# *Національний технічний університет України*

# *«Київський політехнічний інститут»*

#### ***Факультет інформатики та обчислювальної техніки***

## Лабораторна робота №3

*з курсу "* **АЛГОРИТМИ ТА МЕТОДИ ОБЧИСЛЕНЬ***"*

***Виконав:***

*Бедь А.М.*

***Група*** *ІО-12,*

***Номер залікової книжки:*** *1202*

***Київ - 2013р.***

**Тема:** «Інтерполяція функцій».

**Мета:** Ознайомлення з інтерполяційними формулами Лагранжа, Ньютона.

**Завдання:** Закріплення, поглиблення і розширення знань студентів при

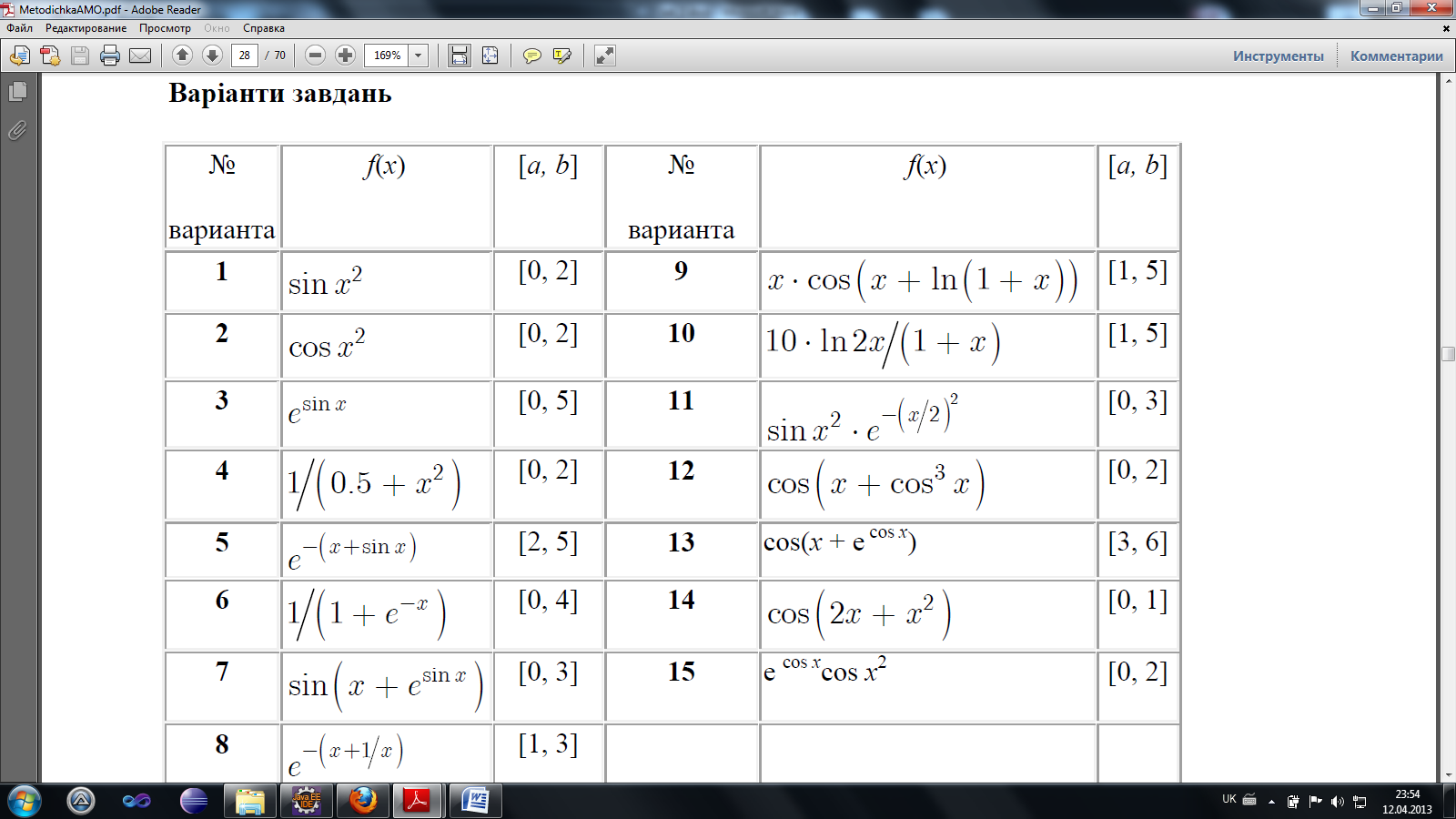
вирішенні практичних обчислювальних завдань. Оволодіння обчислювальними

методами і практичними методами оцінки похибки обчислень. Придбання умінь і навичок при програмуванні та налагодженні обчислювальних завдань на

комп'ютері.

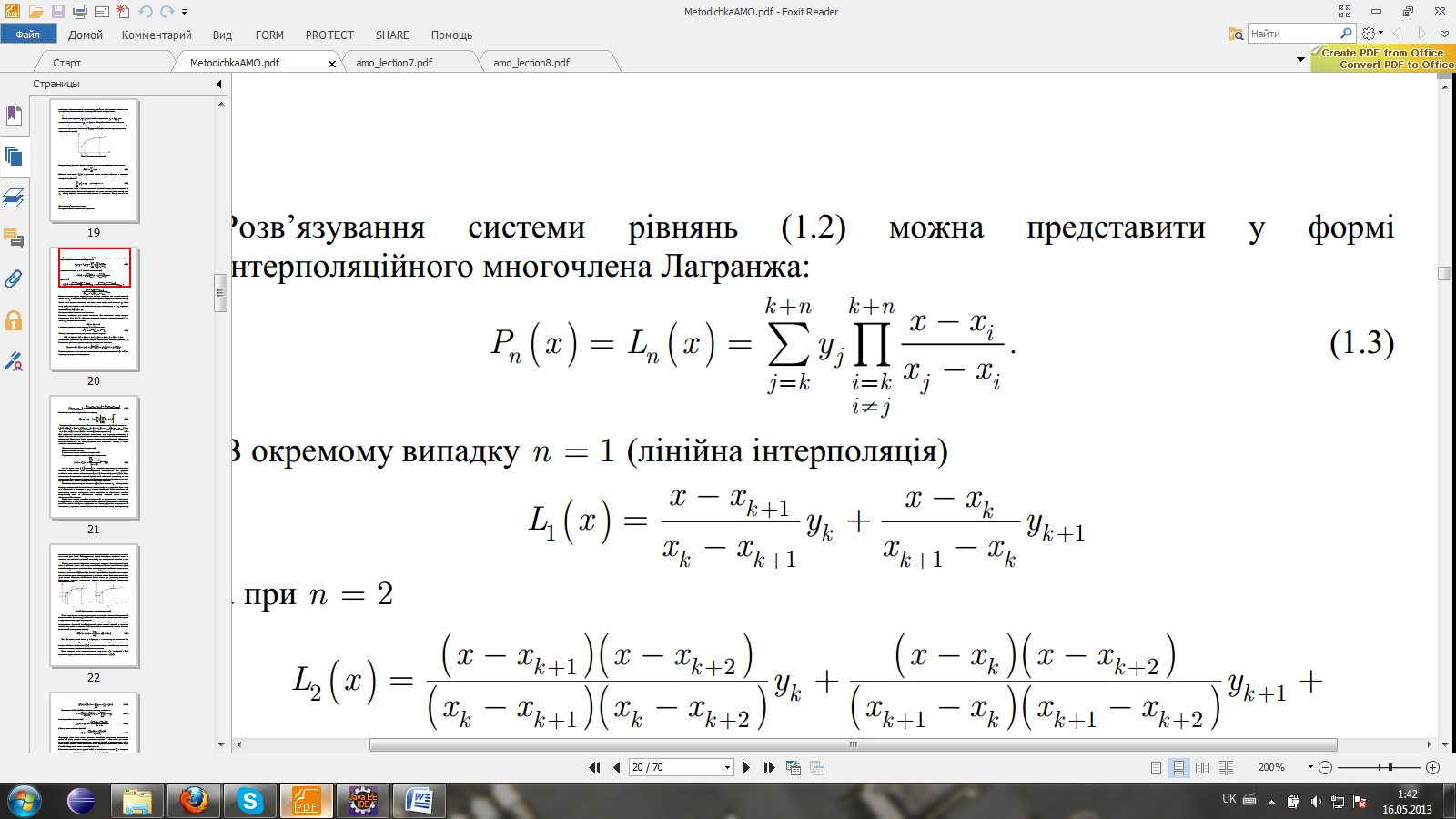
Варіант:

Метод Лагранджа.



Короткі теоретичні відомості.

Інтерполяційний многочлен Лагранжа



Лістинг коду.

**package** com.fiot.amo.lab3;

**public** **abstract** **class** Function {

**public** **abstract** **double** value(**double** x);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**package** com.fiot.amo.lab3;

**public** **class** Interpolation {

**double**[] x;

**double**[] y;

Function function;

**double** a;

**double** b;

**int** m;

**int** n;

**private** **double** h;

**public** Interpolation(**int** m, **int** n, **double** a, **double** b) {

**this**.a = a;

**this**.b = b;

**this**.m = m;

**this**.n = n;

h = (b - a) / (m - 1);

x = **new** **double**[m];

y = **new** **double**[m];

function = **new** Function() {

@Override

**public** **double** value(**double** x) {

**return** Math.*cos*(x \* x);

}

};

x[0] = a;

y[0] = function.value(x[0]);

**for** (**int** i = 1; i < x.length; i++) {

x[i] = x[i - 1] + h;

y[i] = function.value(x[i]);

}

}

**public** **double** interpolate(**double** value) {

**double** result = 0;

**double** mul;

**for** (**int** k = 0; k < m - 1; k++) {

**if** (value >= x[k] && value < x[k + 1]) {

**if** (k <= m / 2) {

**for** (**int** j = k; j <= k + n; j++) {

mul = y[j];

**for** (**int** i = k; i <= k + n; i++) {

**if** (i != j) {

mul = mul \* (value - x[i]) / (x[j] - x[i]);

}

}

result = result + mul;

}

} **else** {

**for** (**int** j = k + 1; j >= k - n + 1; j--) {

mul = y[j];

**for** (**int** i = k + 1; i >= k - n + 1; i--) {

**if** (i != j) {

mul = mul \* (value - x[i]) / (x[j] - x[i]);

}

}

result = result + mul;

}

}

}

}

**return** result;

}

**public** **double**[] getError() {

**double**[] result = **new** **double**[m];

**for** (**int** i = 0; i < result.length; i++) {

result[i] = Math.*abs*(interpolate(h / 2 + i \* h)

- function.value(h / 2 + i \* h));

}

**return** result;

}

**public** **double** getMeanError() {

**double**[] error = getError();

**double** result = 0;

**for** (**int** i = 0; i < error.length; i++) {

result = result + error[i];

}

result = result / error.length;

**return** result;

}

@Override

**public** String toString() {

StringBuilder builder = **new** StringBuilder();

**for** (**int** i = 0; i < x.length; i++) {

builder.append("(" + x[i] + ", " + y[i] + ") \n");

}

**return** builder.toString();

}

**public** Function getFunction() {

**return** function;

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
**package** com.fiot.amo.lab3;

**import** java.applet.Applet;

**import** javax.swing.JFrame;

**import** org.jfree.chart.ChartFactory;

**import** org.jfree.chart.ChartPanel;

**import** org.jfree.chart.JFreeChart;

**import** org.jfree.chart.plot.PlotOrientation;

**import** org.jfree.data.xy.XYSeries;

**import** org.jfree.data.xy.XYSeriesCollection;

**public** **class** Test **extends** Applet {

**enum** Task {

*Interpolate*, *Error*

}

**public** **static** **void** main(String[] args) {

**int** m = 11;

**int** n = 1;

**int** a = 0;

**int** b = 2;

Task task = Task.*Interpolate*;

Interpolation interpolation = **new** Interpolation(m, n, a, b);

JFrame frame = **new** JFrame();

frame.setTitle("Графік");

frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);

XYSeries series = **new** XYSeries("Функція");

XYSeries series2 = **new** XYSeries("Інтерполяція");

XYSeries series3 = **new** XYSeries("Похибка");

**double** tempI;

**double** tempO;

**for** (**int** i = 0; i < 50 \* b; i++) {

// series.add(interpolation.x[i], interpolation.y[i]);

tempI = interpolation.interpolate(i / 50.0f);

tempO = interpolation.function.value(i / 50.0f);

series.add(i / 50.0f, tempO);

series2.add(i / 50.0f, tempI);

series3.add(i / 50.0f, Math.*abs*(tempI - tempO));

}

XYSeriesCollection data = **new** XYSeriesCollection();

**if** (task == Task.*Interpolate*) {

data.addSeries(series);

data.addSeries(series2);

} **else** {

data.addSeries(series3);

}

**final** JFreeChart chart = ChartFactory.*createXYLineChart*(

"Декартова система", "X", "Y", data, PlotOrientation.*VERTICAL*,

**true**, **true**, **false**);

**final** ChartPanel chartPanel = **new** ChartPanel(chart);

chartPanel.setPreferredSize(**new** java.awt.Dimension(2000, 700));

frame.setContentPane(chartPanel);

frame.pack();

frame.setVisible(**true**);

System.*out*.println("\nПохибка на кожному проміжку: ");

**double**[] error = interpolation.getError();

**for** (**int** i = 0; i < error.length; i++) {

System.*out*.format("%.8f%n", error[i]);

}

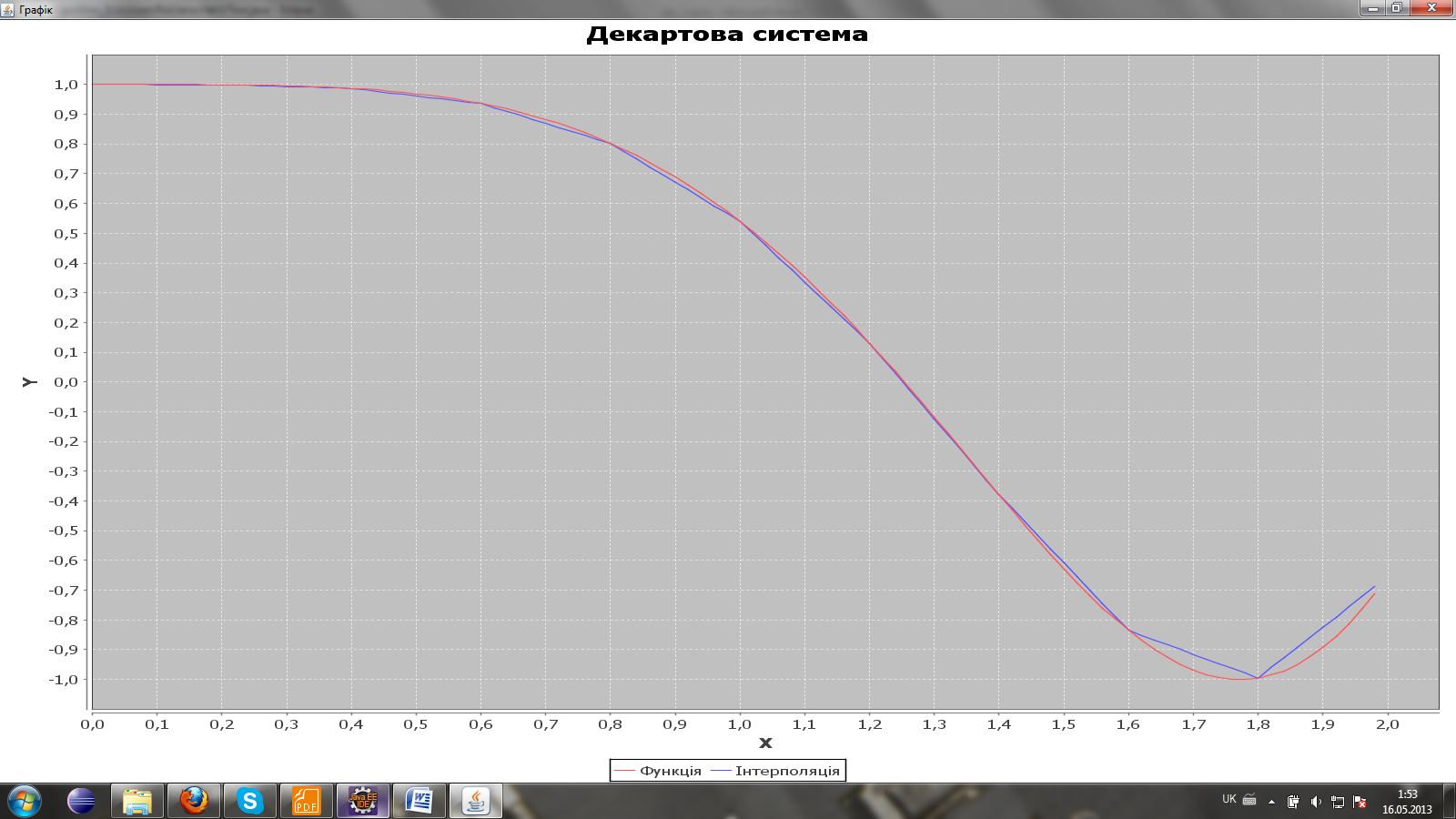
System.*out*.println("\nСередня похибка: ");

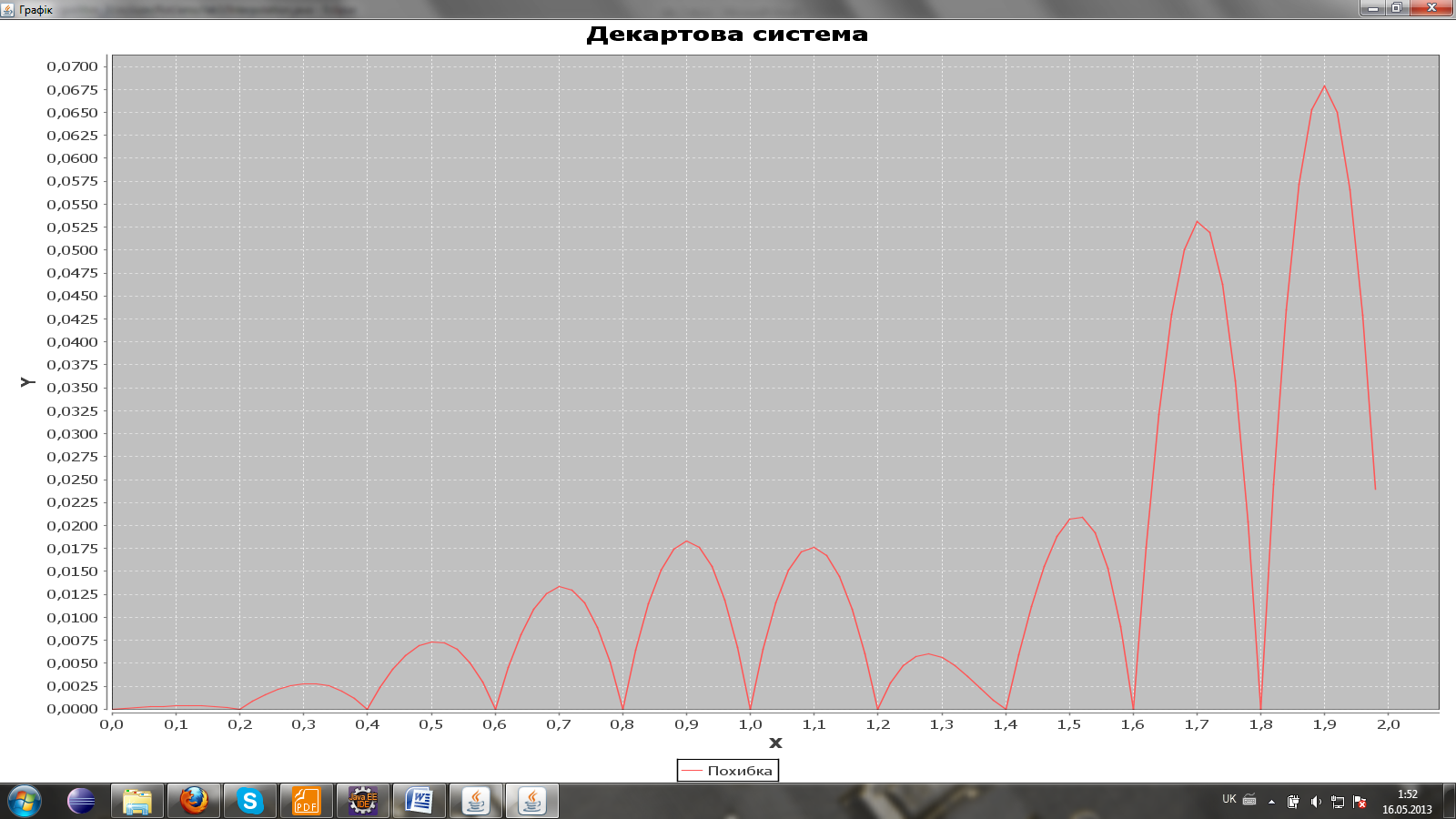
System.*out*.format("%.8f%n", interpolation.getMeanError());

}

}

Приклад роботи програми





Похибка на кожному проміжку:

0,00021059

0,00034139

0,00028903

0,00023218

0,00136281

0,00251167

0,00006931

0,00082372

0,00213260

0,00255443

Середня похибка:

0,00105277