Silesian University of Technology
Faculty of Automatic Control,
Electronic and Computer Science
Major: Makro sem.4

Gliwice
Academic year 2018/2019

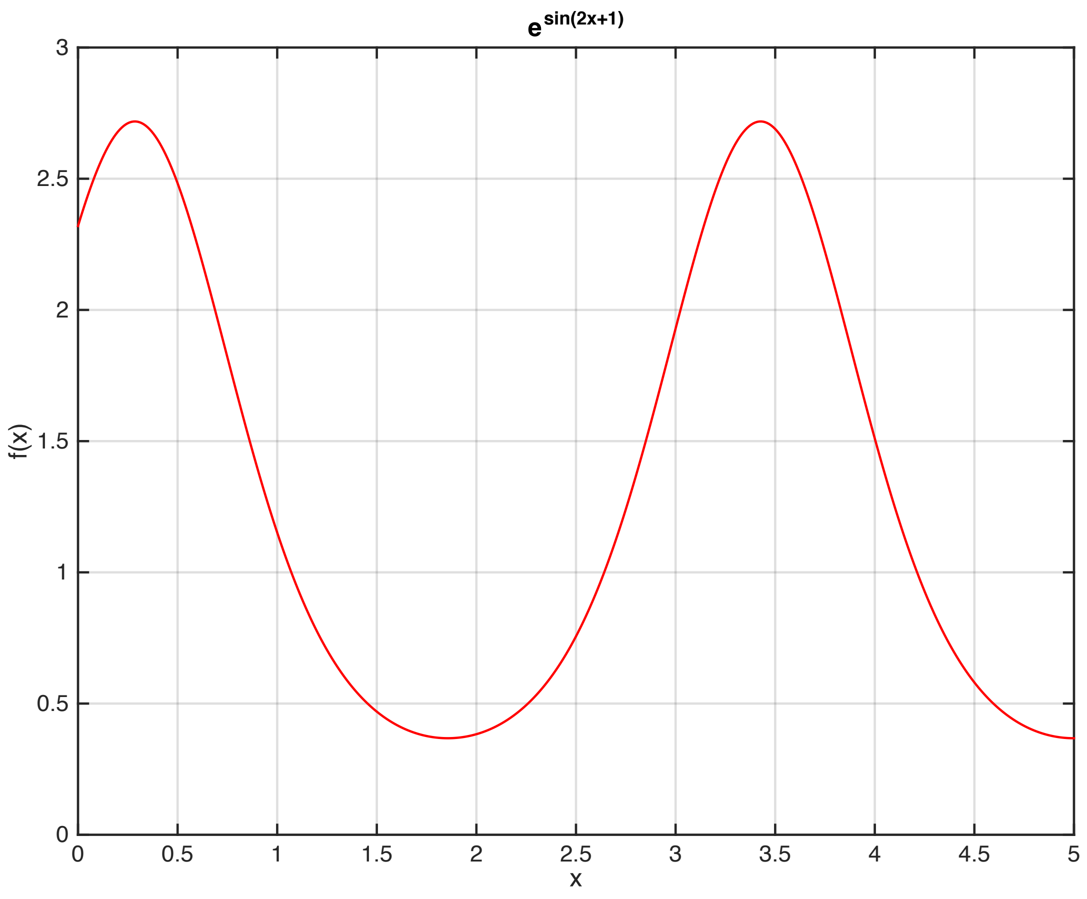
# Numerical Methods
# Laboratory Exercise

## Ex.5 Numerical integration

Group 2, Section 4
Mateusz Warmuz
Szymon Skorupa
Exercise preformed on: 11.04.2019

Analyze the arbitrary function for numerical integration methods considered in closed interval [a,b]

$$f(x) = \int_1^4 e^{sin(2x+1)} dx \approx 3.78983$$



Source code:

```
#include <iostream>
#include <cmath>
using namespace std;

double f (double x){
    return exp(sin(2*x+1));
    //return pow(x,3)+2*pow(x,2)-3*x+1;
    //return pow(x,2)-x+1;
}
double f2(double x){
    return -4*exp(sin(2*x+1))*(sin(2*x+1)-(cos(2*x+1)*cos(2*x+1)));
    //return 6*x+4;
   // return 2;
}
double f4(double x){
    return 16*exp(sin(2*x+1))*((sin(2*x+1)*(3*sin(2*x+1)+1))+
(cos(2*x+1)*cos(2*x+1)*cos(2*x+1)*cos(2*x+1))-2*(3*sin(2*x+1)+2)*cos(2*x+1)*cos(2*x+1));
    //return 0;
}

double composite_rectangle(double a, double b, int m){
    double h;
```

```cpp
    double *x;
    double value=0;
    x = new double [m+1];
    h=(b-a)/m;
    for(int i=0;i<=m;i++){
        x[i]=a+i*h;

        value += f(x[i])*h;

    }
    delete []x;
    cout<<"Composite rectangle formula: "<<value<<endl;
    cout<<endl;
    return value;
}
//Composite trapezoid formula
double composite_trapezoid(double a, double b, int m){
    double h;
    double integral;
    double value;
    double error;
    double *x;
    double *y;                  //the largest element
    y = new double [m];
    x = new double [m+1];
    h=(b-a)/m;
    for(int i=0;i<=m;i++){
        x[i]=a+i*h;

        y[i]=f2(x[i]);          //finding the largest element necesary for error

        if(y[0]<y[i]){
            y[0]=y[i];
        }
    }
    value=0;
    for(int i=0;i<=m-1;i++){
        value += (f(x[i])+f(x[i+1]))/2;
    }
    integral=h*value;
    error=fabs((m*pow(h,3))/12)*y[0];

    delete []y;
    delete []x;

    cout<<"Composite Trapezoid formula: "<<integral<<endl;
    cout<<"Error of Composite trapezoid formula: "<<error<<endl;
    cout<<endl;
    return integral;
}

double composite_simpson(double a, double b, int m){
    double h;
    double *x;
    double *y;
    double amount=(2*m)+1;
    double value=0;
    double error;

    x = new double [amount];
    y = new double [amount];

    h=((b-a)/(2*m));

    for(int i=0;i<amount;i++){

        x[i]=a+i*h;



        y[i]=f4(x[i]);              //finding the largest element necesary for error
        if(y[0]<y[i]){
            y[0]=y[i];
        }

        if (i == 0 || i == 2*m)
            value += f(x[i]);
        else if (i%2 == 1)
            value += 4*f(x[i]);
        else if (i%2 == 0)
            value += 2*f(x[i]);
    }

    value *=h/3;
    error=fabs((m*pow(h,5)/90)*y[0]);

    delete []y;
    delete []x;

    cout<<"Composite Simpson formula: "<<value<<endl;
    cout<<"Error of Composite trapezoid formula: "<<error<<endl;
    cout<<endl;
    return h;
}


double rectangle(double a, double b)
{
    cout << "Rectangle formula: " << f(a)*(b-a) << endl;
    cout<<endl;
    return f(a)*(b-a);
}

double trapezoid(double a, double b)
{
    double h;
    double integral;
    double error;
    double point=(b-a)/2;

    h=(b-a);
    integral=(h/2)*(f(a)+f(b));

    error=(pow(h,3)*f2(point))/-12;
    if(error>0){
        integral -= error;
        cout<<"Error: "<<error<<" So we substract to "<<(h/2)*(f(a)+f(b))<<endl;
    }
    else{
        integral += error;
        cout<<"Error: "<<error<<" So we add to "<<(h/2)*(f(a)+f(b))<<endl;
    }

    cout<<"Trapezoid formula: "<<integral<<endl;
    cout<<endl;
    return integral;
}
double simpson(double a, double b)
{
    double x0 = a;
    double x1 = (a+b)/2;
    double x2 = b;
    double h = (x2-x0)/2;
    double integral =1;
    double error;

    integral = (h/3)*(f(x0)+(4*f(x1))+f(x2));
    error=(pow(h,5)*f4(x1))/-90;

    cout<<"Simpson formula: "<<integral<<endl;
    cout<<"Error of simpson formula: "<<error<<endl;
    cout<<endl;
    return integral;
}
```

```
double boole (double a, double b){
    double x[5];
    double h=(b-a)/4;
    double integral;
    for(int i=0;i<5;i++){
        x[i]=a+i*h;

    }
    integral = ((2*h)/45)*(4*f(x[0])+32*f(x[1])+12*f(x[2])+32*f(x[3])+7*f(x[4]));
    cout<<"Boole'a formula: "<<integral<<endl;
    cout<<endl;
    return integral;
}
double chebyshev(double a, double b, int n)
{
    double *t;
    t = new double[n];
    double *x;
    x=new double [n];
    double value=0;
    double integral=0;
    if(n==2)
    {
        t[0]=-0.57735;
        t[1]=0.577350;
    }
    if(n==3){
        t[0]=-0.707107;
        t[1]=0;
        t[2]=0.707107;
    }

    if(n==4)
    {
        t[0]=-0.794654;
        t[1]=-0.187592;
        t[2]=0.187592;
        t[3]=0.794654;
    }
    if(n==5){
        t[0]=-0.832498;
        t[1]=-0.374541;
        t[2]=0;
        t[3]=0.374541;
        t[4]=0.832498;
    }
    if(n==6){
        t[0]=-0.866247;
        t[1]=-0.422519;
        t[2]=-0.266635;
        t[3]=0.266635;
        t[4]=0.422519;
        t[5]=0.866247;
    }
    if(n==7){
        t[0]=-0.883862;
        t[1]=-0.529657;
        t[2]=-0.323912;
        t[3]=0;
        t[4]=0.323912;
        t[5]=0.529657;
        t[6]=0.883862;
    }

    for(int i=0;i<n;i++){
        x[i]=((a+b)/2)+(((a-b)/2)*(t[i]));
        value += f(x[i]);
    }

    integral = value *((b-a)/n);
    delete []x;
    delete []t;

    cout<<"Chebyshev formula: "<<integral<<endl;
    cout<<endl;
    return integral;

}

int main() {
    double a=1, b=4;
    int n=3;            //Amount of chebyshev nodes
    int m=20;            //small sub-intervals

    rectangle(a,b);
    trapezoid(a,b);
    simpson(a,b);
    chebyshev(a,b,n);

    //composite methods
    boole(a,b);
    composite_trapezoid(a, b, m);
    composite_simpson(a,b,m);
    composite_rectangle(a,b,m);

    cout<< chebyshev(a,(b-a)/2,n)+chebyshev((b-a)/2,b,n)<<endl; //Chebyshev for m=2
}
```

Results of testing:

| | | | Newton-Cotes | | | Chebyshev | |
|---|---|---|---|---|---|---|---|
| n | m | Method | $\int_a^b$ f(x) dx | $\Delta_{absolute}$ | $\Delta_{equetion}$ | $\int_a^b$ f(x) dx | $\Delta_{absolute}$ |
| 0 | 1 | Rectangle | 3,45469 | 0,33514 | - | - | - |
| 1 | 1 | Trapezoid | -1,00727 | 4,7971 | -4,99964 | - | - |
| 2 | 1 | Simpson | 2,84324 | 0,94659 | 1,36531 | 4,65538 | 0,86555 |
| 4 | 1 | Boole's | 3,79945 | 0,00962 | | 3,59545 | 0,19438 |
| 0 | 4 | Rectangle | 4,76168 | 0,97185 | - | - | - |
| 1 | 4 | Trapezoid | 3,62937 | 0,16046 | 0,543443 | - | - |
| 2 | 2 | Simpson | 3,97461 | 0,18478 | 0,524928 | 3,79181 | 0,00198 |
| 0 | 20 | Rectangle | 3,98855 | 0,19872 | - | - | - |

Conclusions:

Regarding the part for m = 1 we found out that the best result was given by Newton-Coles with n = 4. Rectangle method got surprisingly close to the analytical result mostly due to our function and the range in which the integral was calculated (a and b being in almost the same point of the cyclic function but different period).

Concerning the second part being the composite methods we figured that the best result was obtained using the chebyshev method with m = 2 (our integration range was divided into new two equal ranges and the chebyshev method applied separately to both then added up).