Homework 1

Zangarini Riccardo

November 8, 2021

1 Code

The function is not continuous on the whole x-axis. The function is:

$$f(x) = \frac{\sin x}{x} \tag{1}$$

and it presents a point of discontinuity in x = 0. However it can be shown that:

$$\lim_{x \to 0^+} f(x) = \lim_{x \to 0^-} f(x) = 1 \tag{2}$$

and so the discontinuity is negligible. In order to extend the continuity in the point x = 0, the function must be splitted in two parts:

$$f(x) = \begin{cases} 1, & \text{if } x = 0 \\ \frac{\sin(x)}{x}, & \text{if } x \neq 0 \end{cases} \tag{3}$$

With this definition of $f(x)$, the quadrature methods used in the code will not have any problem for the evaluation in x = 0. The code is the following:

```
#include <iostream>
#include <math.h>
#include <ctime>
#include <iomanip>
#include <stdlib.h>
using namespace std;

double trapezoidal(double (*F)(double), double a, double b, int N);

double simpson( double(*F)(double), double a, double b, int N);

double gaussian (double (*F)(double), double  a,  double  b,  int N,  int Ng);

double func(double x);

double functay(double x);

int main(){
  double a = 0.0, b = 1.0, sum_trapezoidal, sum_simpson, sum_gauss;
  double val = 1.618194443708e+00;  // given value
  double err_trapezoidal, err_simpson, err_gauss;
  int N = 3;
  cout << setiosflags(ios::scientific);
  // Compute the sum of 15 sub-intervals, from x = 0 to x = 15.
  // Each sub-interval has the same length: xb - xa = 1.
  do{
    sum_trapezoidal = sum_trapezoidal + trapezoidal(func, a, b, N);
    sum_simpson = sum_simpson + simpson(func, a, b, N);
    sum_gauss = sum_gauss + gaussian(func, a, b, 1, 3);
    cout << "x = " << std::setprecision(1) << b << "    ";
    cout << std::setprecision(4) << sum_gauss;
    cout << "    " << std::setprecision(4) << sum_simpson;
```

```
    cout << "     " << std::setprecision(4) << sum_trapezoidal << endl;
    a++;
    b++;
  }
  while (b <= 15.0);
  // Error is calculated only for x = 15;
  cout << endl;
  err_trapezoidal = abs(sum_trapezoidal - val);
  err_simpson = abs(sum_simpson - val);
  err_gauss = abs(sum_gauss - val);
  cout << "Err =          " << std::setprecision(4) << err_gauss;
  cout << "     " << err_simpson << "     " << err_trapezoidal << endl;
  return 0;
}
//Trapezoidal Quadrature
double trapezoidal(double (*F)(double), double a, double b, int N){
  double h = (b - a) / N, sum;
  int i;
  sum = 0.5 * (F(a) + F(b));
  for(i = 1; i < N; i++){
    sum = sum + (F(a + i * h));
  }
  return h * sum;
}
// Simpson Quadrature
double simpson(double (*F)(double), double a, double b, int N){
  double h = (b - a) / N, sum;
  int i, j;
  sum = F(a) + F(b);
  for(i = 1; i < N; i++){
    if((i % 2) == 0) j = 2;
    else j = 4;
    sum = sum + j * (F(a + i * h));
  }
  return sum * h / 3.0;
}
// Gaussian Quadrature
double  gaussian (double  (*F)(double), double  a,  double  b,  int N,  int Ng){
  double w[32], x[32];
  int k, i;
  double x0, x1;
  double sum = 0.0, sumk;
  if(Ng == 2){
    x[0] = -1.0 / sqrt(3.0) ; x[1] =  1.0 / sqrt(3.0);
    w[0] =  1.0; w[1] = 1.0;
  }
  if(Ng == 3){
    x[0] = -sqrt(3.0 / 5.0); x[1] = 0; x[2] = sqrt(3.0 / 5.0);
    w[0] =  5.0 / 9.0; w[1] = 8.0 / 9.0; w[2] = 5.0 / 9.0;
  }
  double h = (b - a) / N;
  for (i = 0; i < N;  i++){
    sumk  = 0.0;
    x0 = a + (i * h); x1 = b - ((N - 1 - i) * h);
    for (k = 0; k < Ng; k++){
      sumk = sumk + w[k] * F((0.5 * (x1 - x0) * x[k]) + (0.5 * (x1 + x0)));
    }
    sum = sum + sumk;
  }
  return  0.5 * (x1 - x0) * sum;
}

double func(double x){
  if (x > 1.e-16) return sin(x) / x;    // use for x != 0
  else return 1.;                       // use for x = 0 -> extension in continuity
}
```

Listing 1: code in C++ language.

2 Output

The output produced is the following. From left to right, the columns represent Gauss Quadrature, Simpson Quadrature and Trapezoidal Quadrature. Here, the error is calculated for x = 15 by comparison with the given exact value of the integral.

```
x = 1.0e+00    9.4608e-01    8.4699e-01    9.4329e-01
x = 2.0e+00    1.6054e+00    1.4477e+00    1.6014e+00
x = 3.0e+00    1.8487e+00    1.6793e+00    1.8454e+00
x = 4.0e+00    1.7582e+00    1.6077e+00    1.7571e+00
x = 5.0e+00    1.5499e+00    1.4225e+00    1.5508e+00
x = 6.0e+00    1.4247e+00    1.3056e+00    1.4262e+00
x = 7.0e+00    1.4546e+00    1.3268e+00    1.4555e+00
x = 8.0e+00    1.5742e+00    1.4320e+00    1.5739e+00
x = 9.0e+00    1.6650e+00    1.5158e+00    1.6641e+00
x = 1.0e+01    1.6583e+00    1.5137e+00    1.6576e+00
x = 1.1e+01    1.5783e+00    1.4439e+00    1.5784e+00
x = 1.2e+01    1.5050e+00    1.3769e+00    1.5057e+00
x = 1.3e+01    1.4994e+00    1.3690e+00    1.5000e+00
x = 1.4e+01    1.5562e+00    1.4181e+00    1.5563e+00
x = 1.5e+01    1.6182e+00    1.4742e+00    1.6177e+00

Err =          3.0772e-08    1.4395e-01    4.9670e-04
```

Figure 1: output of the C++ code.