Sprawozdanie MOwNiT
Laboratorium 5

Mateusz Buta

Zadanie 1

Własna implementacja interpolacji wielomianowej stosując wprost wzór na wielomian interpolacyjny Lagrange'a.

1 Wielomian l_k

l_k to taki wielomian, który jest zależny od węzłów $x_0, x_1, \dots, x_n$, ale nie zależy od wartości $y_0, y_1, \dots, y_n$. Ma taką własność, że dla $x = x_k$: $l_k(x_k) = 1$, a dla pozostałych węzłów $i \neq k l_k : (x_i) = 0$. Wielomian ten ma postać:

$$l_k = \prod_{i=0, i\neq k}^{n} \frac{x - x_i}{x_k - x_i}$$

```
function lagrageBase(k,xs,x)
    product=1;
    for i in 1:length(xs)
        if(i!=k)
            product = product*((x-xs[i])/(xs[k]-xs[i]))
        end
    end
    product
end
```

Listing 1: Funkcja wielomianu l_k

2 Wielomian interpolacyjny Lagrange'a

Na podstawie wielomianu l_k można wyrazić wielomian interpolacyjny Lagrange'a $p(x)$ w postaci:

$$p(x) = \sum_{k=0}^{n} y_k l_k(x)$$

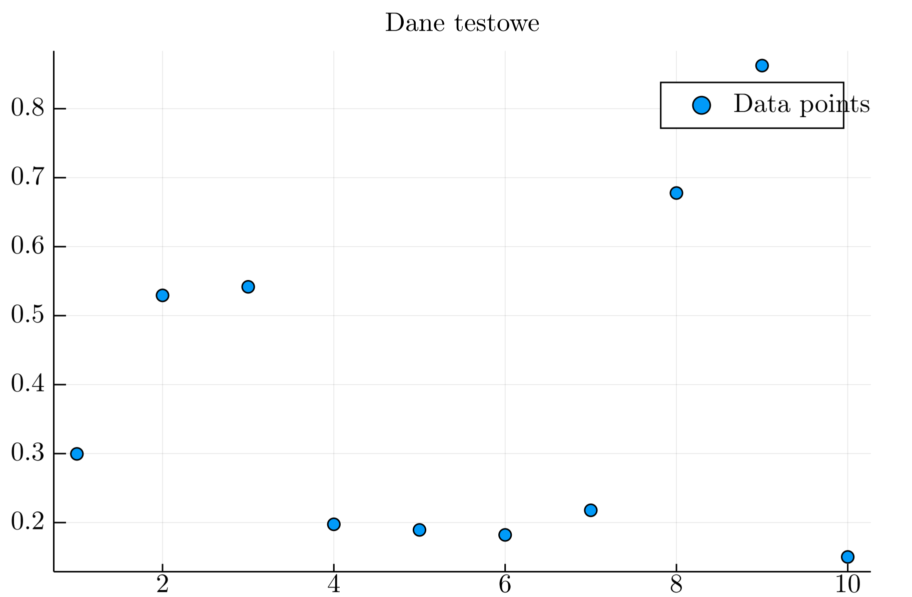
```
function lagrangeFit(xs, y)
    function lagrangeFitX(xs, y, x)
        sum = 0;
        for k in 1:length(xs)
            sum+=y[k]*lagrageBase(k,xs,x)
        end
        sum
    end
    x -> lagrangeFitX(xs,y,x)
end
```

Listing 2: Funkcja obliczająca wielomian interpolacyjny Lagrange'a

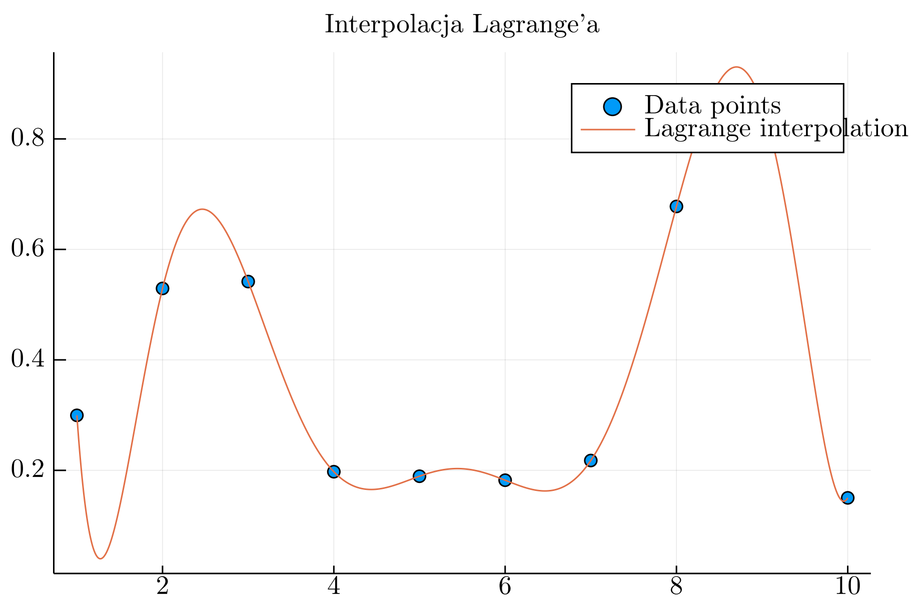
3 Testowanie rozwiązania

Testowanie implementacji na wylosowanych węzłach interpolacji w wybranym przedziale. Wykres wielomianu interpolacyjnego w tym przedziale wraz z wezlami interpolacji.

Rysunek 1



Rysunek 2



Zadanie 2

Napisać własną implementację interpolacji wielomianowej stosując metodę ilorazów róznicowych. Wielomain interpolujący wyrażony jest w postaci:

$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + ... + a_n(x - x_0)(x - x_1)...(x - x_{n-1})$

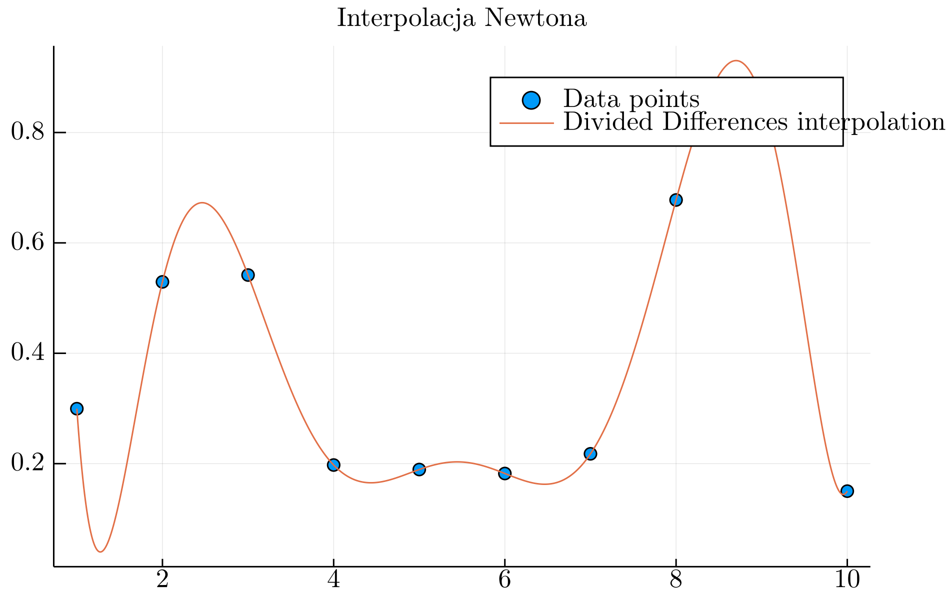
```
function dividedDifferencesFit(xs,A)
    size = length(A)
    D = zeros(Float64, size, size)
    D[:,1]=A

    for ibis in size-1:-1:1
        i=size-ibis+1
        for j in 1:ibis
            D[j,i]=(D[j+1,i-1]-D[j,i-1])/(xs[j+i-1]-xs[j])
        end
    end
    function dividedDifferences(xs,A,x)
        sum=0
        for i in 1:size
            prod=1
            for j in 1:i-1
                prod*=(x-xs[j])
            end
            sum+=prod*D[1,i]
        end
        sum
    end
    x -> dividedDifferences(xs,A,x)
end
```

Listing 3: Funkcja obliczająca wielomian metodą ilorazów róznicowych

4 Testowanie rozwiązania

Testowanie implementacji na wylosowanych węzłach interpolacji w wybranym przedziale. Wykres wielomianu interpolacyjnego w tym przedziale wraz z wezlami interpolacji.

Rysunek 3



Zadanie 3

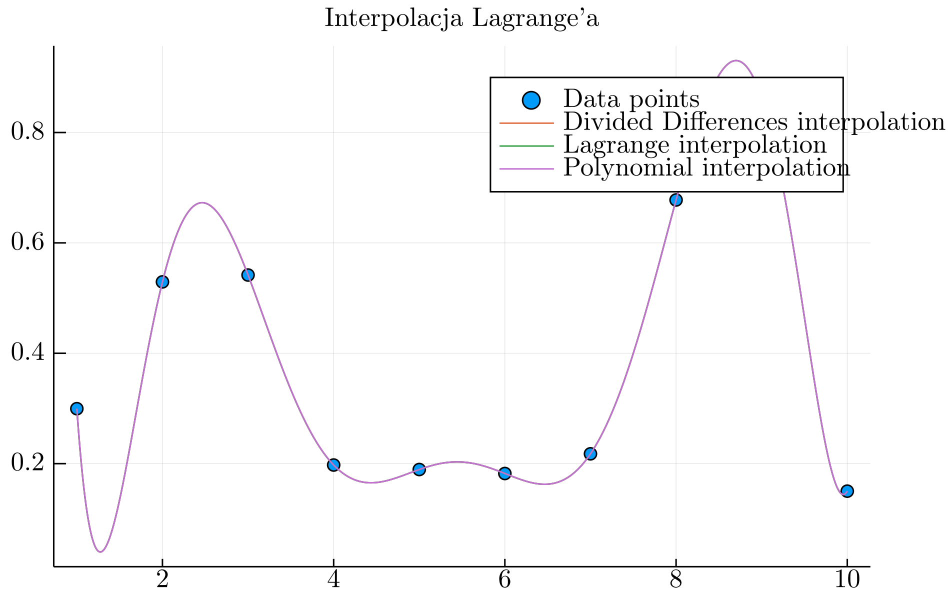
Porównanie interpolacji wielomianowej na jednym wykresie:

a. wielomian interpolacyjny Lagrange'a

b. metoda ilorazów róznicowych

c. interpolacja wielomianową z pakietu Polynomials

Rysunek 4



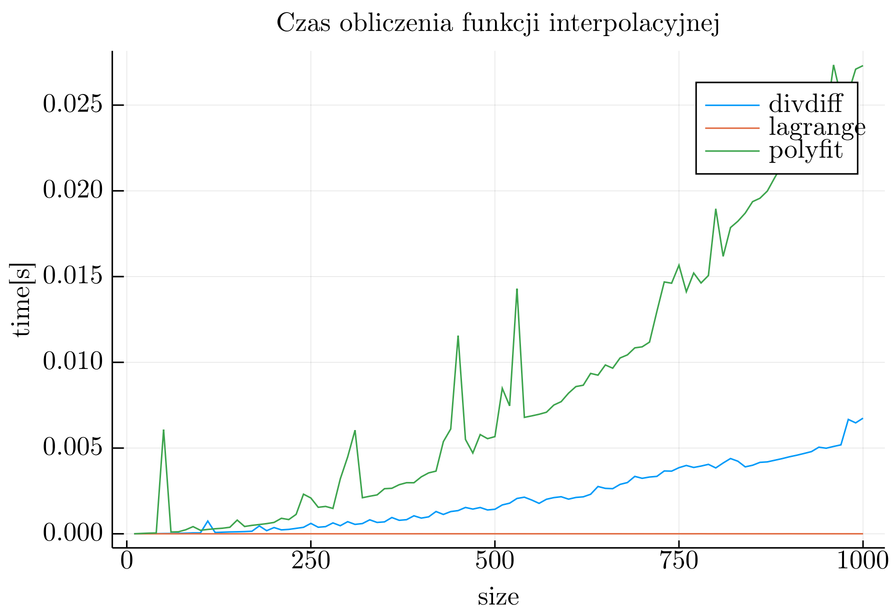
5 Jednoznaczność rozwiązania

Rozwiązania są identyczne dla różnych metod, ponieważ z twierdzenia o jednoznacznosci rozwiązania wynika, że istnieje dokładnie jeden wielomian $P_n(x)$ stopnia $\leqslant n$, przechodzący przez n punktów.

Zadanie 4

Porownanie metod poprzez pomiar czasu wykonania dla zmiennej ilości węzłow interpolacji.

Rysunek 5



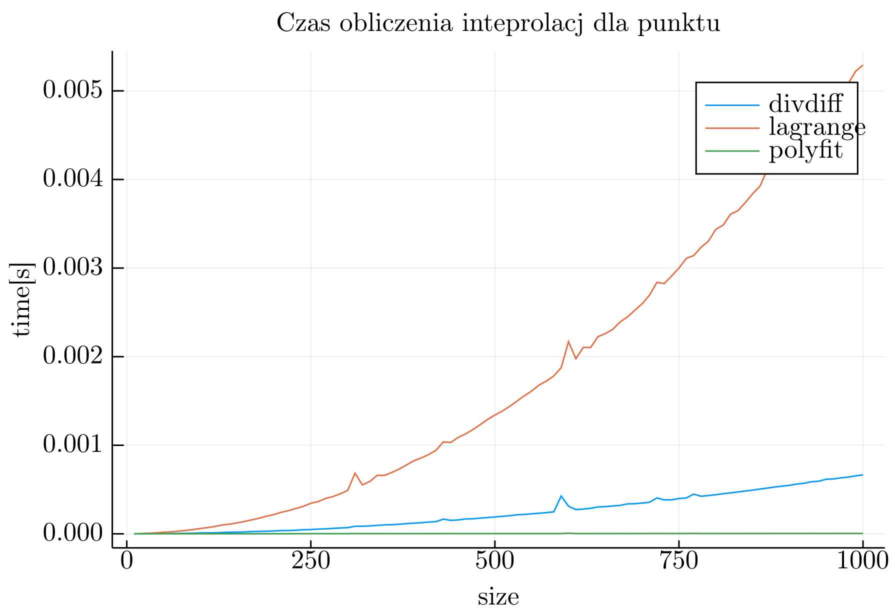
Na bazie wykresu czasu obliczenia funkcji interpolującej w zależności od liczby węzłów interpolacji można stwierdzić:

a. Metoda Lagrange'a nie dokonuje żadnych wstępnych obliczeń.

b. Metoda ilorazów różnicowych dokonuje obliczeń wstępnych.

c. Wbudowana funkcja polyfit dokonuje bardzo wielu obliczeń po wywołaniu, a czas działania jest dosyć nieregularny.

Rysunek 6



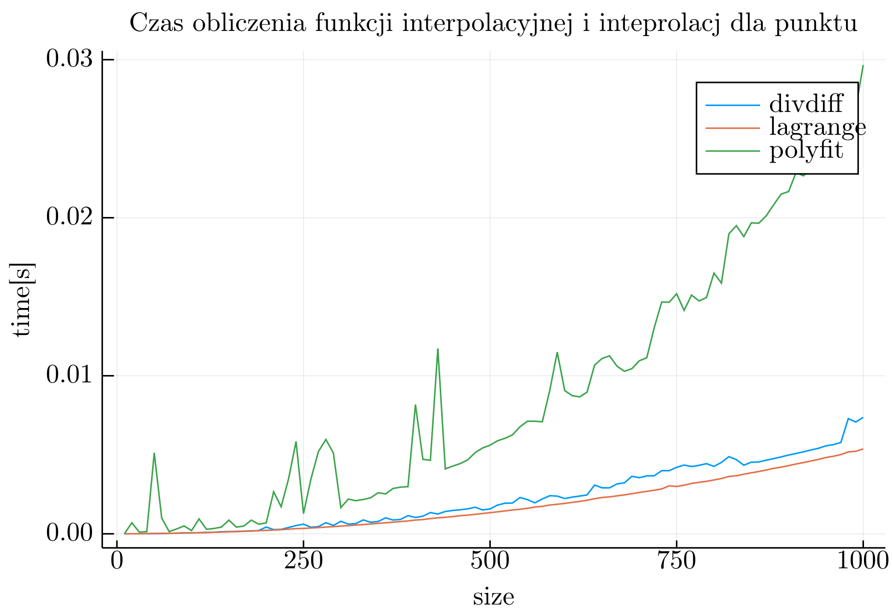
Na bazie wykres czasu obliczenia interpolacji punktu dla obliczonej już funkcji, w zależności od liczby węzłów interpolacji można stwierdzić:

a. Metoda Lagrange'a wszystkie obliczenia wykonuje dla konkretnego punktu, dlatego tym razem trwa zdecydowanie nadłużej, i bardzo szybko rośnie.

b. Metoda ilorazów różnicowych dokonuje pozostałych obliczeń, działa dużo szybciej od Lagrange'a.

c. Wbudowana funckja polyfit dokonuje obliczenia w zasadzie w czasie jednostkowym.

Rysunek 7



Wykres czasu obliczenia funkcji interpolacyjnej i obliczenie inteprolacji dla jednego punktu różnymi metodami.

Na wykresie widać, że metoda Lagrangea działa najszybciej, natomiast metoda Newtona jest tylko niewiele wolniejsza. Wbudowana funckja polyfit działa zdecydowanie wolniej i dosyć nieregularnie.

6 Wnioski

Jak widać, wybór metody ma istotny wpływ na czas obliczeń i powinien zależeć od tego, jak dużo będziemy obliczać wartości funkcji dla innych punktów. W przypadku obliczenia jednej wartości najszybsza jest metoda Lagrange'a. Jesli tych wartości jest więcej to lepiej sprawdzi się metoda Newtona która dokonuje części wspólnych obliczeń dla różnych punktów przy wyznaczaniu funkcji. Natomiast wpudowana funkcja polyfit dokonuje mozolnych obliczeń wzoru funkcji, ale później bardzo szybko zwraca wynik dla punktów. Dlatego sprawdzi się w sytuacjach, kiedy funkcja interpolacyjna jest bardzo czesto wywoływane.

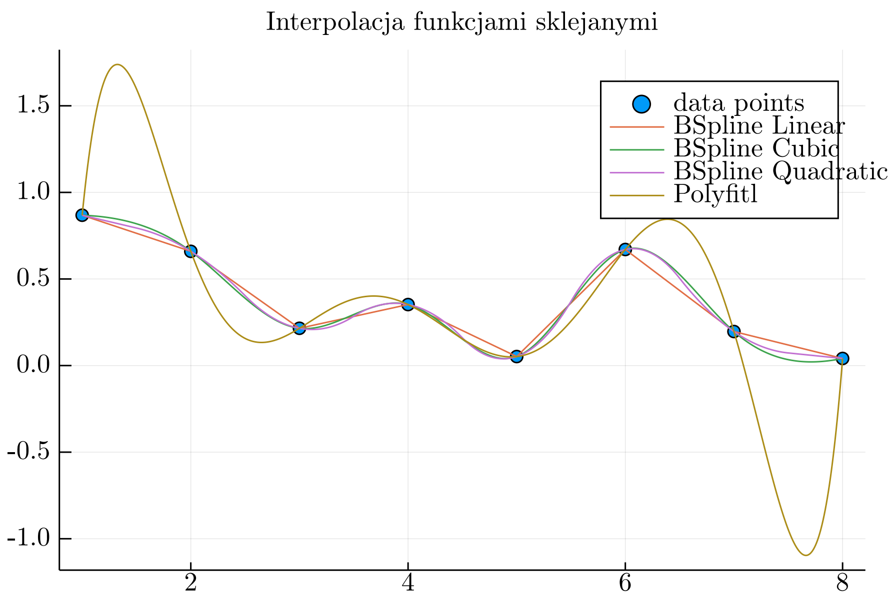
7 Obliczenia symboliczne w Julii

Udało mi się również dokonać inteprpolacji Lagrange'a przy pomocy obliczeń symbolicznych dostepnych w Julii, ale metoda była aż 100 krotnie wolniejsza od tej w zadaniu 1.

Zadanie 5

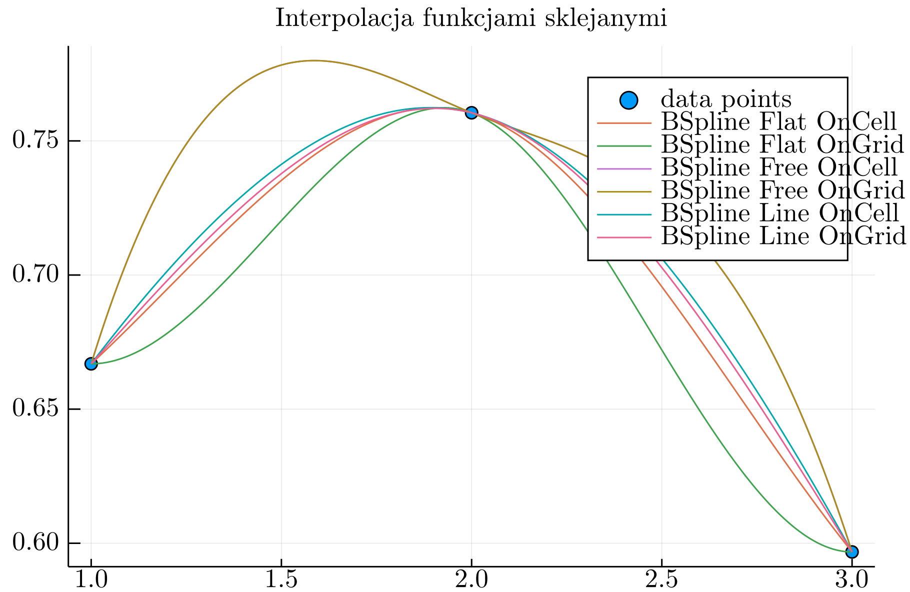
Interpolacja funkcjami sklejanymi

Rysunek 8



Porównianie interpolacji funkcjami sklejanymi. Interpolacja liniowa, kwadratowa i kubiczna w porównaniu do interpolacji wielomianami. Im funkcja jest niższego rzędu, tym jest bardziej płaska.

Rysunek 9

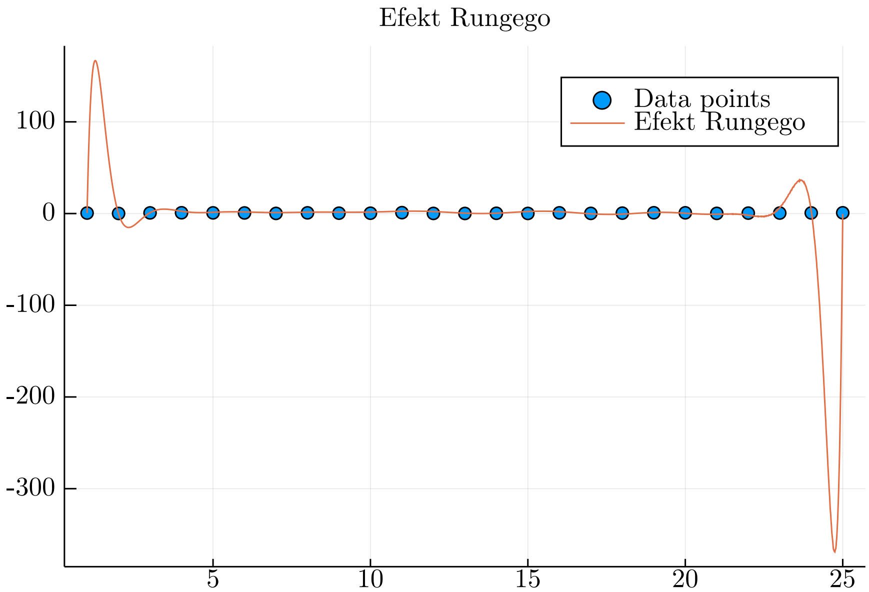


Porównanie interpolacji kubicznej dla różnych warunków brzegowych oraz różnych punktów siatki do których te warunki brzegowe są stosowane. Dla trzech punktów można uzyskać zupełnie różne wykresy.

Zdanie 6

Demonstracja efektu Rungego

Rysunek 10

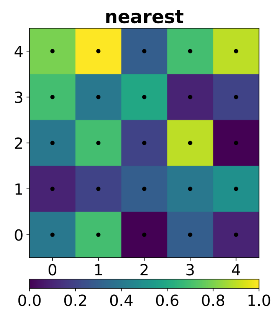


Zdanie 7

Algorytmy interpolacji stosowane w grafice komputerowej do zmiany wielkości obrazu

8 Interpolacja metodą najbliższego sąsiada

Polega na wiernym kopiowaniu sąsiednich pikseli

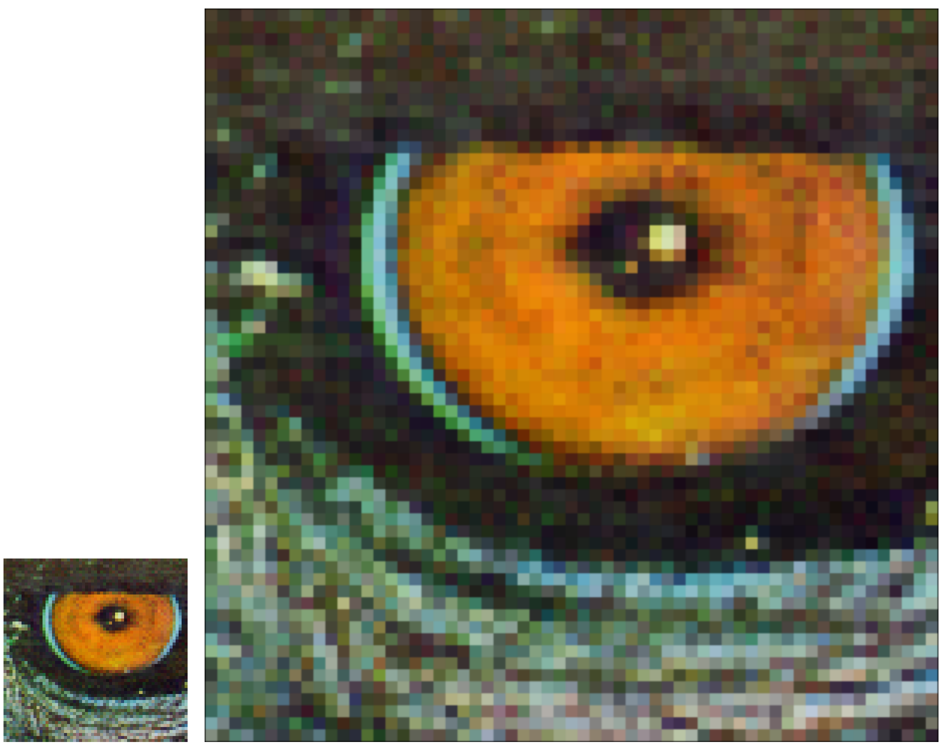


Rysunek 11

```
function resizeNearest(img,num)
    sizeX=length(img[:,1])*num
    sizeY=length(img[1,:])*num

    newImg = Array{RGB{Normed{UInt8,8}}}(undef, sizeX, sizeY)

    for x in 1:sizeX
        for y in 1:sizeY
            newImg[x,y]=img[div(x-1,num)+1,div(y-1,num)+1];
        end
    end
    newImg
end
```

Listing 4: Funkcja interpolująca metodą najbliższego sąsiada

Rysunek 12: Dziesięciokrotne powiększenie grafiki

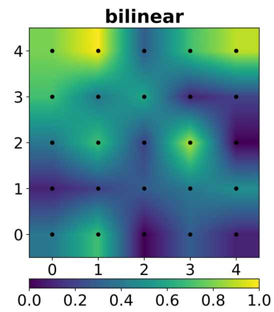


(a) Oryginał

(b) Powiększenie metodą najbliższego sąsiada

9 Interpolacja dwuliniowa

Kolor piksela jest obliczony na podstawie czterech sąsiednich pikseli, stykających się bokami.

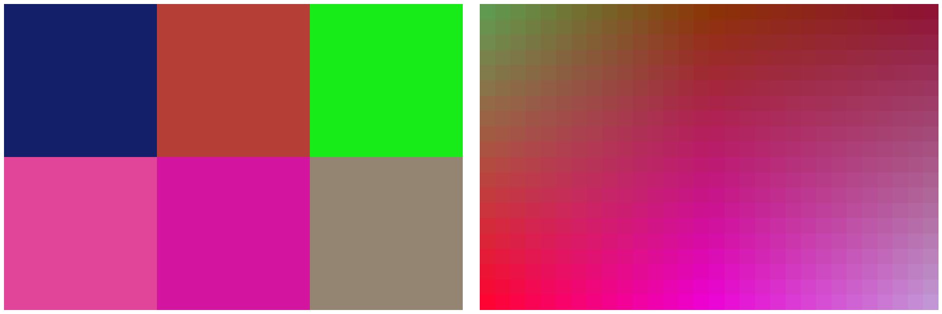


Rysunek 13

```
function resizeBilinear(img,num)
    sizeX=length(img[:,1])*num
    sizeY=length(img[1,:])*num
    newImg = fill(RGB{Normed{UInt8,8}}(0,0,0), sizeX, sizeY)

    for x in 1:sizeX
        leftX = floor(Int64,(x-1)/(sizeX-1)*(sizeX/num-1))+1
        rightX = ceil(Int64,(x-1)/(sizeX-1)*(sizeX/num-1))+1
        ratioX =
         ↪  1-Normed{UInt8,8}((x-1)/(sizeX-1)*(sizeX/num-1)+1-leftX)

        for y in 1:sizeY
            leftY =
             ↪  floor(Int64,(y-1)/(sizeY-1)*(sizeY/num-1))+1
            rightY =
             ↪  ceil(Int64,(y-1)/(sizeY-1)*(sizeY/num-1))+1

            leftRef = ratioX*img[leftX,leftY] +
             ↪  (1-ratioX)*img[rightX,leftY]
            rightRef = ratioX*img[leftX,rightY] +
             ↪  (1-ratioX)*img[rightX,rightY]

            ratioY = 1-
             ↪  Normed{UInt8,8}((y-1)/(sizeY-1)*(sizeY/num-1)+1-leftY)

            newImg[x,y] = ratioY*leftRef +
             ↪  (1-ratioY)*rightRef
        end
    end
    newImg
end
```

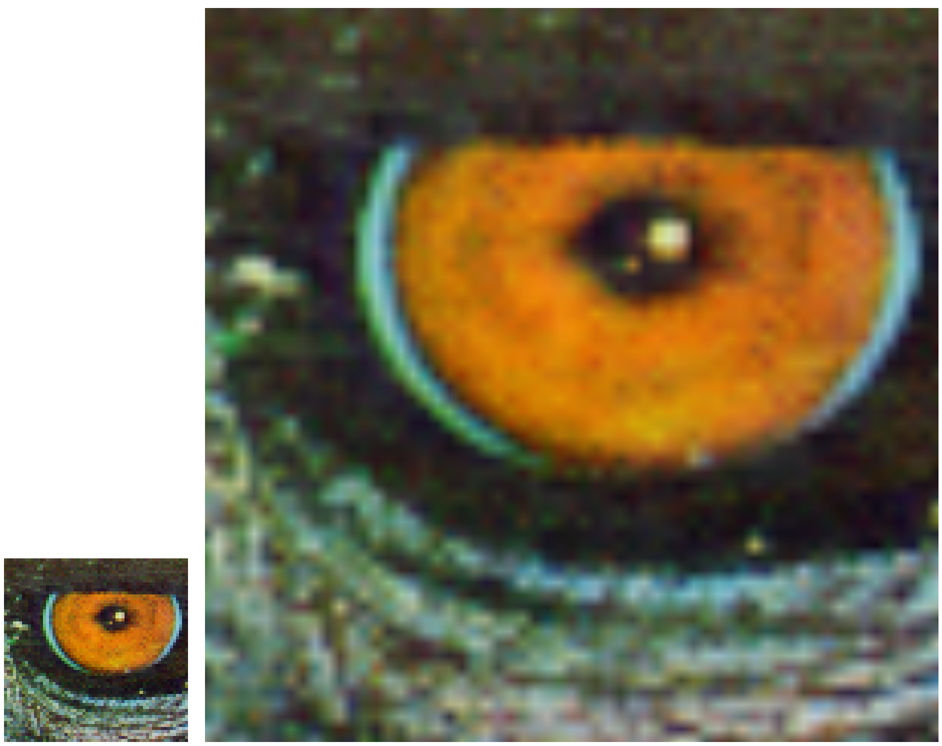
Listing 5: Funkcja interpolująca dwuliniowa

Rysunek 14: Zmiękczenie krawędzi



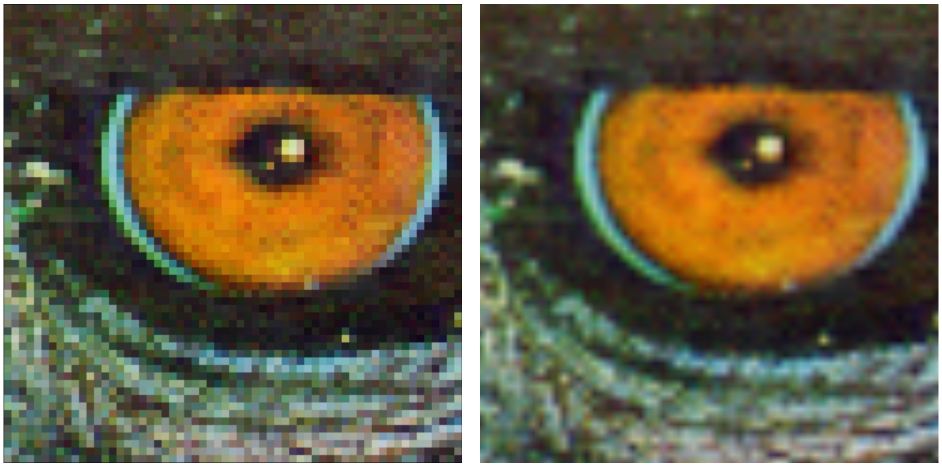
(a) Oryginał (b) Powiększenie inteprolacją dwuliniową

Rysunek 15: Dziesięciokrotnie powiększenie grafiki



(a) Oryginał (b) Powiększenie inteprolacją dwuliniową

Rysunek 16: Prównanie metod powiększania grafiki



(a) Powiększenie metodą sąsiada

(b) Powiększenie inteprolacją dwuliniową

10 Wnioski

Przy interpolacja metodą najbliższego sąsiada przy znacznym powiększeniu widać grupy identycznych pikseli, ale nie powoduje rozmycia kształtów. Pozwala zachować ostre krawędzie.

Interpolacja dwuliniowa daje łagodny, ale rozmyty obraz.