Katedra Informatyki

Metody obliczeniowe w nauce i technice

Laboratorium 6 - interpolacja sprawozdanie z ćwiczenia

Ludwik Ciechański

6 grudnia 2018

0 Przygotowanie

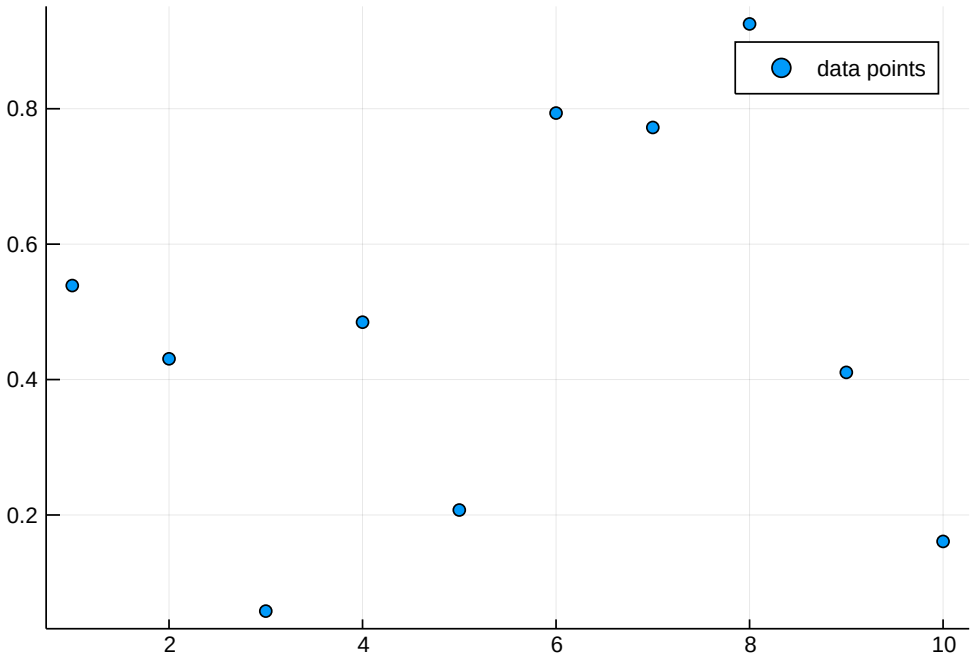
0.1 używane pakiety

```
using Plots
using Polynomials
using DataFrames
using Statistics
using Interpolations
```

0.2 przykładowe dane

```
# wylosowanie wezlow interpolacji
xs = 1:1:10
A = [rand() for x in xs]
# geste punkty do rysowania wykresow funkcji interpolujacych
xsf = 1:0.02:10
```

0.3 ilustracja



Rysunek 0: Wylosowane węzły interpolacji

1 Wielomian interpolacyjny Lagrange'a

1.1 wzory

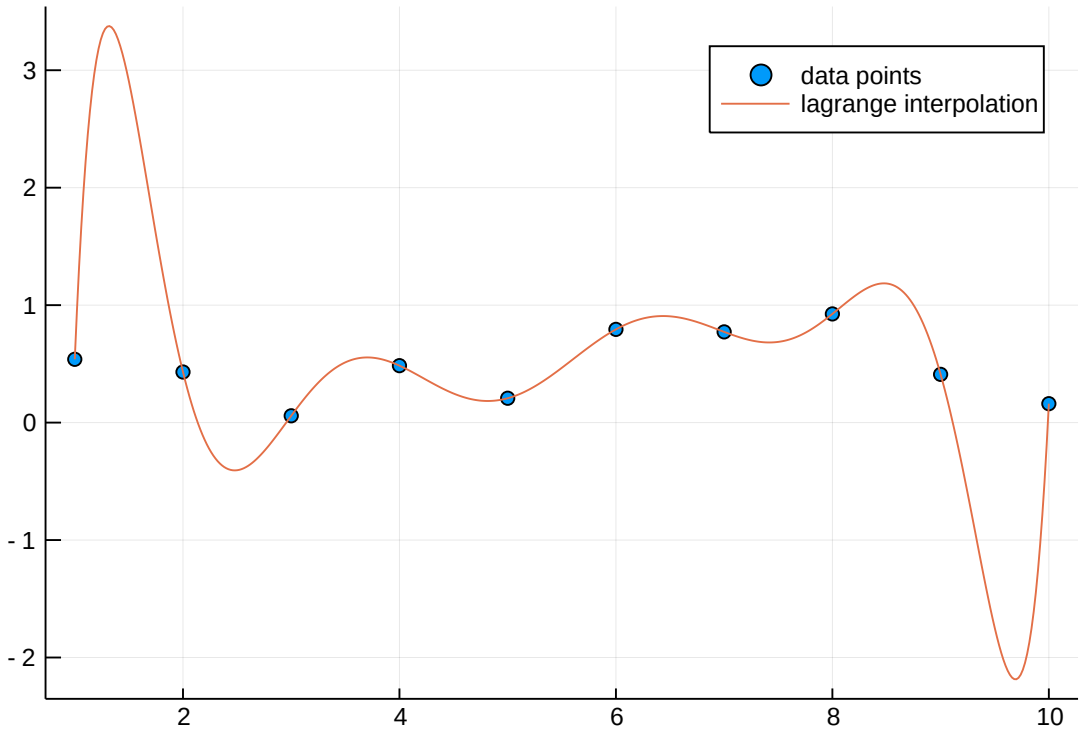
$$L_k(x) = \prod_{i=0, i\neq k}^{n} \frac{x - x_i}{x_k - x_i} \qquad P_n(x) = \sum_{k=0}^{n} L_k(x) f(x_k)$$

1.2 kod

```
function lagrange_interpolation(xs, A)
    n = size(A,1)
    P = Poly([0])
    for k = 1:n
        l = Poly([1.0])
        for i = 1:n
            if i != k
                l = l * poly([xs[i]]) / (xs[k] - xs[i])
            end
        end
        P += (l * A[k])
    end
    P
end
```

```
fit1 = lagrange_interpolation(xs, A)
B1 = [fit1(x) for x in xsf]
p1 = scatter(xs, A, label="data points")
plot!(xsf, B1, label="lagrange interpolation")
savefig(p1, "img/plot1.pdf")
```

1.3 wykres



Rysunek 1: Interpolacja wielomianem Lagrange'a

2 Metoda Newtona (ilorazów róznicowych)

2.1 wzory

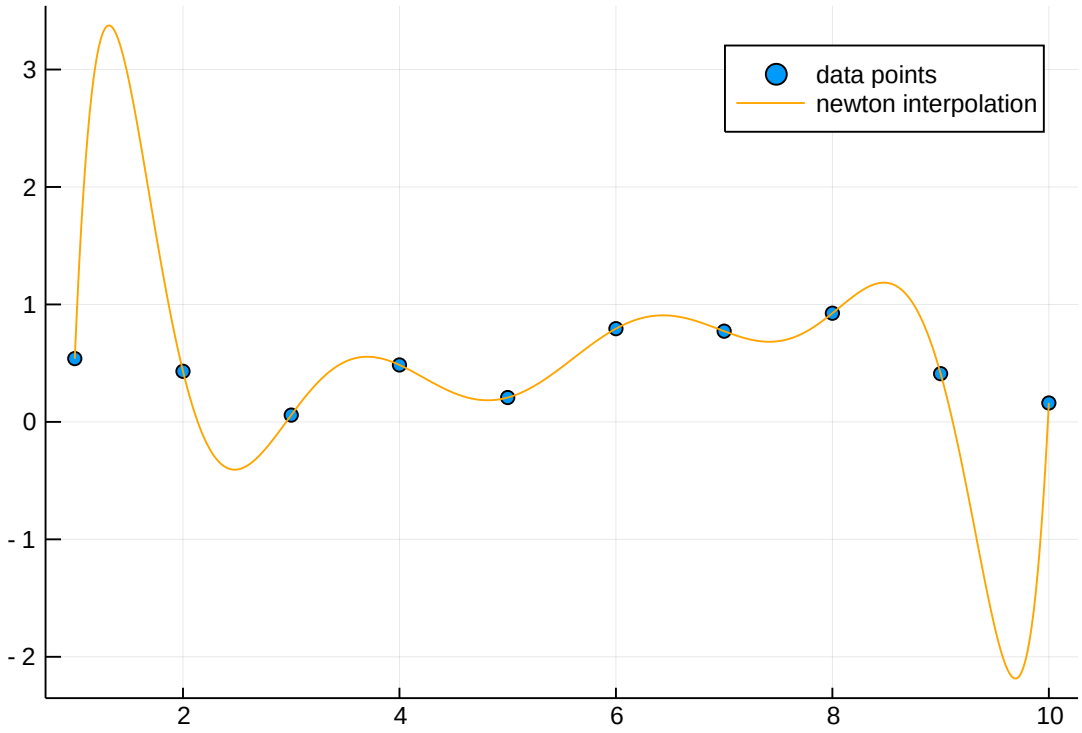
$$w(x) = a_0 + \sum_{i=1}^{n} a_i \prod_{j=0}^{i-1}(x - x_j)$$

2.2 kod

```
function newton_interpolation(xs, A, n)
    if n == 1
        Poly(float(A[1]))
    else
        prev = newton_interpolation(xs, A, n-1)
        p = A[n] - polyval(prev, xs[n])
        q = 1
        for i = 1:n-1
            q = q * (xs[n] - xs[i])
        end
        poly([xs[i] for i in 1:n-1]) * (p / q) + prev
    end
end
```

```
fit2 = newton_interpolation(xs, A, size(A,1))
B2 = [fit2(x) for x in xsf]
p2 = scatter(xs, A, label="data points")
plot!(xsf, B2, color=:orange, label="newton interpolation")
savefig(p2, "img/plot2.pdf")
```

2.3 wykres



Rysunek 2: Interpolacja metodą Newtona

3 Pakiet Polynomials

3.1 wybrane funkcje

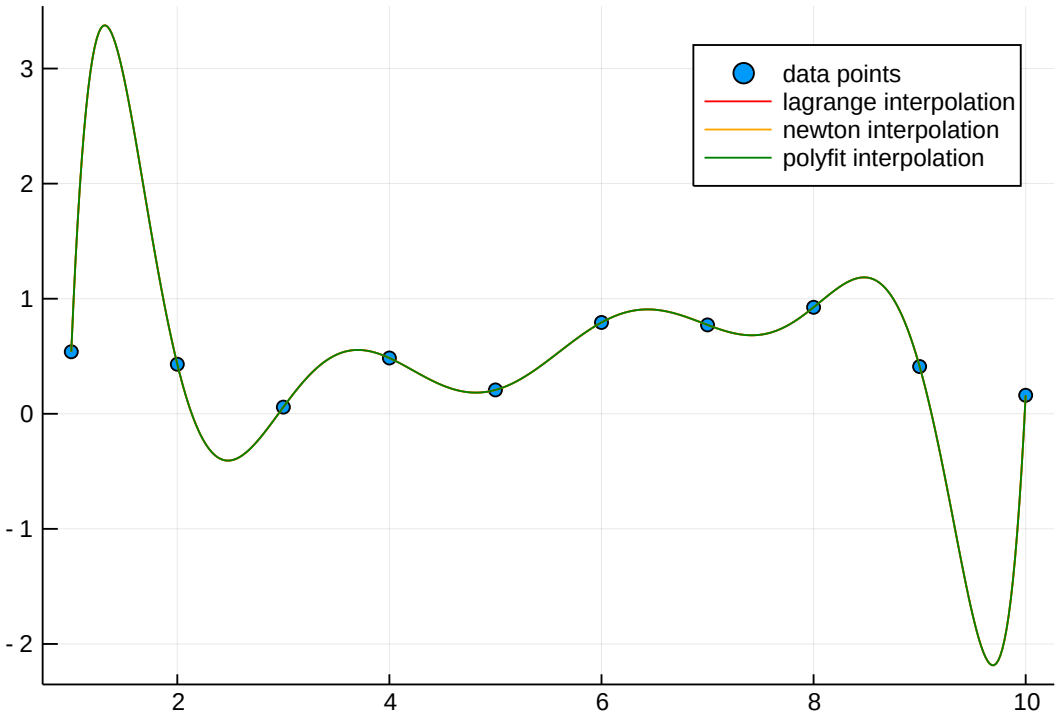
```
Poly(a::Vector) where {T<:Number}
    # tworzy wielomian na podstawie wspolczynnikow
poly(r::AbstractVector)
    # tworzy wielomian na podstawie pierwiastkow
polyval(p::Poly, x::Number)
    # oblicza wartosc wielomianu p w punkcie x
polyfit(x, y, n=length(x)-1)
    # dopasowuje wielomian (stopnia n) do punktow (x,y)
    # aproksymacja metoda najmniejszych kwadratow
```

Porównać wszystkie 3 wyniki interpolacji wielomianowej na jednym wykresie.

3.2 kod

```
fit3 = polyfit(xs, A)
B3 = [fit3(x) for x in xsf]
p3 = scatter(xs, A, label="data points")
plot!(xsf, B1, color=:red, label="lagrange interpolation")
plot!(xsf, B2, color=:orange, label="newton interpolation")
plot!(xsf, B3, color=:green, label="polyfit interpolation")
savefig(p3, "img/plot3.pdf")
```

3.3 wykres



Rysunek 3: Interpolacja polyfit z pakietu Polynomials

Co zauważamy? Niezależnie od zastosowanej metody, dla danych węzłów interpolacji wyznaczany jest ten sam wielomian.
Dlaczego? Otóż dla $n+1$ punktów istnieje *dokładnie* jeden wielomian n-tego stopnia interpolujący te punkty.

4 Porównanie metod

Porownać metody poprzez pomiar czasu wykonania dla zmiennej ilości węzłow interpolacji. Dokonać pomiaru 10 razy i policzyc wartość średnią oraz oszacować błąd pomiaru za pomocą odchylenia standardowego.

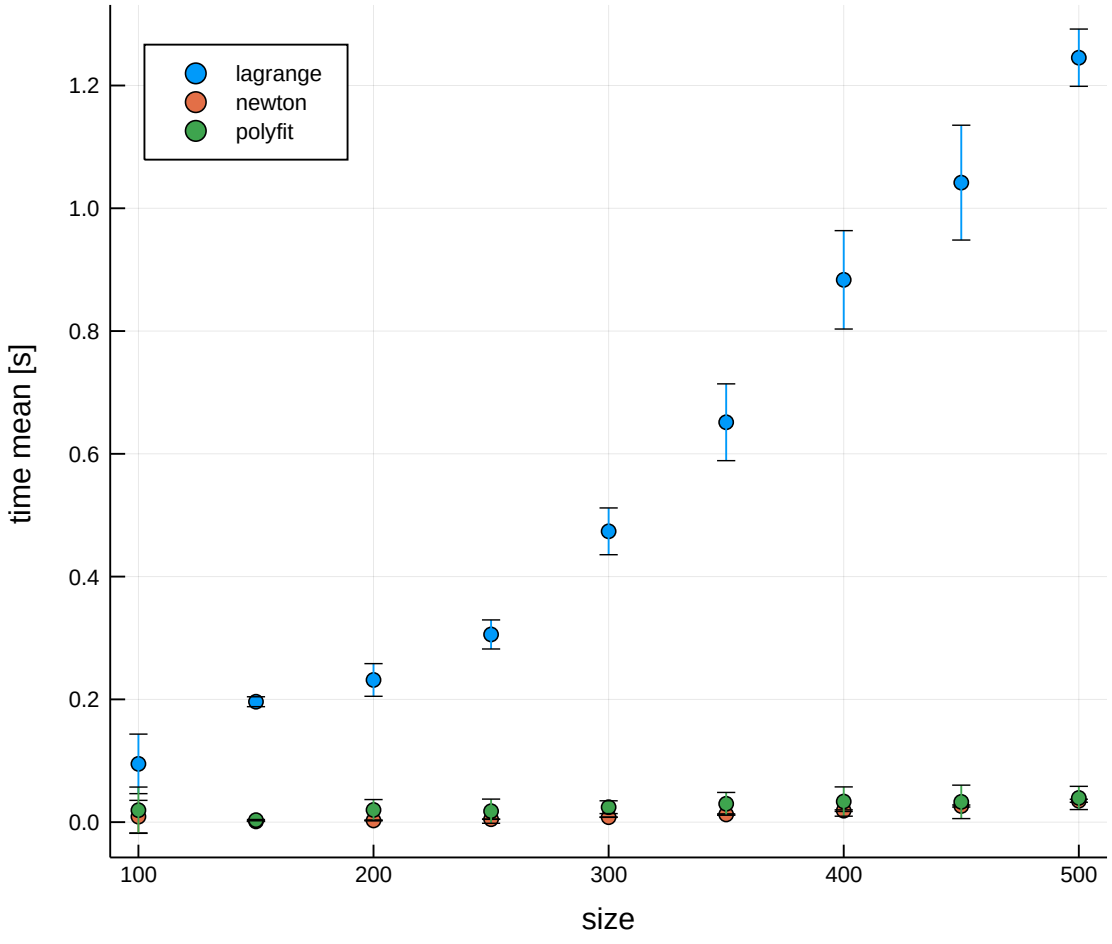
4.1 pomiary

```
df = DataFrame(size = Int64[], method = String[], time = Float64[])
for r in 100:50:500
    x = 0:1:r
    for i in 1:10
        y = rand(Int, r+1)
        push!(df, [r, "lagrange",@elapsed lagrange_interpolation(x, y)])
        push!(df, [r, "newton", @elapsed newton_interpolation(x, y, r)])
        push!(df, [r, "polyfit", @elapsed polyfit(x, y)])
    end
end
```

4.2 obliczenia

```
dfm = by(df, [1,2]) do grouped
    DataFrame(time_mean = mean(grouped[3]), time_std = std(grouped[3]))
end
```

4.3 wykres



Rysunek 4: Porównanie metod interpolacji wielomianowej

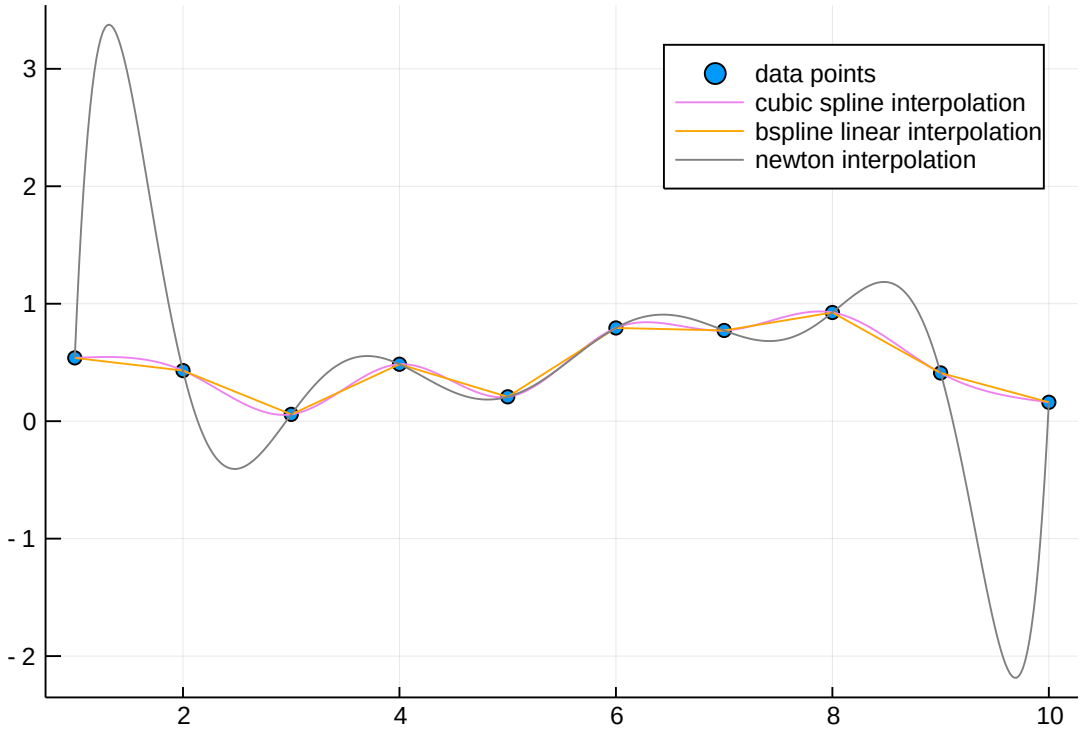
5 Interpolacja funkcjami sklejanymi

5.1 kod

```
#interpolacja szescienna
cubic = CubicSplineInterpolation(xs, A)
#interpolacja BSpline'ami
bspline = interpolate(A, BSpline(Linear()))

B5 = [cubic(x) for x in xsf]
B5b = [bspline(x) for x in xsf]
p5 = scatter(xs, A, label="data points", legend=:topright)
plot!(xsf, B5, color=:violet, label="cubic spline interpolation")
plot!(xsf, B5b, color=:orange, label="bspline linear interpolation")
plot!(xsf, B2, color=:grey, label="newton interpolation")
savefig(p5, "img/plot5.pdf")
```

5.2 wykres



Rysunek 5: Interpolacja funkcjami sklejanymi

6 Efekt Rungego

Widoczny jest na rysunku 5.
Polega na pogorszeniu jakości interpolacji wielomianowej, mimo zwiększenia liczby jej węzłów. Początkowo ze wzrostem liczby węzłów n przybliżenie poprawia się, jednak po dalszym wzroście n, zaczyna się pogarszać, co jest szczególnie widoczne na końcach przedziałów.
Takie zachowanie się wielomianu interpolującego jest zjawiskiem typowym dla interpolacji za pomocą wielomianów wysokich stopni przy stałych odległościach węzłów. Występuje ono również, jeśli interpolowana funkcja jest nieciągła albo odbiega znacząco od funkcji gładkiej. [źródło: wikipedia]

7 Podsumowanie

Jest kilka metod interpolacji wielomianowej. Dla danych węzłów interpolacji wszystkie zwracają ten sam wielomian (rysunki 1, 2 oraz 3). Metoda Lagrange'a jest najwolniejsza spośród badanych (rysunek 4).
Interpolacja funkcjami sklejanymi daje lepsze efekty niż wielomianowa, ponadto nie występuje wówczas efekt Rungego (rysunek 5).

Spis treści