolacji,
n — stopień wielomiany interpolacyjnego.

Output: — Funkcja rysuje wielomian interpolacyjny i interpolowaną funkcję w przedziale [a, b].

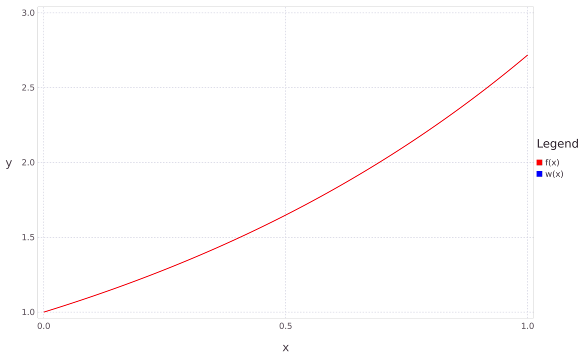
```
Function naturalna(f, a, b, n)
    kh ← 0
    max ← n + 1
    h ← (b − a)/(max − 1)
    mult ← 15
    for i ← 1 to max do
        x[i] ← a + kh
        y[i] ← f(x[i])
        kh ← kh + h
    end
    fx ← ilorazyRoznicowe(x, y)
    kh ← 0
    max ← max * mult
    h ← (b − a)/(max − 1)
    for i ← 1 to max do
        plot_x[i] ← a + kh
        plot_h[i] ← warNewton(x, fx, plot_x[i])
        plot_y[i] ← f(plot_x[i])
        kh ← kh + h
    end
```

5. Zadanie 5

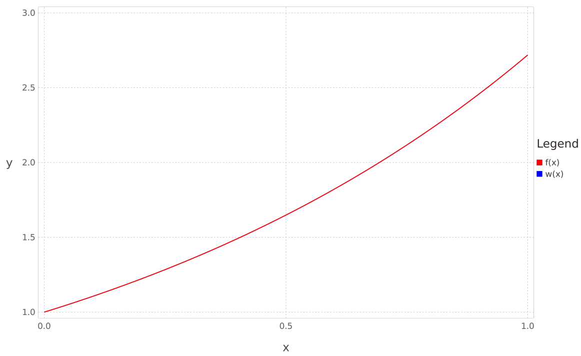
5.1. Opis problemu

Celem zadania było przetestowanie utworzonej w Zadaniu 4 funkcji `rysujNnfx` na następujących przykładach:

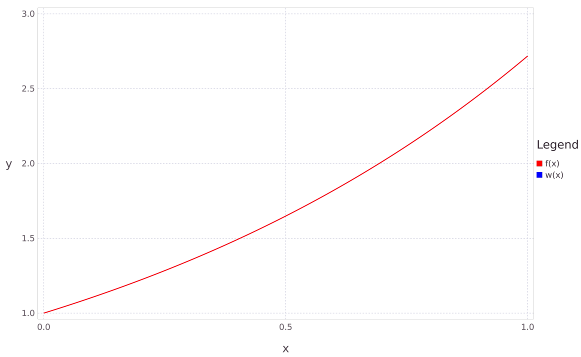
(a) $\exp x,\ [0,1],\ n = 5, 10, 15,$
(b) $x^2 \sin x,\ [-1,1],\ n = 5, 10, 15.$



(a) $n = 5$

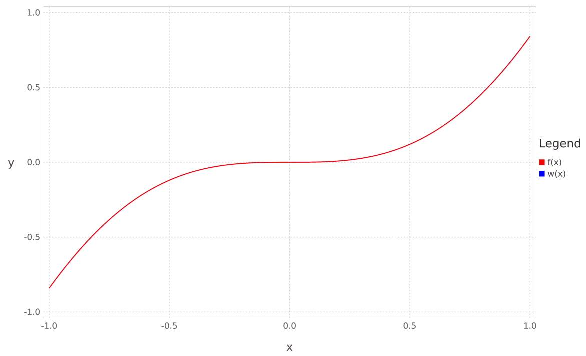


(b) $n = 10$

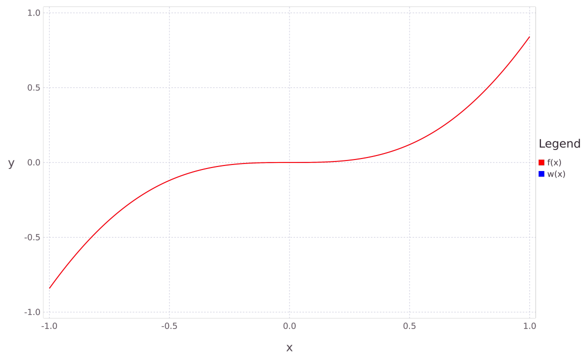


(c) $n = 15$

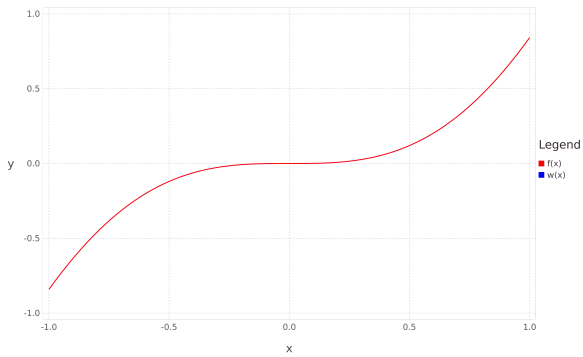
Rysunek 1: $\exp x$



(a) $n = 5$



(b) $n = 10$



(c) $n = 5$

Rysunek 2: $x^2 \cdot \sin x$

5.2. Opis rozwiązania

5.3. Wyniki

5.4. Wnioski

6. Zadanie 6

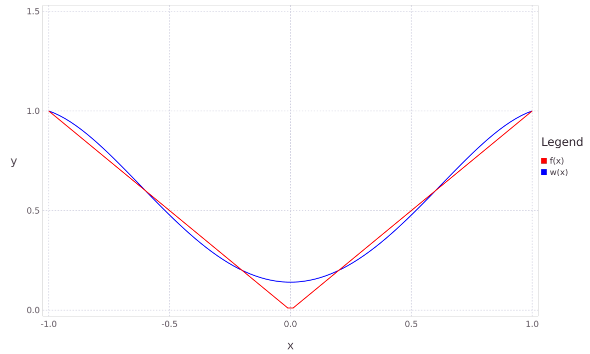
6.1. Opis problemu

Celem zadania było przetestowanie utworzonej w Zadaniu 4 funkcji `rysujNnfx` na następujących przykładach:

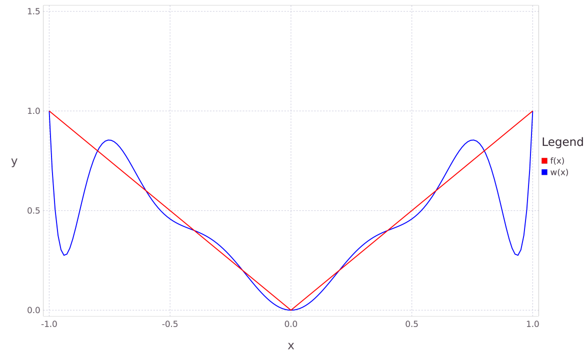
(a) $|x|, \ [-1,1], \ n = 5, 10, 15,$
(b) $\frac{1}{1+x^2}, \ [-5,5], \ n = 5, 10, 15.$

6.2. Opis rozwiązania

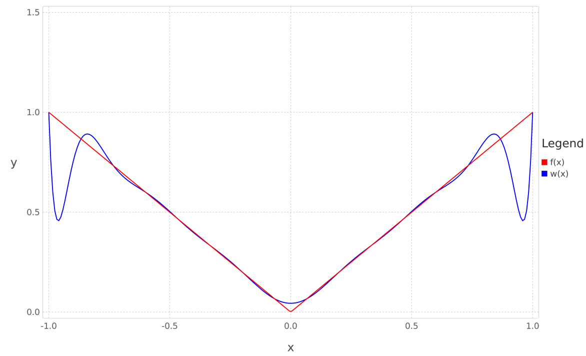
6.3. Wyniki



(a) $n = 5$

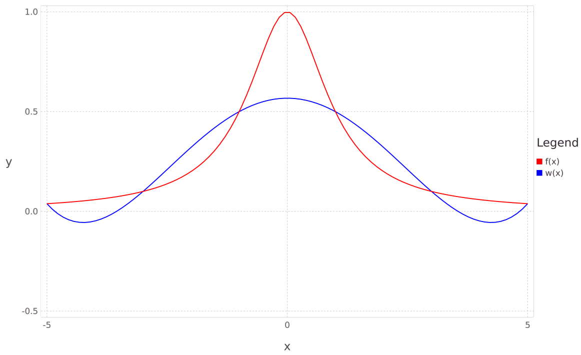


(b) $n = 10$

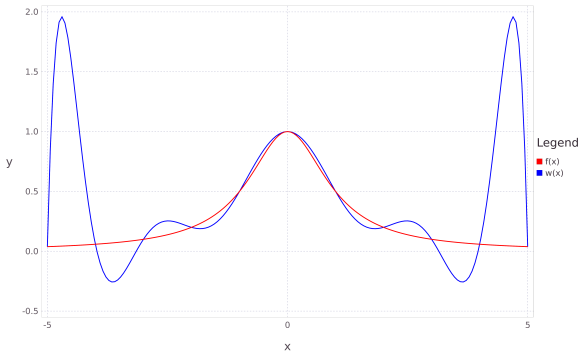


(c) $n = 15$

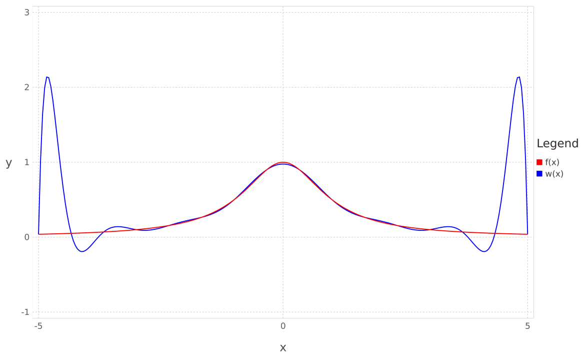
Rysunek 3: $|x|$



(a) $n = 5$



(b) $n = 10$



(c) $n = 15$

Rysunek 4: $\frac{1}{1+x^2}$

6.4. Wnioski

W przypadku funkcji $|x|$ duże odchylenia spowodowane były tym, iż nie jest ona różniczkowalna.

Dla drugiej funkcji zaobserwować można ciekawe zjawisko. Otóż początkowo ze wzrostem liczby węzłów przybliżenie poprawia się, jednak sytuacja ulega zmianie wraz z dalszym zwiększaniem n — widoczne jest znaczne odchylenie, szczególnie uwydatnione na końcach przedziałów. Taki efekt nosi nazwę zjawiska Rungego i oznacza pogorszenie jakości interpolacji wielomianowej, mimo zwiększenia liczby jej węzłów. Zastosowanie podziału równoległego skutkuje stosunkowo niewielką liczbą punktów na krawędziach przedziału, przez co wielomian interpolacyjny zdecydowanie odbiega tam od właściwej wartości funkcji, zaś stan ten pogarsza się przy zagęszczeniu węzłów równoodległych. Aby zwiększyć dokładność interpolacji należy zmienić rozkład punktów. Problem ten sformułował rosyjski matematyk P.L. Czebyszew, jako problem znalezienia wielomianu, który najlepiej przybliżałby zero na danym przedziale.

W celu uniknięcia zaobserwowanych błędów najlepszym rozwiązaniem byłoby zastosowanie interpolacji z węzłami rozmieszczonymi gęściej w pobliżu miejsc ”oscylacji” (np. miejsca zerowe wielomianu Czebyszewa).