Zadanie 9:

8. Znaleźć wartości funkcji

$$f(x) = \frac{1}{1+5x^2} \tag{7}$$

w punktach -1, $-1+\frac{1}{32}$, $-1+\frac{2}{32}$, $\ldots$, $1-\frac{1}{32}$, 1, a następnie skonstruować wielomian interpolacyjny Lagrange'a oparty na tych węzłach i wartościach funkcji (7) w tych węzłach. Narysować wykres wielomianu interpolacyjnego.

9 O. Skonstruować naturalny splajn kubiczny dla funkcji i węzłów z zadania 8. Sporządzić jego wykres.

Kod w języku C++:

```cpp
#include<iostream>
#include<array> //std::array
#include<math.h> //std::sqrt
#include<fstream> //std::ofstream

using std::cout;
using std::endl;

#define Vector std::array<long double,65>
#define VectorSpline std::array<long double, 63>
#define Matrix std::array<std::array<long double,63>, 63>

void fillXY(Vector& x, Vector& y);
void fillSystem(Matrix& m, VectorSpline& f, Vector& y);
void CholeskyTriDiag(Matrix& m, Vector& ksi, VectorSpline& f);
void writeToFile(const Vector& x, const Vector& y, const char* f);
void constructPoly(Vector& x, Vector& y, Vector& ksi);

int main(int argc, char const *argv[]){
    Vector x; //x-y funkcji zmieniajace sie o 1/32
    Vector y ; //wartosci funkcji 1/(1+5*x*x)
    Matrix m = {}; //macierz trojdiagonalna do ukladu rownan na wielkosci ksi
    Vector ksi = {}; //wektor drugich pochodnych wyrazenia interpolacyjnego
    VectorSpline f = {}; //wektor potrzebny do obliczenia wektora ksi
    //wypelnianie
    fillXY(x,y);
    fillSystem(m,f,y);
    //obliczanie ksi za pomoca faktoryzacji choleskiego
    CholeskyTriDiag(m,ksi,f);
    //zapisywanie do pliku rzeczywistej funkcji
    writeToFile(x,y,"function.txt");
    //zapisuje do pliku wartosci obliczone za pomoca spline'u
    constructPoly(x,y,ksi);
    return 0;
}


//wypelnia wektory wartosciami x i wartosciami funkcji odpowiednio
void fillXY(Vector& x, Vector& y){
```

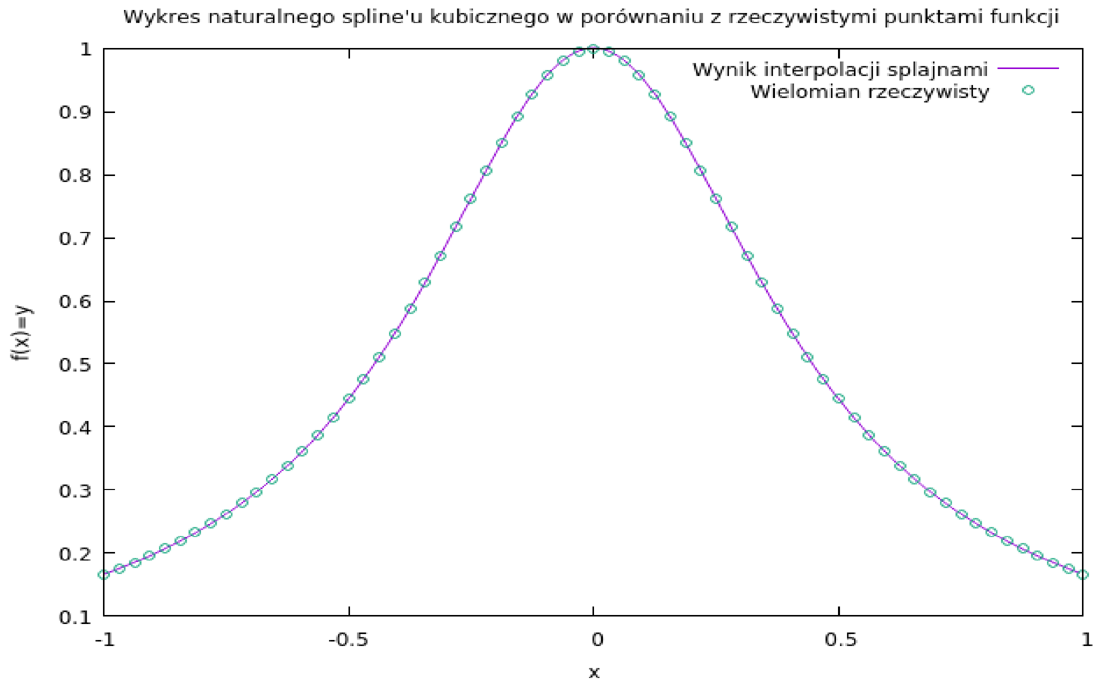
```cpp
    int i=0;
    for(long double j=-1; j<=1; j=j+1.0/32){
        x[i]=j;
        y[i]=(1/(1+5*j*j));
        i++;
    }
}


//wypelnia elementy do ukladu rownan macierzowych
void fillSystem(Matrix& m, VectorSpline& f, Vector& y){
    //wezly interpolacji sa rownoodlegle, krok wynosi h=1/32
    long double h=1.0/32;
    for(int i=0; i<m.size(); i++){
        f[i]=y[i]-2*y[i+1]+y[i+2];
        f[i]=f[i]*(6/(h*h));
        if(i==0){
            m[0][1]=1;
        }
        else if(i==m.size()-1){
            m[i][i-1]=1;
        }
        else{
            m[i][i-1]=1;
            m[i][i+1]=1;
        }
        m[i][i]=4;
    }
}


//faktoryzacja Choleskiego z modyfikacja do macierzy symetrycznej
void CholeskyTriDiag(Matrix& m, Vector& ksi, VectorSpline& f){
    Matrix C;
    C[0][0]=sqrt(m[0][0]);
    for(int i=1;i<m.size();i++){
        C[i][i-1]=m[i][i-1]/C[i-1][i-1];
        C[i][i]=sqrt((m[i][i]-C[i][i-1]*C[i][i-1]));
    }
    VectorSpline y;
    //forwardsubstitution Cy=b, gdzie y=CT*x
    y[0]=f[0]/C[0][0];
    for(int i=1;i<y.size();i++)
        y[i]=(f[i]-y[i-1]*C[i][i-1])/C[i][i];
    //backsubstitution CTx=y
    ksi[ksi.size()-2]=y[y.size()-1]/C[y.size()-1][y.size()-1];
    for(int i=f.size()-1;i>=1;i--)
        ksi[i]=(y[i]-C[i+1][i]*ksi[i+1])/C[i][i];
}


//zapisuje wektor x i wartosci funkcji y do pliku f
void writeToFile(const Vector& x, const Vector& y, const char* f){
    std::ofstream file;
    file.open(f);
    for(int i=0; i<x.size(); i++){
        file<<x[i]<<" "<<y[i]<<endl;
```

```cpp
    }
    file.close();
}



//tworzy wielomian dla kazdego przedzialu i wylicza wartosci ktore nastepnie
zapisuje do pliku spline.txt
void constructPoly(Vector& x, Vector& y, Vector& ksi){
    //do zapisu pliku
    std::ofstream file;
    file.open("spline.txt");
    int j=0;
    long double h=1.0/32;
    long double A,B,C,D;
    for(long double i=-1; i<=1; i=i+0.01){
        //gdy wyszlismy z przedzialu nalezy przejsc do nastepnego
        if(i>x[j+1]){
            j++;
            i=i-0.01;
        }
        else{
            //obliczanie wspolczynnikow
            A=(x[j+1]-i)/h;
            B=(i-x[j])/h;
            C=(A*A*A-A)*(h*h)/6;
            D=(B*B*B-B)*(h*h)/6;
            //zapis do pliku
            file<<i<<" "<<A*y[j]+B*y[j+1]+C*ksi[j]+D*ksi[j+1]<<endl;
        }
    }
    file.close();
}
```

Skrypt gunplota do stworzenia wykresu:

```
set title "Wykres naturalnego spline'u kubicznego w porównaniu z rzeczywistymi punktami funkcji"
set xlabel "x"
set ylabel "f(x)=y"
set xrange [-1: 1]
plot 'spline.txt' with lines title 'Wynik interpolacji splajnami' ,
'function.txt' title 'Wielomian rzeczywisty' with points pointtype 6
```

Wynik działania programu:



Opis metody:
Naturalne splajny kubiczne to specjalny rodzaj splajnów dla którego drugie pochodne wyrażenia interpolacyjnego(nie funkcji interpolowanej) $\xi_1=0$ i $\xi_n = 0$. Aby je obliczyć musimy skorzystać z faktu, że pierwsza pochodna $y_j(x)$ w prawym krańcu przedziału musi być równa $y_{j+1}(x)$ w lewym krańcu przedziału. W podanym zadaniu możemy zauważyć, że węzły interpolacji są równoodległe i $x_{j+1}-x_j=1/32$ co upraszcza nasz układ równań do postaci:

$$\begin{pmatrix} 4 & 1 & 0 & 0 & \dots & 0 \\ 1 & 4 & 1 & 0 & \dots & 0 \\ 0 & 1 & 4 & 1 & \dots & 4 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & 4 & 1 \\ 0 & 0 & \dots & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} \xi_2 \\ \xi_3 \\ \xi_4 \\ \dots \\ \xi_{n-2} \\ \xi_{n-1} \end{pmatrix} = \frac{6}{h^2} \begin{pmatrix} f_1 - 2f_2 + f_3 \\ f_2 - 2f_3 + f_4 \\ \dots \\ \dots \\ f_{n-3} - 2f_{n-2} + f_{n-1} \\ f_{n-2} - 2f_{n-1} + f_n \end{pmatrix}$$

W powyższym układzie równań można zastosować faktoryzację Choleskiego, aby obliczyć nieznane wartości ξ.
Następnie posługując się poniższymi wzorami możemy obliczyć wartość wielomianu interpolacyjnego w odpowiednim przedziale.

$$A = \frac{x_{j+1} - x}{h}$$

$$B = \frac{x - x_j}{h}$$

$$C = \frac{1}{6} * (A^3 - A) * (h^2)$$

$$D = \frac{1}{6} * (b^3 - B) * (h^2)$$

$$y_j(x) = A * f_j + B * f_{j+1} + C * \xi_j + D * \xi_{j+1}$$