Vadym Liss (257264)
Lista 4

Zadanie 1

Opis algorytmu

Aby obliczyć ilorazy różnicowe należy skorzystać z faktu, iż $f[x_i] = f(x_i) = y_i$ oraz ze wzoru $f[x_i, x_{i+1}, \ldots, x_{i+j}] = \frac{f[x_{i+1}, x_{i+2}, \ldots, x_{i+j}] - f[x_i, x_{i+1}, \ldots, x_{i+j-1}]}{x_{i+j} - x_i}$. Na początku za fx przyjmujemy kopię wektora f, a następnie w podwójnej pętli obliczamy kolejne ilorazy różnicowe. Zakładając, że $f[x_i, x_j] = \frac{f(x_j) - f(x_i)}{x_j - x_i}$ w prosty sposób możemy skonstruować tablicę trójkątną, łatwą do reprezentacji graficznej. Analizując jej konstrukcję, można zauważyć, że interesujący nas wynik znajduje się w jej pierwszym rzędzie, więc algorytm nie musi korzystać z macierzy (zgodnie z treścią zadania).

Na początku, tworzona jest kopia wektora f (fx), która będzie przechowywała ostateczny wynik. Następnie swoje działanie rozpoczynają dwie pętle: zewnętrzna, odpowiedzialna za nawigację między kolejnymi kolumnami tablicy trójkątnej; oraz wewnętrzna, która uzupełnia wartości w kolejnych wierszach każdej kolumny, wędrując od ostatniego wiersza (dzięki czemu nie grozi nam utrata potrzebnych wartości). Z każdą iteracją kolejne kolumny są nadpisywane nowymi wartościami, zgodnie z podanym wcześniej wzorem. Obliczenia są zatem wykonywane bez użycia macierzy.

Dzięki temu, że wartości są aktualizowane od końca wektora, dostępne są informacje do obliczenia kolejnych ilorazów różnicowych, co robimy aż do momentu, gdy wszystkie wartości wektora fx będą miały postać $f_{1,k}$, dla $k = 1,2, \ldots, n+1$. Po zakończonych iteracjach, zwracany jest wektor reprezentujący pierwszy rząd tablicy trójkątnej.

Dane testowe:

Parametr	Dane
x_i	-2.0, -1.0, 0.0, 1.0, 2.0, 3.0
$f(x_i)$	-25.0, 3.0, 1.0, -1.0, 27.0, 235.0

Wyniki:

Array{Float64, 1}:
[-25.0, 28.0, -15.0, 5.0, 0.0, 1.0]

Wynik zwrócony przez algorytm jest identyczny z tym, który otrzymaliśmy podczas rozwiązywania zadania (również korzystając z tablicy trójkątnej).

Zadanie 2

Opis algorytmu

Wielomian interpolacyjny Newtona możemy przedstawić następująco: Taka reprezentacja pozwala na obliczenie wartości tegoż wielomianu, w dowolnym punkcie, za pomocą następujących wzorów (uogólniony algorytm Hornera):
1) $w_n(x) = f[x_0, x_1, \ldots, x_n]$
2) $w_k(x) = f[x_0, x_1, \ldots, x_k] + (x - x_k)w_{k+1}(x)$, gdzie $k = n-1, \ldots, 0$
3) $N_n(x) = w_0(x)$
Wzory te są wykorzystane w podanym algorytmie.
Na początku tworzona jest zmienna nt, która będzie przechowywała ostateczny wynik. Ponieważ pętla będzie iterowała po elementach fx od końca (k = n-1), wartość zmiennej nt ustawiamy na ostatnią wartość przechowywaną w wektorze fx. Kolejne iteracje pętli to de facto kolejne kroki uogólnionego algorytmu Hornera (w postaci wzoru 2)). Gdy pętla zakończy swoje działanie, zwracany jest wynik, przechowywany przez zmienną nt.
Koszt wykonania tego algorytmu to $O(n)$ - jedna pętla wykonująca się n-1 razy.

Dane testowe:

Parametr	Dane
x_i	-2.0, -1.0, 0.0, 1.0, 2.0, 3.0
$f(x_i)$	-25.0, 3.0, 1.0, -1.0, 27.0, 235.0

Wyniki:

235. Zwrócony wynik pokrywa się z wartością $f(x_3)$.

Zadanie 3

Opis algorytmu

Obliczenie współczynników wielomianu dla jego postaci naturalnej również opiera się na wykorzystaniu wzorów, które podałem w poprzednim zadaniu. Sama konstrukcja algorytmu jest bardzo podobna, z wyjątkiem paru istotnych różnic, które objaśnię poniżej.

Na początku działania, tworzony jest wektor a, który będzie przechowywał obliczone współczynniki. Analogicznie jak w zadaniu 2, obliczenia będą wykonywane "od końca", dlatego $a[n] = fx[n]$. Następnie pojawiają się dwie pętle, pierwsza, podobnie jak w zadaniu 2, oblicza "składowe" wielomiany (zgodnie z podanymi wzorami), jednak tym razem są one osobno zapisywane w wektorze a. Pętla wewnętrzna ma za zadanie doprowadzić utworzony w pętli zewnętrznej, "składowy" wielomian do postaci naturalnej. Kiedy pętla zakończy działanie (przeiteruje po każdym elemencie fx), zostaje zwrócony wektor a, ze współczynnikami dla postaci naturalnej wielomianu interpolacyjnego.
Koszt wykonania tego algorytmu to $O(n^2)$ - pętla zewnętrzna wykonuje się $n-1$ razy, a pętla wewnętrzna co najwyżej n razy. W przypdaku pesymistycznym mamy zatem: $O(n*(n-1)) = O(n^2)$.

Dane testowe:

Parametr	Dane
x_i	-2.0, -1.0, 0.0, 1.0, 2.0, 3.0
$f(x_i)$	-25.0, 3.0, 1.0, -1.0, 27.0, 235.0

Wyniki:

Array{Float64, 1}:
[1.0, -3.0, 0.0, 0.0, 0.0, 1.0]

Postać naturalna sprawdzanego wielomianu (-25 + 28(x+2)-15(x+2)(x+1) + 5(x+2)(x+1)x + (x+2)(x+1)x(x-1)(x-2)) to x^5 - 3x + 1, czyli zgodnie ze specyfikacją tego zadania: [1, -3, 0, 0, 0, 1]. Wynik zwrócony przez algorytm jest zatem poprawny.

Zadanie 4

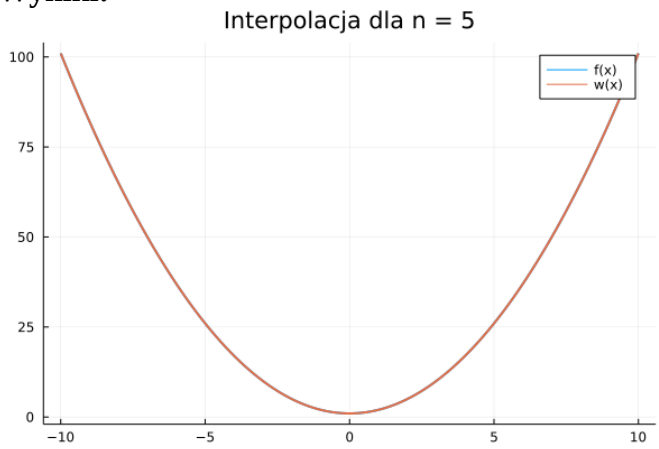
Opis algorytmu

Wykonanie interpolacji funkcji opiera się na algorytmach, które zostały zaimplementowane wcześniej (ilorazy różnicowe (zad1), obliczanie wartości wielomianu w punkcie w postaci Newtona (zad2)). Elementem w algorytmie, który został dodany to wyznaczenie równoodległych od siebie węzłów, które wykonywane jest w pierwszej pętli. Wiąże się to z następującymi wzorami: $x_k = a + kh, h = \frac{b-a}{n}$ dla $k = 0,1, \dots, n$.
Na początku ustalana jest dokładność rysowanych wykresów (mnożnik accuracy, który zwiększa liczbę węzłów, na których zostanie wywołana funkcja warNewton. Po zdefiniowaniu potrzebnych zmiennych i wektorów, wyznaczane są węzły: ustalana jest odległość h oraz inicjalizowany współczynnik kh. Następnie w pętli obliczane jest n_{max} węzłów oraz wartości podanej funkcji (f), w tych węzłach. Tak przygotowane dane są przekazywane do funkcji ilorazyRoznicowe. W kolejnym kroku, będą liczone wartości wielomianu interpolacyjnego, za pomocą funkcji warNewton. Przed wykonaniem pętli, wykonującej to zadanie, ponownie wyznaczane są węzły (tym razem będzie ich więcej, tak żeby wykres funkcji był dokładniejszy). Otrzymane w ten sposób dane (plot_x, plot_y oraz wartości po przeprowadzonej interpolacji - plot_ip) przedstawiane są na wykresie.

Dane testowe:

Parametr	Dane
f	x->x^2 + 1
a	-10.0
b	10.0
n	5

Wyniki:

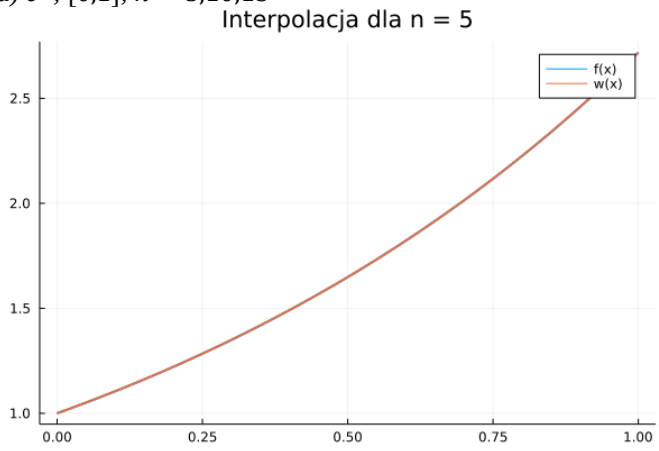


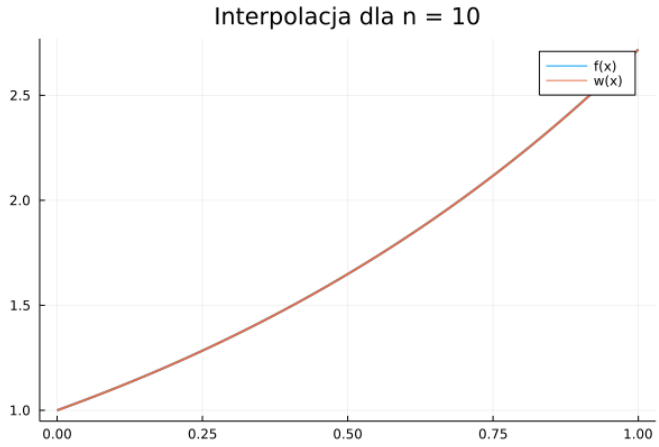
Dla funkcji $x^2 + 1$ wielomian interpolacyjny pokrywa się z wykresem funkcji. Algorytm działa.

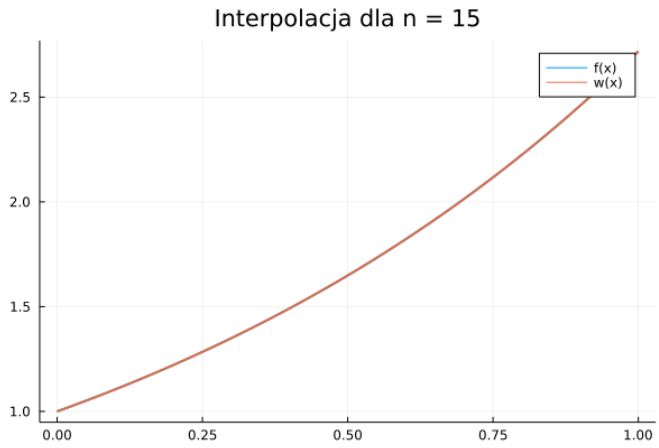
Zadanie 5

Przetestować funkcję rysujNnfx(f,a,b,n) na następujących przykładach:

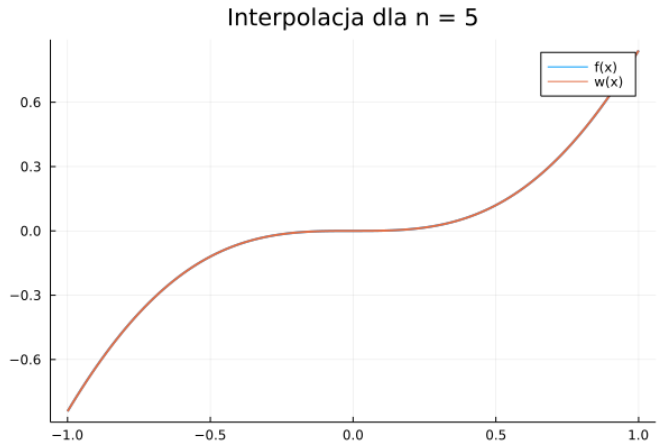
a) e^x, $[0,1]$, $n = 5,10,15$

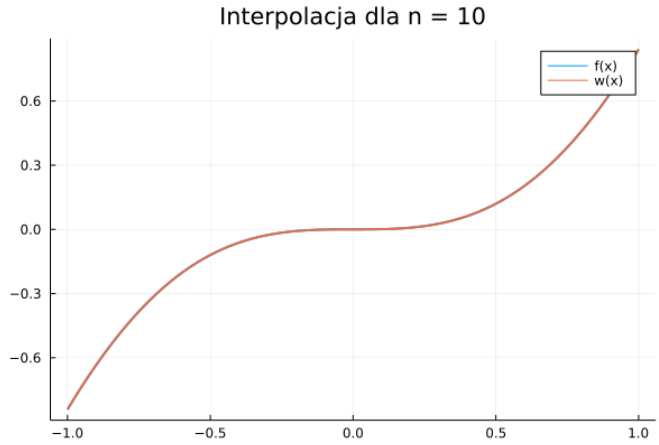


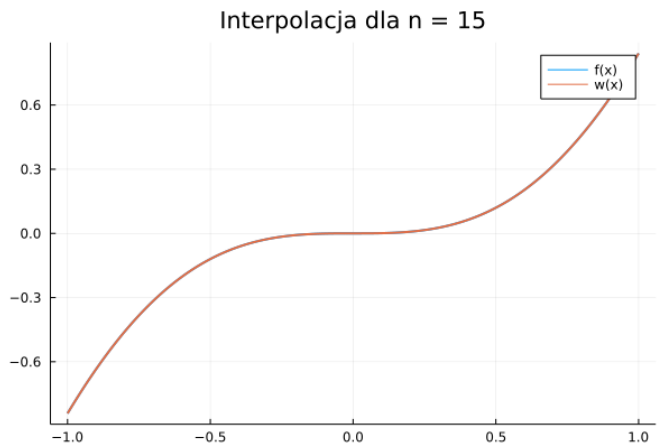




b) $x^2 sinx$, $[-1,1]$, $n = 5,10,15$





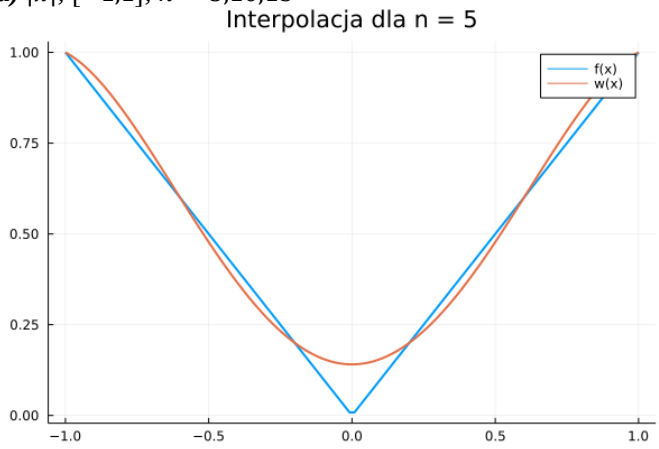


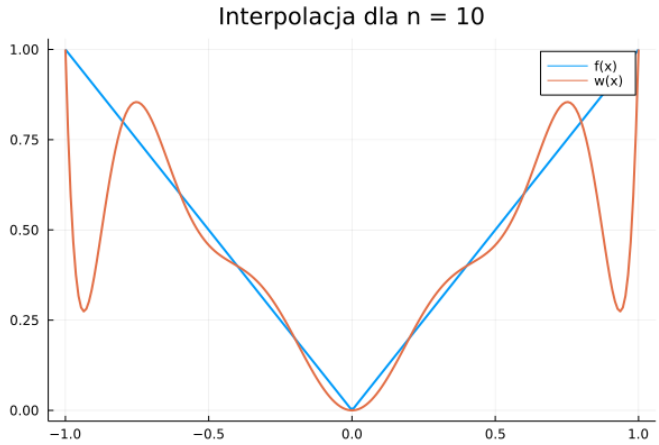
Wniosek: Interpolacja dała dobre przybliżenie obydwu funkcji, wykres wielomianu praktycznie pokrywa się z faktycznym wykresem funkcji, nawet dla najmniejszych wartości n. Powodem tak dobrego przybliżenia jest m.in dobrze zdefiniowane generowanie węzłów (generowanie węzłów równoodległych), dzięki czemu obraz funkcji, jaki oddają jej wartości w węzłach jest ogólny i nie skupia się na jakimś fragmencie przedziału $[a, b]$, kosztem informacji o innym.

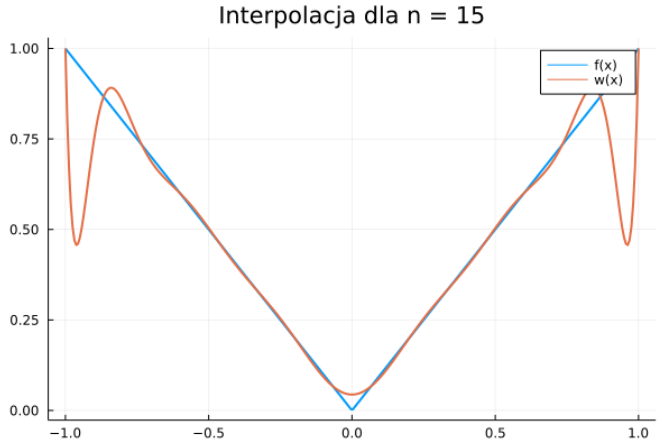
Zadanie 6

Przetestować funkcję rysujNnfx(f,a,b,n) na następujących przykładach (zjawisko rozbieżności):

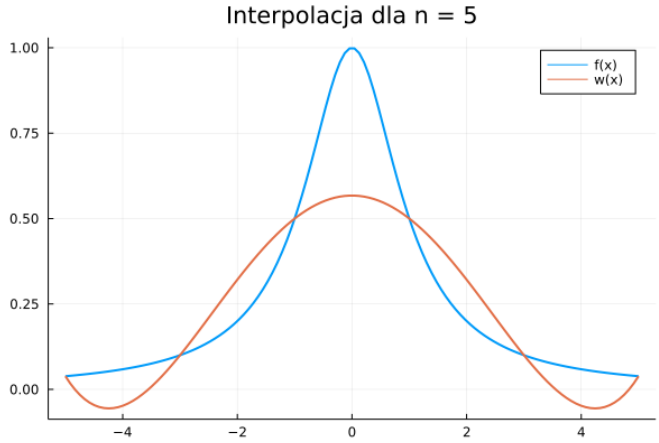
a) $|x|$, $[-1,1]$, $n = 5,10,15$

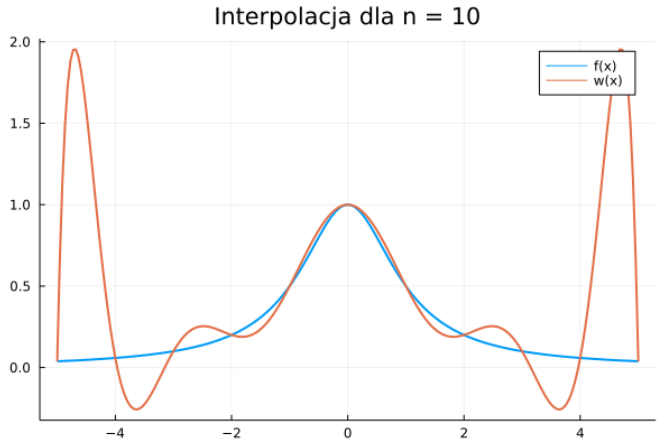


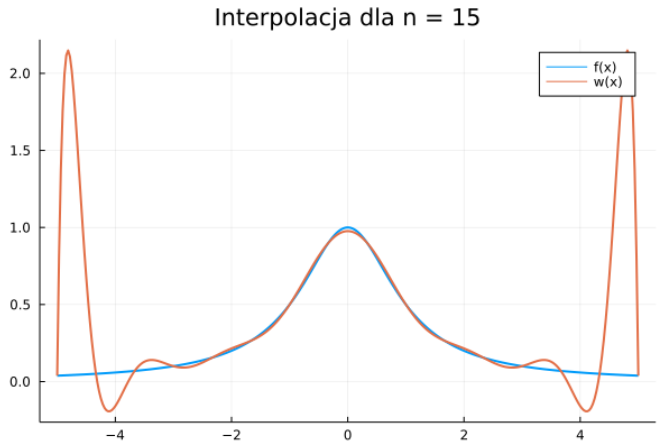




b) $\frac{1}{1+x^2}$, $[-5,5]$, $n = 5,10,15$







Wniosek: Dla obydwu funkcji można zaobserwować, iż wraz ze wzrostem stopnia wielomianu interpolacyjnego przybliżenie polepsza się na środku przedziału, ale pogarsza się na jego końcach. W przypadku funkcji $f(x) = |x|$ wykresy nie pokrywają się dla każdego parametru n. Dla małego stopienia wielomianu interpolacyjnego przybliżenia są w miarę dokładne (jednak nadal jest widoczna rozbieżność). Im stopień wyższy tym większa jest rozbieżność między wykresami, na końcach podanego przedziału. Funkcja $f(x) = |x|$ nie jest różniczkowalna w całej swojej dziedzinie. To właśnie w wybranym przedziale [-1, 1] funkcja nie posiada pochodnej (dla x_0 = 0), dlatego przybliżenie nie jest wystarczająco dokładne i zachodzi zjawisko rozbieżności. Natomiast w przypadku funkcji $f(x) = \frac{1}{1+x^2}$ wykresy również się nie pokrywają. Wzrost liczby węzłów pogarsza otrzymywane wyniki (powinno być przeciwnie), zwiększając rozbieżności m.in. na końcach przedziału. Obserwujemy zatem zjawisko Runge'ego, które jest typowe dla algorytmów wykorzystujących węzły równoodległe oraz wielomiany interpolacyjne wysokiego stopnia. Polega ono na tym, że paradoksalnie po zwiększeniu liczby węzłów otrzymujemy gorsze przybliżenie funkcji i dotyczy ono zazwyczaj węzłów równomiernie rozmieszczonych, czyli takich, jakie tworzone są w funkcji z zadania 4. Aby uniknąć tego zjawiska stosuje się interpolację z gęściej rozmieszczonymi węzłami na końcach przedziałów.