Gliwice, 17.12.2018

PRZEDMIOT: METODY NUMERYCZNE
PROJEKT NR. 2
PARAMETRY DRGAŃ POWIERZCHNI TERENU WYWOŁANYCH WSTRZĄSAMI GÓROTWORU

MARCIN STANIEK
AiIP GIG
ROK II
SEM 3

Dane wejściowe:

Nr stanowiska	X[m]	Y[m]
1	3728	426
2	3728	2494
3	640	2688

s. nr.		Eng[j]	X[m]	Y[m]	Am[m/s2]	Vm[m/s]
1	1	10000000	2071	2059	0.113	0.0033
1	2	40000000	3078	2040	0.217	0.0063
1	3	40000000	3201	2724	0.155	0.0043
1	4	50000000	999	1063	0.127	0.0035
1	5	700000	3337	2743	0.043	0.0012
1	6	8000000	781	2757	0.048	0.0014
1	7	4000000	941	292	0.067	0.0020
1	8	400000	497	461	0.019	0.0005
1	9	7000000	1744	908	0.120	0.0036
1	10	1000000	1898	1226	0.062	0.0019
1	11	60000000	2115	1008	0.240	0.0069
1	12	70000000	543	672	0.110	0.0029
1	13	3000000	3066	288	0.180	0.0062
1	14	7000000	2774	729	0.204	0.0067
1	15	1000000	1694	998	0.058	0.0018
1	16	80000000	1131	240	0.157	0.0042
1	17	400000	1861	2485	0.025	0.0006
1	18	700000	3198	450	0.104	0.0035
1	19	100000	2311	2222	0.006	0.0001
1	20	9000000	3688	168	0.280	0.0098
2	1	10000000	2071	2059	0.158	0.0048
2	2	40000000	3078	2040	0.342	0.0109
2	3	40000000	3201	2724	0.371	0.0121
2	4	50000000	999	1063	0.109	0.0029
2	5	700000	3337	2743	0.107	0.0036
2	6	8000000	781	2757	0.075	0.0022
2	7	4000000	941	292	0.044	0.0013
2	8	400000	497	461	0.014	0.0004
2	9	7000000	1744	908	0.091	0.0027
2	10	1000000	1898	1226	0.055	0.0016
2	11	60000000	2115	1008	0.185	0.0052
2	12	70000000	543	672	0.084	0.0022
2	13	3000000	3066	288	0.080	0.0024
2	14	7000000	2774	729	0.122	0.0037
2	15	1000000	1694	998	0.047	0.0014
2	16	80000000	1131	240	0.098	0.0026
2	17	400000	1861	2485	0.041	0.0011
2	18	700000	3198	450	0.049	0.0014
2	19	100000	2311	2222	0.009	0.0001
2	20	9000000	3688	168	0.110	0.0032
3	1	10000000	2071	2059	0.171	0.0053
3	2	40000000	3078	2040	0.142	0.0039
3	3	40000000	3201	2724	0.139	0.0038
3	4	50000000	999	1063	0.237	0.0069
3	5	700000	3337	2743	0.036	0.0010
3	6	8000000	781	2757	0.275	0.0098
3	7	4000000	941	292	0.083	0.0025
3	8	400000	497	461	0.034	0.0009
3	9	7000000	1744	908	0.117	0.0035
3	10	1000000	1898	1226	0.065	0.0020
3	11	60000000	2115	1008	0.181	0.0050
3	12	70000000	543	672	0.210	0.0059
3	13	3000000	3066	288	0.043	0.0013
3	14	7000000	2774	729	0.075	0.0022
3	15	1000000	1694	998	0.062	0.0019
3	16	80000000	1131	240	0.166	0.0045
3	17	400000	1861	2485	0.056	0.0016
3	18	700000	3198	450	0.024	0.0007
3	19	100000	2311	2222	0.008	0.0001
3	20	9000000	3688	168	0.044	0.0013

```
//
//  main.cpp
//  mn_projekt_2_parametry_drgań
//
//  Created by Marcin Staniek on 19/11/2018.
//  Copyright © 2018 Marcin Staniek. All rights reserved.
//

#include <iostream>
#include <iomanip>
#include <fstream>
#include <sstream>
#include <math.h>
#include <cmath>

using namespace std;

const double eps = 1e-12;//stała przybliżenia zera

bool gauss(int n, double ** GAUSS, double * B)
{
    int i,j,k;
    double m,s;

    for(i = 0; i < n - 1; i++)//eliminacja współczynników
    {
        for(j = i + 1; j < n; j++)
        {
            if(fabs(GAUSS[i][i]) < eps) return false;
            m = -GAUSS[j][i] / GAUSS[i][i];
            for(k = i + 1; k <= n; k++)
                GAUSS[j][k] += m * GAUSS[i][k];
        }
    }
    for(i = n - 1; i >= 0; i--)//wyliczanie niewiadomych
    {
        s = GAUSS[i][n];
        for(j = n - 1; j >= i + 1; j--)
            s -= GAUSS[i][j] * B[j];
        if(fabs(GAUSS[i][i]) < eps) return false;
        B[i] = s / GAUSS[i][i];
    }
    return true;
}
```

```
int main()
{
    int n=4;//ilosc niewiadomych do gaussa
    int i, j, l;//zmienne w petlach
    int w=0, k=0;//ilosci wierszy i kolumn
    //double energia_sejsmiczna_wstrzasu=200000;
    double macierz_wspolrzednych_x[3], macierz_wspolrzednych_y[3];
    macierz_wspolrzednych_x[0]=3728;//wspolrzedne x stanowisk 1  2 3
    macierz_wspolrzednych_x[1]=3728;
    macierz_wspolrzednych_x[2]=640;

    macierz_wspolrzednych_y[0]=426;//wpolrzedne y stanowisk 1 2 3
    macierz_wspolrzednych_y[1]=2494;
    macierz_wspolrzednych_y[2]=2688;

    //zczytanie danych z pliku do macierzy 60x7----------------------------
    cout << "wczytane dane:" << endl;
    w=60;//
    k=7;
    string macierz_danych[w][k];//stringi z pliku przed konwersja na double
    double macierz_danych_double[w][k];//double po kowersji
    ifstream dane; //otwarcie pliku z danymi

dane.open("/Users/stanik/Desktop/studies/sem3/metody_numeryczne/mn_projekt_
2_parametry_drgań/mn_projekt_2_parametry_drgań/dane");//ścieżka
    if(dane.is_open())//odczyt danych z pliku do macierzy
    {
        for(i = 0; i < w; i++)
        {
            for(j = 0; j < k; j++)
            {
                dane >> macierz_danych[i][j];
                cout << macierz_danych[i][j] << " ";
            }
            cout << endl;
        }
    }
    cout << endl;
    for(i = 0; i < w; i++)//konwersja na stringow na liczby
    {
        for(j = 0; j < k; j++)
        {
            stringstream s;
            s << macierz_danych[i][j];
            s >> macierz_danych_double[i][j];
            cout << macierz_danych_double[i][j] << " "; //wyswietlenie czy
te same liczby
        }
        cout << endl;
    }
    cout << endl;
    double suma;//sprawdzenie czy da sie sumowac inty po konwersji czyli
czy konwersja działa ;)
    suma=macierz_danych_double[0][0]+macierz_danych_double[1][1];
    cout <<"sprawdzenie czy dodaje ('jesli jest 3 to oki') "<< "'"<< suma
<< "'" << endl;
    cout << endl;


    //A 60x4---------------------------------------------------------------
```

```
    cout << "A 60x4: " << endl;
    w=60;
    k=4;
    double macierza[w][k];
    for (i=0; i<w; i++)//tworzenie macierzy A
    {
        for (j=0; j<k; j++)
        {
            macierza[i][j]=0;
            if(j==0)
            {
                macierza[i][j]=log10(macierz_danych_double[i][2]);
            }
            if(j==1)//kolumna druga zmienia sie co 20 bo 3 wstrzasy
!!dziwnie zapisane R=log10 powinno byc log10(R)
            {
                if((i >= 0) && (i <= 19))
                {
                macierza[i][j]=log10(sqrt(pow(macierz_danych_double[i][3]-
macierz_wspolrzednych_x[0],2) + pow(macierz_danych_double[i][4]-
macierz_wspolrzednych_y[0],2) + pow(500,2)));
                }
                if((i >= 20) && (i <= 39))
                {
                macierza[i][j]=log10(sqrt(pow(macierz_danych_double[i][3]-
macierz_wspolrzednych_x[1],2) + pow(macierz_danych_double[i][4]-
macierz_wspolrzednych_y[1],2) + pow(500,2)));
                }
                if((i >= 40) && (i <= 59))
                {
                macierza[i][j]=log10(sqrt(pow(macierz_danych_double[i][3]-
macierz_wspolrzednych_x[2],2) + pow(macierz_danych_double[i][4]-
macierz_wspolrzednych_y[2],2) + pow(500,2)));
                }
            }
            if(j==2)//kolumna trzecia zmienia sie co 20 bo 3 wstrzasy
            {
                if((i >= 0) && (i <= 19))
                {
                    macierza[i][j]=sqrt(pow(macierz_danych_double[i][3]-
macierz_wspolrzednych_x[0],2) + pow(macierz_danych_double[i][4]-
macierz_wspolrzednych_y[0],2) + pow(500,2));
                }
                if((i >= 20) && (i <= 39))
                {
                    macierza[i][j]=sqrt(pow(macierz_danych_double[i][3]-
macierz_wspolrzednych_x[1],2) + pow(macierz_danych_double[i][4]-
macierz_wspolrzednych_y[1],2) + pow(500,2));
                }
                if((i >= 40) && (i <= 59))
                {
                    macierza[i][j]=sqrt(pow(macierz_danych_double[i][3]-
macierz_wspolrzednych_x[2],2) + pow(macierz_danych_double[i][4]-
macierz_wspolrzednych_y[2],2) + pow(500,2));
                }
            }
```

```
        if(j==3)
        {
            macierza[i][j]=1;
        }
    }
}
for (i=0; i<w; i++)//wyswietlenie macierzy A
{
    for (j=0; j<k; j++)
    {
        cout << macierza[i][j] << " ";
    }
    cout << endl;
}
cout << endl;

    //AT 4x60-------------------------------------------------------------
cout << "AT 4x60: " << endl;
w=4;
k=60;
double macierzat[w][k];
for(i = 0; i < k; i++) //transponowanie macierzy
{
    for(j = 0;j < w; j++)
    {
        macierzat[j][i]=macierza[i][j];
    }
}
for(i = 0; i < w; i++) //wyswietlanie macierzy transponowanej
{
    for(j = 0;j < k; j++)
    {
        cout << macierzat[i][j] << " ";
    }
    cout << endl;
}
cout << endl;

    //Y 60x1--------------------------------------------------------------
cout << "Y 60x1: " << endl;
w=60;
k=1;
double macierzY[w][k];
for (i=0; i<w; i++)//tworzenie macierzy B
{
    for (j=0; j<k; j++)
    {
        macierzY[i][j]=0;
        macierzY[i][j]=log10(macierz_danych_double[i][5]);
        cout << macierzY[i][j] << " ";
    }
    cout << endl;
}
cout << endl;
```

```
    //AT*A 4x60*60x4=4x4-------------------------------------------------------
    cout << "AT*A 4x60*60x4=4x4:" << endl;
    w=4;
    k=4;
    double macierzata[w][k];
    for(i=0; i<w; i++)//mnożenie
    {
        for(j=0; j<k; j++)
        {
            macierzata[i][j]=0;
            for(l=0; l<60; l++)
            {
                macierzata[i][j] += macierzat[i][l] * macierza[l][j];
            }
        }
    }
    for(i=0; i<w; i++)//wyswietlenie AT*A
    {
        for(j=0; j<k; j++)
        {
            cout << macierzata[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl;

    //AT*Y 4X60*60x1=4x1-------------------------------------------------------
-----------------------------------------------------
    cout << "AT*Y 4x60*60x1=4x1" << endl;
    w=4;
    k=1;
    double macierzatY[w];
    for(i=0; i<w; i++)//mnożenie
    {
        for(j=0; j<k; j++)
        {
            macierzatY[i]=0;
            for(l=0; l<60; l++)
            {
                macierzatY[i]+= macierzat[i][l] * macierzY[l][j];
            }
        }
    }
    for(i=0; i<w; i++)//wyswietlenie AT*Y
    {
        for(j=0; j<k ;j++)
        {
            cout << macierzatY[i]<< " ";
        }
        cout << endl;
    }
    cout << endl;
```

```
//rozwiazanie gaussem-----------------------------------------------------
double **GAUSS, *B;
GAUSS = new double * [n];//tworzymy macierze GAUSS i B
B  = new double [n];
for(i = 0; i < n; i++)//tworzymy macierz GAUSS dynamiczną
{
    GAUSS[i] = new double[n + 1];
}
cout << "GAUSS 4x5:" << endl;
for(i = 0; i < 4; i++)//odczytujemy dane dla macierzy GAUSS
{
    for(j = 0; j < 5; j++)
    {
        if(j<4)
        {
            GAUSS[i][j]=macierzata[i][j];
        }
        else
        {
            GAUSS[i][j]=macierzatY[i];
        }

    }
}
cout << "Wstawienie do Gauss'a 4x5" << endl;
for(i = 0; i < 4; i++)//wyswietlenie GAUSS
{
    for(j = 0; j < 5; j++)
    {
        cout << GAUSS[i][j] << " ";
    }
    cout << endl;
}
cout << endl;
cout << "Rozwiazanie GAUSSA: " << endl;
if(gauss(n,GAUSS,B))//wyliczenie gaussem
{
    for(i = 0; i < n; i++)
    {
        if(i==0)
        {
            cout << "b1= " << B[i] << endl;
        }
        if(i==1)
        {
            cout << "b2= " << B[i] << endl;
        }
        if(i==2)
        {
            cout << "b3= " << B[i] << endl;
        }
        if(i==3)
        {
            cout << "b4= " << B[i] << endl;
        }
    }
}
else
{cout << "DZIELNIK ZERO\n";}
cout << endl;
```

```
    double odleglosci_epicentralne[4];
    odleglosci_epicentralne[0]=0;//strefa epicentralna
    odleglosci_epicentralne[1]=500;//odleglosci epicentralne
    odleglosci_epicentralne[2]=1000;
    odleglosci_epicentralne[3]=1500;

    double Repicentralne,R500,R1000,R1500;
    Repicentralne=sqrt(pow(odleglosci_epicentralne[0],2)+pow(500, 2));
    R500=sqrt(pow(odleglosci_epicentralne[1],2)+pow(500, 2));
    R1000=sqrt(pow(odleglosci_epicentralne[2],2)+pow(500, 2));
    R1500=sqrt(pow(odleglosci_epicentralne[3],2)+pow(500, 2));

    double Aepicentralne,Aepicentralne500, Aepicentralne1000,
Aepicentralne1500;

Aepicentralne=pow(10,(B[0]*log10(200000))+(B[1]*log10(Repicentralne))+(B[2]
*Repicentralne)+B[3]);

Aepicentralne500=pow(10,(B[0]*log10(200000))+(B[1]*log10(R500))+(B[2]*R500)
+B[3]);

Aepicentralne1000=pow(10,(B[0]*log10(200000))+(B[1]*log10(R1000))+(B[2]*R10
00)+B[3]);

Aepicentralne1500=pow(10,(B[0]*log10(200000))+(B[1]*log10(R1500))+(B[2]*R15
00)+B[3]);


    //przedziały ufnosci-----------------------------------------------
    double odchylenie_standardowe[60], odchylenie_standardoweSR=0;//E
    double sigma=0;//o-
    double lambda=1.6707;//h

    //Yprog maciedz 60x1 przyspieszen prognozowanych
    cout << "Yprog 60x1:" << endl;
    w=60;
    k=1;
    double macierzYprog[w][k];//logAuf
    for (i=0; i<60; i++)
    {
        for (j=0; j<1; j++)
        {

macierzYprog[i][j]=(B[0]*macierza[i][0])+(B[1]*macierza[i][1])+(B[2]*macier
za[i][2])+B[3];
            cout << macierzYprog[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl;

    //odchylenie standardowe
    cout << "Odchylenie standardowe: " << endl;
    for(i=0;i<60;i++)
    {
        odchylenie_standardowe[i]=macierzY[i][0]+macierzYprog[i][0];
        cout << odchylenie_standardowe[i] << endl;
    }
```

```
    cout << endl;

    for(i=0;i<60;i++)//odchylenie standardowe średnie
    {
        odchylenie_standardoweSR+=odchylenie_standardowe[i];
    }
    odchylenie_standardoweSR=odchylenie_standardoweSR/60;
    cout << "Odchylenie standardowe średnie: " << endl <<
odchylenie_standardoweSR << endl;
    cout << endl;

    //sigma
    cout << "Sigma: " << endl;
    for(i=0;i<60;i++)
    {
        sigma+=pow(odchylenie_standardowe[i]-odchylenie_standardoweSR,2);
    }
    sigma=sqrt(sigma)/59;
    cout << sigma << endl;
    cout << endl;

    double AepicentralneUF,Aepicentralne500UF, Aepicentralne1000UF,
Aepicentralne1500UF;
    AepicentralneUF=pow(10,log10(Aepicentralne)+(sigma*lambda));
    Aepicentralne500UF=pow(10,log10(Aepicentralne500)+(sigma*lambda));
    Aepicentralne1000UF=pow(10,log10(Aepicentralne1000)+(sigma*lambda));
    Aepicentralne1500UF=pow(10,log10(Aepicentralne1500)+(sigma*lambda));

    cout << "A: " << Aepicentralne << endl;
    cout << "A500: " << Aepicentralne500 <<  endl;
    cout << "A1000: " << Aepicentralne1000 << endl;
    cout << "A1500: " << Aepicentralne1500 << endl;
    cout << endl;
    cout << "Auf: " << AepicentralneUF<< endl;
    cout << "Auf500: " << Aepicentralne500UF <<  endl;
    cout << "Auf1000: " << Aepicentralne1000UF << endl;
    cout << "Auf1500: " << Aepicentralne1500UF << endl;

}
```

Wyniki działania programu:

```
Odchylenie standardowe średnie:
-2.18524

Sigma:
0.0986468

A: 0.0680803
A500: 0.0573165
A1000: 0.0432811
A1500: 0.0329831

Auf: 0.0995017
Auf500: 0.0837701
Auf1000: 0.0632568
Auf1500: 0.0482059
```