Wojciech Ładyga - zadanie 11

Język technologia: c++, GSL

Aby wyliczyć jawnie współczynniki wielomianu interpolacyjnego Lagrange's na podstawie zadanych węzłów i funkcji należało wyliczyć wartości funkcji na podstawie podanych węzłw, a następnie dekompozycją LU rozwiązać taki układu i wyniki zapisać do wektora a przechowującego współczynniki wielomianu. Aby narysować wykres zastosowałem polecenia

```
./a.out > interp.dat
graph -T ps < interp.dat > interp.ps
```

Wykres został wygenerowany na przedziale <-1,1> przeskakując od -1 do 1 o 0.001, korzystając z interpolacji.

Generowanie wykresu oparłem o przykład ze strony https://www.gnu.org/software/gsl/doc/html/interp.html. Wykorzystywany jest tam gnu plotunit.

W tym zadaniu mamy do dyspozycji 65 węzłów gdyż zaczynając od -1 , -1+1/32 + ... , i przechodząc aż do 1 skaczemy o 1/32. Więc wyliczając mamy 65 mozliwych argumętów funkcji.
Przykład wyliczenia ilości węzłów stosując exela:

	A	B	C	D	E	F	G
1	-1		0		-0,03125		31
2	-0,96875		1		0		32
3	-0,9375		2		0,03125		33
4	-0,90625		3		0,0625		34
5	-0,875		4		0,09375		35
6	-0,84375		5		0,125		36
7	-0,8125		6		0,15625		37
8	-0,78125		7		0,1875		38
9	-0,75		8		0,21875		39
10	-0,71875		9		0,25		40
11	-0,6875		10		0,28125		41
12	-0,65625		11		0,3125		42
13	-0,625		12		0,34375		43
14	-0,59375		13		0,375		44
15	-0,5625		14		0,40625		45
16	-0,53125		15		0,4375		46
17	-0,5		16		0,46875		47
18	-0,46875		17		0,5		48
19	-0,4375		18		0,53125		49
20	-0,40625		19		0,5625		50
21	-0,375		20		0,59375		51
22	-0,34375		21		0,625		52
23	-0,3125		22		0,65625		53
24	-0,28125		23		0,6875		54
25	-0,25		24		0,71875		55
26	-0,21875		25		0,75		56
27	-0,1875		26		0,78125		57
28	-0,15625		27		0,8125		58
29	-0,125		28		0,84375		59
30	-0,09375		29		0,875		60
31	-0,0625		30		0,90625		61
32					0,9375		62
33					0,96875		63
34					1		64
35							

z formułą `=-1+(CX/32)` gdzie cx to numer komurek od 1-65

Kod programu:

```
/*
* @Author: Wojciech Ladyga
* @Date: 2018-12-24
* @Description: Zad 11
*/
#include <iostream>
#include <iomanip>
#include <gsl/gsl_errno.h>
#include <gsl/gsl_linalg.h>
```

```
#include <gsl/gsl_eigen.h>

/*
//przydatne źródła
-wykłady
-https://www.gnu.org/software/gsl/doc/html/interp.html
-http://th-www.if.uj.edu.pl/zfs/gora/metnum17/wyklad07.pdf
-http://www.algorytm.org/procedury-numeryczne/interpolacja-lagrange-a.html
*/

//interpolacja wielomianowa - lagrangea
using namespace std;
const int N = 65; //wezly
double f(double x);
void lagrange(double xi[], double yi[]);
gsl_vector *lu(gsl_vector *x, gsl_vector *y, gsl_vector *a, gsl_permutation
*permutation, gsl_matrix *countMat);

int main()
{
    double xi[N], yi[N];

    printf("#m=0,S=2\n");

    for (int i = 0; i < N; i++)
    {
        xi[i] = -1.0 + (i / 32.0);
        yi[i] = f(xi[i]);
        printf("%g %g\n", xi[i], yi[i]);
    }
    printf("#m=1,S=0\n");

    lagrange(xi, yi);

    return 0;
}

double f(double x)
{
    return 1 / (1 + 5 * pow(x, 2));
}

void lagrange(double xi[], double yi[])
{
    //ogolnie wielomian a(n)x^(n)+a(n-1)x^(n-1)+... = y

    gsl_vector *a = gsl_vector_alloc(N);                    //vector na
wspolczynniki
    gsl_permutation *permutation = gsl_permutation_alloc(N); //wektor permutacji
do dekompozycji LU
    //macierz pozwalająca umieścić odpowiednie wartości typu ax^n
```

```
    gsl_matrix *countMat = gsl_matrix_alloc(N, N);

    //umieszczanie danych w wektorach
    gsl_vector *x = gsl_vector_alloc(N);
    gsl_vector *y = gsl_vector_alloc(N);
    gsl_vector_view tmp_x = gsl_vector_view_array(xi, N);
    gsl_vector_view tmp_y = gsl_vector_view_array(yi, N);
    gsl_vector_memcpy(x, &tmp_x.vector);
    gsl_vector_memcpy(y, &tmp_y.vector);

    double yy;
    for (double xx = -1.0; xx <= 1.0; xx += 0.001)
    {
        yy = 0;
        for (int i = 0; i < N; i++)
        {
            yy += gsl_vector_get(lu(x, y, a, permutation, countMat), i) * pow(xx,
N - 1 - i);
        }
        printf("%g %g\n", xx, yy);
    }

    //postac wielomianu interpolacyjnego
    for (int i = 0; i < N; i++)
    {
        cout << setprecision(4) << fixed << gsl_vector_get(lu(x, y, a,
permutation, countMat), i);
        cout << "x^" << N - 1 - i;
        if (i != N - 1)
        {
            cout << " + ";
        }
    }
    cout << endl;
    //zwalnianie pamięci
    gsl_vector_free(x);
    gsl_vector_free(y);
    gsl_matrix_free(countMat);
    gsl_vector_free(a);
}

//funkcja przelicza wektor zawierający współczynniki
gsl_vector *lu(gsl_vector *x, gsl_vector *y, gsl_vector *a, gsl_permutation
*permutation, gsl_matrix *countMat)
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
        {
            gsl_matrix_set(countMat, i, j, pow(gsl_vector_get(x, i), N - 1 - j));
        }
```

```
    }

    //rozwiązujemy macierz stosując dekompozycje LU
    int tmp;
    gsl_linalg_LU_decomp(countMat, permutation, &tmp);
    gsl_linalg_LU_solve(countMat, permutation, y, a);

    return a;
}
```

Wyniki działania programu to:

```
24919323860.0094x^64 + 344699288.2387x^63 + -284757581933.9059x^62 + -3863515302.0001x^61 + 1542121970084.2913x^60 + 20444212400.1412x^59 + -5269428151082.9395x^58 + -6795313
3267.8725x^57 + 12761726530248.0527x^56 + 159240544399.8997x^55 + -23330674902221.5156x^54 + -279936472421.2621x^53 + 3348813131584l.6016x^52 + 383532086046.1447x^51 + -38761
022214850.4297x^50 + -420018735814.1268x^49 + 36879864012972.6406x^48 + 374166628359.5400x^47 + -29266281539480.3320x^46 + -274512223002.5125x^45 + 19592064274017.5430x^44 +
167320507124.9692x^43 + -11168777181268.1543x^42 + -85237514568.7905x^41 + 5466437111320.9004x^40 + 36430541482.0809x^39 + -2314854382604.4038x^38 + -13089363915.1316x^37 + 8
54864087089.7179x^36 + 3954997792.2920x^35 + -277757875121.9265x^34 + -1003771796.9792x^33 + 80247637199.6442x^32 + 213400481.7674x^31 + -20887878355.4062x^30 + -37836636.100
1x^29 + 4978110355.3011x^28 + 5559556.9048x^27 + -1106996015.2388x^26 + -671142.0198x^25 + 234353108.8388x^24 + 65785.9556x^23 + -48120933.8504x^22 + -5152.1176x^21 + 9723404
.0100x^20 + 315.0276x^19 + -1951078.7267x^18 + -14.5122x^17 + 390546.1411x^16 + 0.4725x^15 + -78122.6435x^14 + -0.0093x^13 + 15624.9471x^12 + 0.0000x^11 + -3124.9991x^10 + 0.
0000x^9 + 625.0000x^8 + -0.0000x^7 + -125.0000x^6 + 0.0000x^5 + 25.0000x^4 + -0.0000x^3 + -5.0000x^2 + 0.0000x^1 + 1.0000x^0
```

oraz wykres