Mownit Interpolacja

Magdalena Kozub

4.05.2019

1. **Napisać własna implementacje interpolacji wielomianowej stosujac wprost wzór na wielomian interpolacyjny Lagrange'a. Jezyk implementacji do wyboru (Julia, C). Przetestować swoja implementacje na wylosowanych wezłach interpolacji w wybranym przedziale. Narysować wykres wielomianu interpolacyjnego w tym przedziale wraz z wezłami interpolacji.**

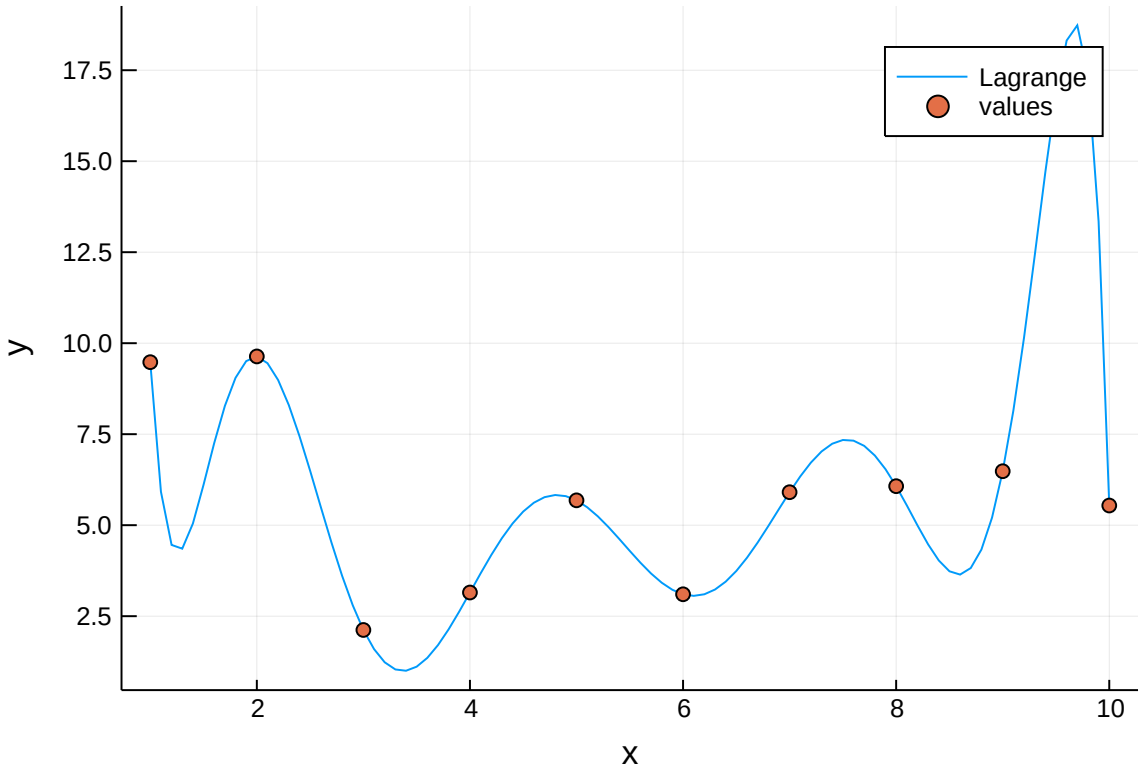
   Aby obliczyć współczynniki wielomianu interpolujacego stosujac metode Lagrange'a należy podstawić do wzoru na wielomian interpolacyjny Lagrange'a wartości danych punktów:

$$L_j(x) = \frac{d}{m} = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}, \quad P_n(x) = \sum_{j=0}^{n} L_j(x) * f(x_j)$$

   Na poczatku wybieramy punkty o losowych wartościach, a nastepnie dla każdego x z podanego przedziału obliczamy wartości wielomianu Lagrange'a.

```
#Interpolacja Lagrange'a
function Lagrange(X, Y, xi)
    n = length(X)
    result = 0
    for i=1:n
        term = Y[i]
        for j=1:n
            if i != j
                term=term*(xi - X[j])/(X[i]-X[j])
             end
        end
        result += term
    end
    return result
end
```

Listing 1: Kod Interpolacji Lagrange'a w jezyku Julia



Interpolacja Lagrange'a

2. **Zrobić to samo dla metody Newtona (metoda ilorazów różnicowych). Zadbać o to, żeby ilorazy wyliczać tylko raz dla danego zbioru wezłów interpolacji. Jezyk implementacji wybrać taki sam, jak w poprzednim punkcie. Narysować wykres wielomianu interpolacyjnego dla tych samych danych, co w poprzednim punkcie.**

Aby obliczyć współczynniki wielomianu interpolacyjnego metoda Newtona potrzebujemy obliczyć najpierw ilorazy skończone dla danych punktów, których wartości wyliczyliśmy wcześniej. Nastepnie korzystamy ze wzoru na interpolacyjny wzór Newtona:

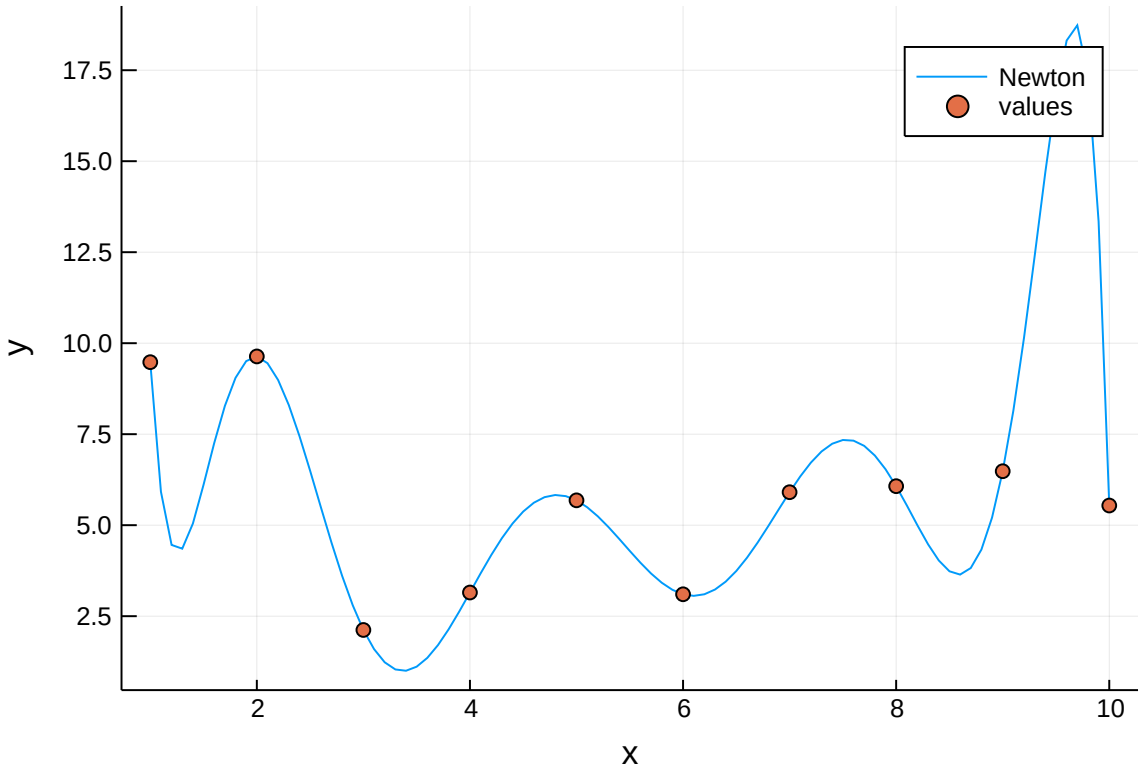
$$P_n(x) = f[x_0] + \sum_{k=1}^{n} [x_0, x_1, ..., x_k] * (x - x_0) * ... * (x - x_{k-1}))$$

K-ty iloraz różnicowy wyliczamy ze wzoru:

$$f[x_i, x_{i+1}, ..., x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, ..., x_{i+k}] - f[x_i, x_{i+1}, ..., x_{i+k-1}]}{x_{i+k} - x_i}$$

```
#Interpolacja Newtona
function diffs(X, Y, i)
    n = length(X)
    M = zeros(n)
    for j=1:n-i
        M[j]=(Y[j + 1, i + 1] - Y[j, i + 1])/(X[j + i] - X[j])
    end
    Y[:,2+i] = M
end
function Newton(X, Y)
    n = length(X)
    M=zeros(n, n+2)
    M[:,1] = X
    M[:,2] = Y
    for i=1:n
        diffs(X, M, i)
    end
    return M
end
function Value(Coefs, xi)
    sum = Coefs[1,2]
    n = length(Coefs[:,1]) - 1
    for i=1:n
        prod = Coefs[1, i + 2]
        for j=1:i
            prod *= (xi - Coefs[j,1])
        end
        sum += prod
    end
    return sum
end
```

Listing 2: Kod Interpolacji Newtona w jezyku Julia

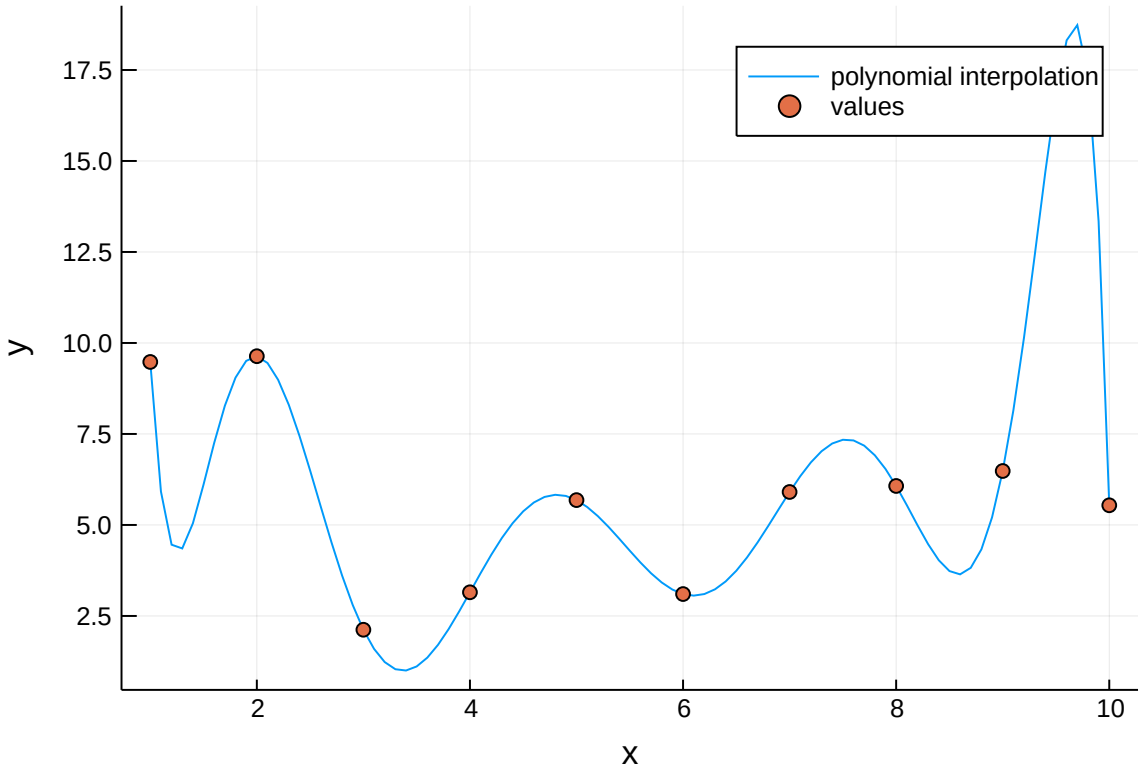


Interpolacja Newtona

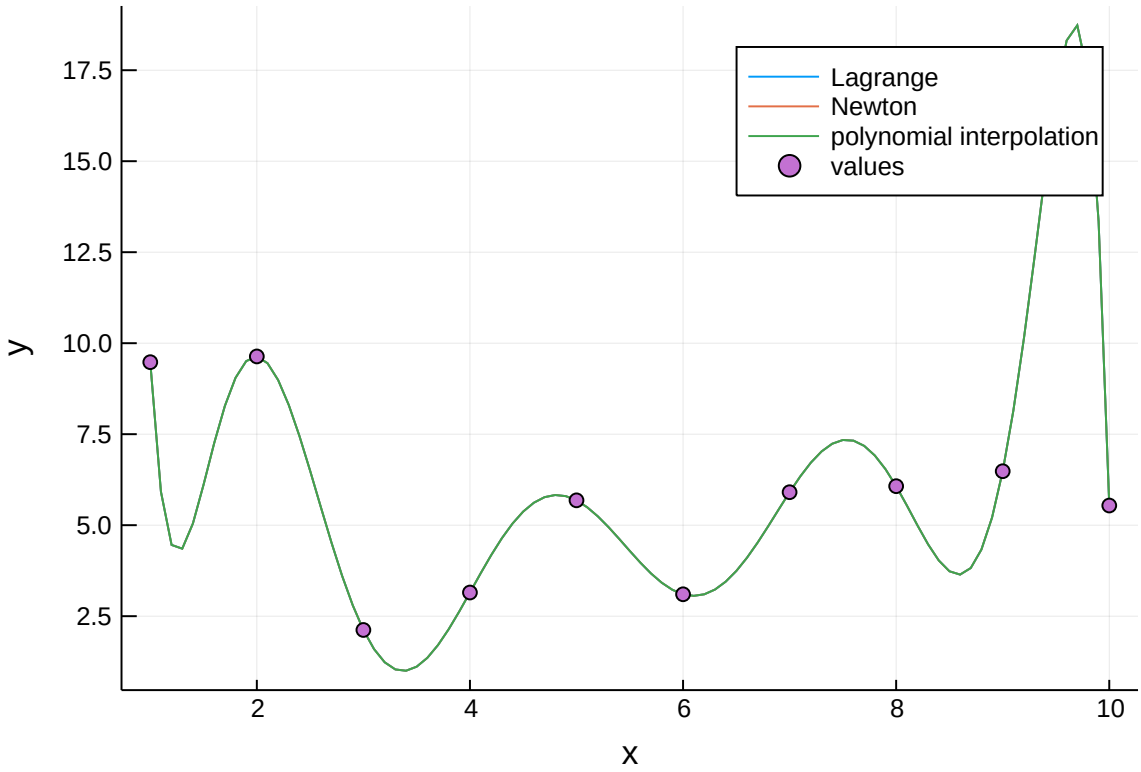
3. **Zastosować interpolacje wielomianowa z pakietu Polynomials (jeśli wybraliśmy Julie) albo z funkcji gsl_interp_polynomial z pakietu GSL (jeśli wybraliśmy C) do tych samych danych, co w poprzednich punktach. Porównać wszystkie 3 wyniki interpolacji wielomianowej na jednym wykresie. Co zauważamy? Dlaczego?**

```
#Interpolacja z pakietu Polynomials
fit1=polyfit(x, f_x)
p_ys = [fit1(x) for x in xs]

plot(xs, p_ys, label="polynomial interpolation", xlabel="x", ylabel="y")
w1 = scatter!(x, f_x, label="values", xlabel="x", ylabel="y")
```

Listing 3: Kod Interpolacji z pakietu Polynomials w jezyku Julia



Interpolacja z pakietu Polynomials



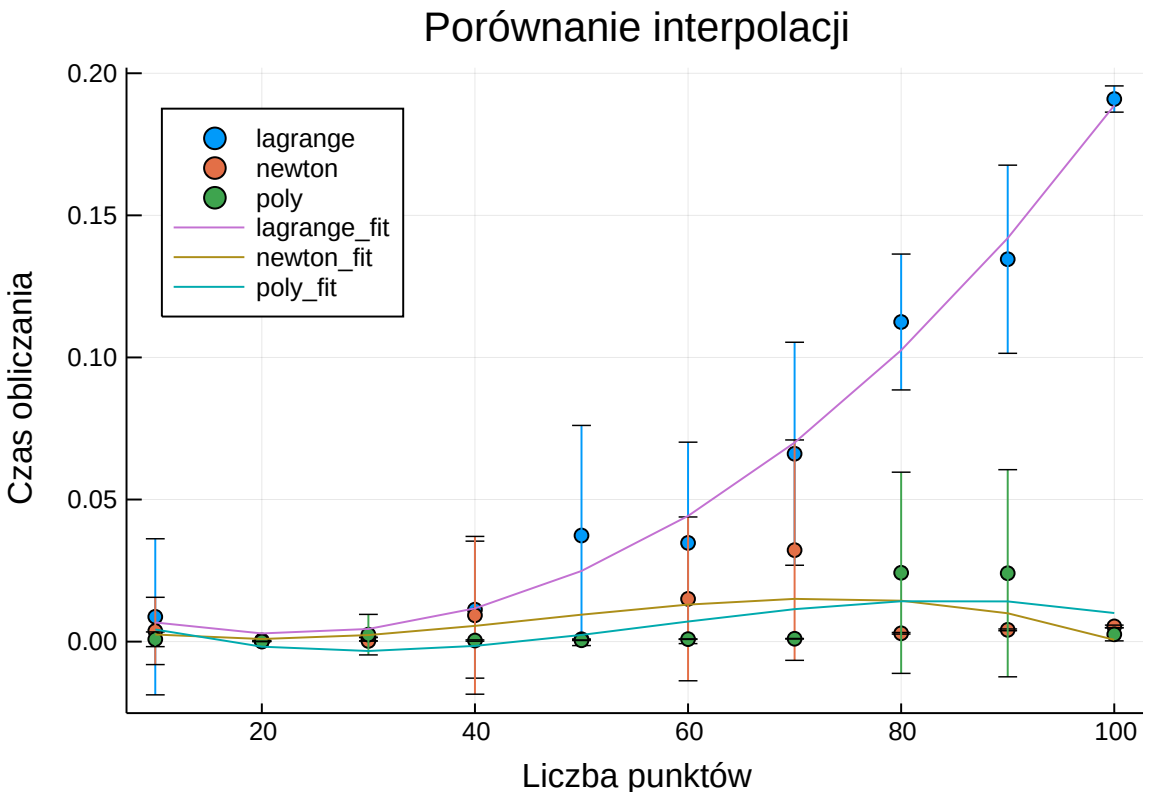
Porównanie 3 sposobów interpolacji

Jak można zauważyć na powyższym wykresie wielomiany interpolujace sa dokładnie identyczne bez wzgledu na to, która metoda je wyliczyliśmy. Zgadza sie to z twierdzeniem o jednoznaczności wielomianu interpolacyjnego, który mówi, że istnieje dokładnie jeden taki wielomian dla podanych wezłów.

4. **Porównać metody poprzez pomiar czasu wykonania dla zmiennej ilości wezłów interpolacji. Dokonać pomiaru 10 razy i policzyć wartość średnia oraz oszacować bład pomiaru za pomoca odchylenia standardowego. Narzedzie do analizy danych do wyboru (Julia, R).**

```
using DataFrames
using Polynomials
using Statistics

function NewtonWrapper(x, f_x, x_s)
    dif = Newton(x, f_x)
    [Value(dif, xi) for xi in x_s]
end
function PolyWrapper(x, f_x, x_s)
    fit1=polyfit(x, f_x)
    [fit1(x) for x in x_s]
end

df1=DataFrame(size = Int[], lagrange = Float64[], newton = Float64[], poly = Float64[])
for i=10:10:100
    for j=1:10
        x = 1:i
        f_x = rand(i)
        xs = 1:0.1:i

        lagrange = @elapsed [Lagrange(x, f_x, xi) for xi in xs]
        newton = @elapsed NewtonWrapper(x, f_x, xs)
        poly = @elapsed PolyWrapper(x, f_x, xs)

        push!(df1, [i lagrange newton poly])
    end
end

df2 = by(df1, :size, df->DataFrame(lagrange_mean=mean(df[:lagrange]),
        lagrange_std=std(df[:lagrange]),
        newton_mean=mean(df[:newton]), newton_std=std(df[:newton]),
        poly_mean=mean(df[:poly]),
        poly_std=std(df[:poly])))
lagrange_fit=polyfit(df2[:size], df2[:lagrange_mean], 3)
newton_fit=polyfit(df2[:size], df2[:newton_mean], 3)
poly_fit=polyfit(df2[:size], df2[:poly_mean], 3)

w1 = scatter(df2[:size], [df2[:lagrange_mean], df2[:newton_mean], df2[:poly_mean]],
    yerr=[df2[:lagrange_std] df2[:newton_std]
    df2[:poly_std]], label=["lagrange" "newton" "poly"],
    legend=:topleft)

plot!(df2[:size], polyval(lagrange_fit, df2[:size]), label="lagrange_fit")
plot!(df2[:size], polyval(newton_fit, df2[:size]), label="newton_fit")
plot!(df2[:size], polyval(poly_fit, df2[:size]), label="poly_fit")
```

Listing 4: Kod wyliczajacy czas działania każdej z interpolacji



Porównanie czasów działania 3 rodzajów interpolacji

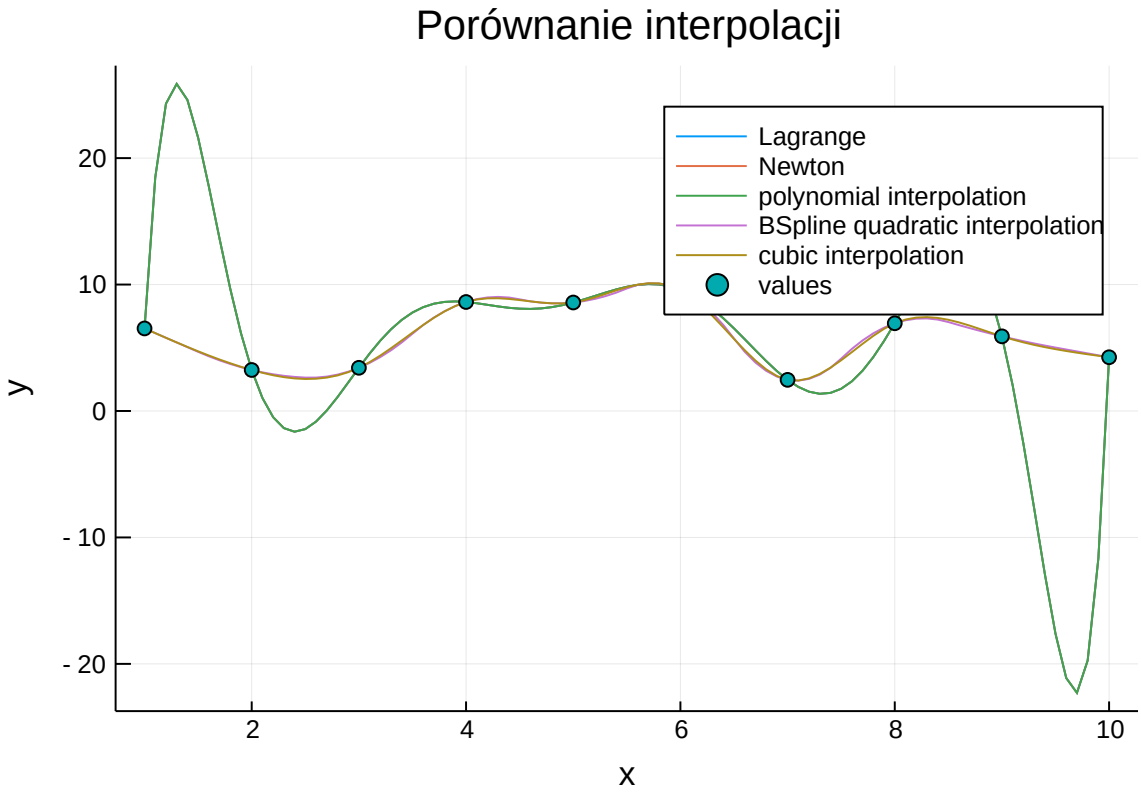
Najwolniejsza interpolacja okazała sie być interpolacja Lagrange'a, szczególnie dla wiekszej liczby punktów. Interpolacja Newtona i z pakietu Polynomials osiagały podobne wyniki.

5. **Poeksperymentowac z interpolacja funkcjami sklejanymi (minimum dwie różne funkcje sklejane), narysować wykresy i porównać z wykresami interpolacji wielomianowej.**

   Porównałam dwie funkcje sklejane - kwadratowa oraz kubiczna (sześcienna).

```
using Interpolations

x = 1:10
f_x = rand(10) * 10
xs = 1:0.1:10
l_ys = [Lagrange(x, f_x, xi) for xi in xs]
dif = Newton(x, f_x)
n_ys = [Value(dif, xi) for xi in xs]
fit1=polyfit(x, f_x)
p_ys = [fit1(x) for x in xs]
itp = interpolate(f_x, BSpline(Quadratic(Line(OnCell()))))
q_ys =[itp(xi) for xi in xs]
interp_cubic = CubicSplineInterpolation(x, f_x)
c_ys = [interp_cubic(xi) for xi in xs]

plot(xs, l_ys, label="Lagrange")
plot!(xs, n_ys, label="Newton")
plot!(xs, p_ys, label="polynomial interpolation")
plot!(xs, q_ys, label="BSpline quadratic interpolation")
plot!(xs, c_ys, label="cubic interpolation")
w1 = scatter!(x, f_x, label="values")
```

   Listing 5: Kod Interpolacji oraz funkcji sklejanych w jezyku Julia



Porównanie wielomianów interpolacyjnych i funkcji sklejanych

Jak można zauważyć używajac funkcji sklejanych nie wystepuje efekt Rungego, który pojawił sie na końcach przedziału dla wielomianów interpolacyjnych. Interpolacja wielomianami jednak dość dobrze przybliża wartości w środku przedziału. Użycie funkcji sklejanej kwadratowej daje w wyniku dokładny wielomian interpolacyjny, jednak mniej dokładny i mniej gładki niż gdy użyjemy sześciennej funkcji sklejanej, dlatego w praktyce cześciej wykorzystujemy funkcje kubiczne.

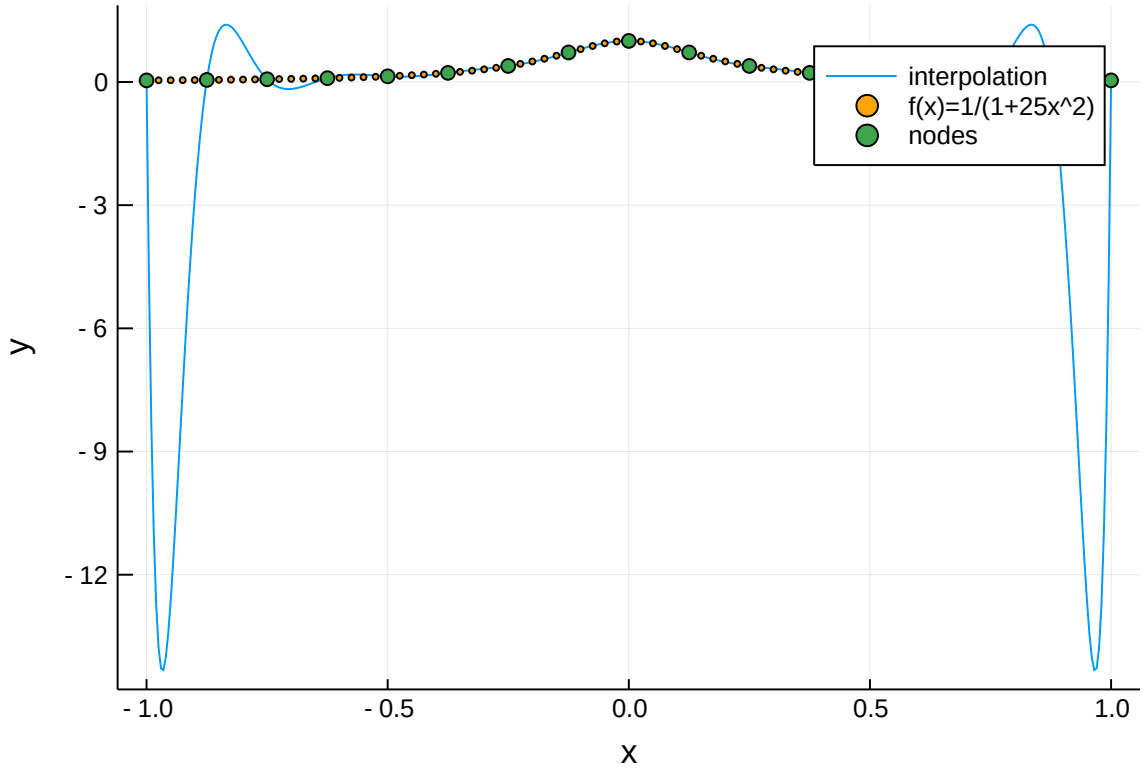
6. **Zademonstrować efekt Rungego.**

Efekt Rungego polega na obniżeniu jakości interpolacji pomimo zwiekszenia liczby wezłów, tzn. poczatkowo wraz ze wzrostem liczby wezłów jakość interpolacji rośnie (bład maleje), ale potem pogarsza sie (szczególnie na końcach przedziału i dla wielomianów interpolujacych wysokich stopni). Wystepuje wtedy kiedy używamy interpolacji wielomianowej i jednocześnie nakładamy warunek równoodległości wezłów.

```
function f(x)
    return 1/(1+25*x^2)
end

x = -1:0.125:1
f_x = [f(xi) for xi in x]

xs = -1:0.005:1
fit1 = polyfit(x, f_x)
p_ys = [fit1(x) for x in xs]
```

```
f_xs = -1:0.025:1
f_ys = [f(x) for x in f_xs]

println(fit1)

plot(xs, p_ys, label="interpolation")
scatter!(f_xs, f_ys, label="f(x)=1/(1+25x^2)", color="orange", marker = (:dot, 2))
w1 = scatter!(x, f_x, label="nodes")
```

Listing 6: Kod efektu Rungego w jezyku Julia



Wykres przedstawiajacy efekt Rungego