
**Silesian University of Technology, Gliwice**

Faculty of Automatic Control, Electronics and Computer Science

Semester 4

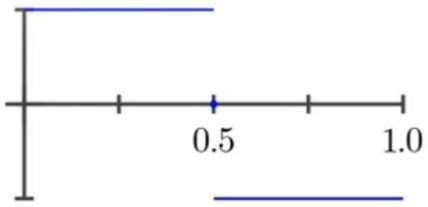
# Approximating piecewise functions using the Fourier method

Group 2, Section 2

- Mateusz Nowotnik
- Dawid Tomala

# 1. Abstract

The Fourier series uses sines and cosines to represent known functions. Originally it was proposed by the French mathematician Joseph Fourier in the early 1800s, while studying the heat equation. To demonstrate, let us consider the heat distribution in time after connecting two rods together, one at 1C, and one at -1C.




$$\begin{cases} 1 & \text{if } x < 0.5 \\ 0 & \text{if } x = 0.5 \\ -1 & \text{if } x > 0.5 \end{cases}$$

fig.1: heat distribution at t=0

As we can clearly see, the initial distribution is a discontinuous function. But thanks to the Fourier series, we can express it using an infinitesum of cosine waves:

Eq.1 $$f(x) = \sum_{i=1}^{\infty}(\cos(n * \left(\frac{\pi}{L}\right) * x)$$

As n approaches infinity, the resulting function looks more and more like the original step function.

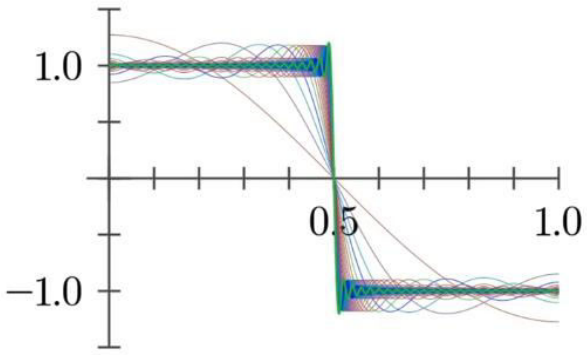



fig.2: f(x) represented with Eq.1

The above example is given only to demonstrate in an easy to understand way how the Fourier series can be used. A more general way of using this method is represented by the equation:

Eq.2 $$f(x) = \frac{A_0}{2} + \sum_{n=1}^{\infty} A_n \cos\left(\frac{n\pi x}{\frac{P}{2}}\right) + \sum_{n=1}^{\infty} B_n \cos\left(\frac{n\pi x}{\frac{P}{2}}\right)$$

Although this equation is written for periodic functions, we can easily convert it to deal with non-periodic functions (which we are mostly concerned here) simply by reflecting the function on the specified interval (0, L) over the y-axis (which creates an even function), or the origin (which creates an odd function). It's also important to note that f(x) *must* be integrable if we want to compute the coefficients of the Fourier series:

$$A_n = \frac{2}{P}\int_{-P/2}^{P/2} f(x)\cos\left(\frac{n\pi x}{\frac{P}{2}}\right)dx, \quad n = 0,1,2 \ldots$$

$$B_n = \frac{2}{P}\int_{-P/2}^{P/2} f(x)\sin\left(\frac{n\pi x}{\frac{P}{2}}\right)dx, \quad n = 1,2,3 \ldots$$

If the function is even, only the A terms will be non-zero and the B terms will only be non-zero if the function is odd. If the function is neither, the Fourier series will contain both.

## 2. Implementation

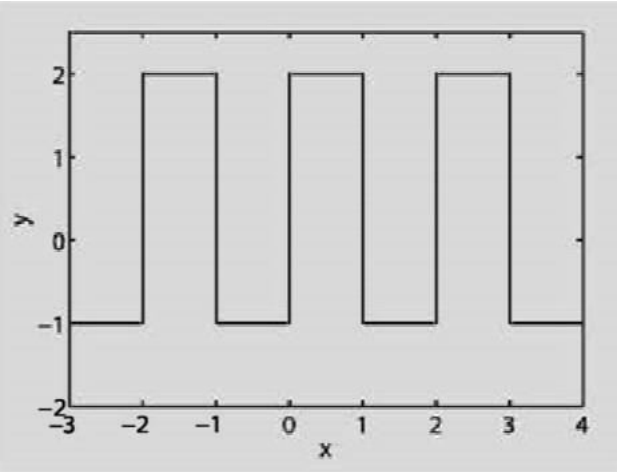
```
#include <iostream>
#include <math.h>
using namespace std;

double integrand(double x, double y, int n, int integ)
{
    double pi = atan(1)*4;

    if (integ == 1) {
        if (y == 1) // x > a && x < b
            return x*sin(pi*n*x);
        else
            return y*sin(pi*n*x);
    }
    if (integ == 2) {
        if (y == 1)
            return x*cos(pi*n*x);
        else
            return y*cos(pi*n*x);
    }
    if (integ == 3) {
        if (y == 1)
            return x;
        else
            return y;
    }
}

// ---------------------------------------- BEGIN OF INTEGRATION-------
double *XsForRect(double a, double h, int m)
{
    double *x = new double[m];

    for (int i = 1; i <= m; i++) {
        x[i-1] = a+((double)i - (0.5))*h;
    }

    return x;
}

// using rectangular method for calculating integrals (n=0,m=20)
double integralRect(double a, double b, double y, int n, int integ)
{
    double h, *x;
    int m = 20;
    double sum = 0;
    h = (b-a)/m;
    x = XsForRect(a, h, m);

    for (int i = 0; i < m; i++) {
        sum += integrand(x[i], y, n, integ);
    }
    return sum*h;
```

```
}
// ---------------------------------------- END OF INTEGRATION----------

double funSum(double x, double L, double A[], double B[], int n)
{
    double pi = atan(1)*4;
    double sum = 0;

    for (int i = 1; i < n; i++) {
        sum += A[i]*sin((pi*i*x)/L) + B[i]*cos((pi*i*x)/L);
    }

    return sum;
}


int main()
{
    double pi = atan(1)*4;
    int n;
    double L, Ai, Bi, B0;
    int subinter; //how many sub-intervals in the interval (a,b)

    cout << "How many loops (n): ";
    cin >> n;
    double A[n] = {0};
    double B[n] = {0};

    cout << "Give the value of sub-intervals: ";
    cin >> subinter;
    double intervs[subinter][2];
    double y[subinter] = {0};

    cout << "Give each sub-interval and solution: ";
    for (int i = 0; i < subinter; i++) {
        cin >> intervs[i][0]; // a
        cin >> intervs[i][1]; // b
        cin >> y[i];
    }

    L = (intervs[subinter-1][1]-intervs[0][0])/2; // (b-a)/2

    for (int i = 1; i < n; i++) {
        Ai = 0; Bi = 0; B0 = 0;
        for (int j = 0; j < subinter; j++) {
            Ai += integralRect(intervs[j][0], intervs[j][1], y[j], i, 1);
                  Bi += integralRect(intervs[j][0], intervs[j][1], y[j], i,
    2);
                  B0 += integralRect(intervs[j][0], intervs[j][1], y[j], i,
    3);
              }
              A[i] += Ai/L;
              B[i] += Bi/L;
          }

          cout << B0/(2*L) + funSum(0.5, L, A, B, n) << endl << endl;


```

```
111.          for (int i = 1; i < n; i++) {
112.              cout << A[i] << endl;
113.          }
114.          cout << "---------" << endl;
115.          for (int i = 1; i < n; i++) {
116.              cout << B[i] << endl;
117.          }
118.
119.          return 0;
120.      }
```

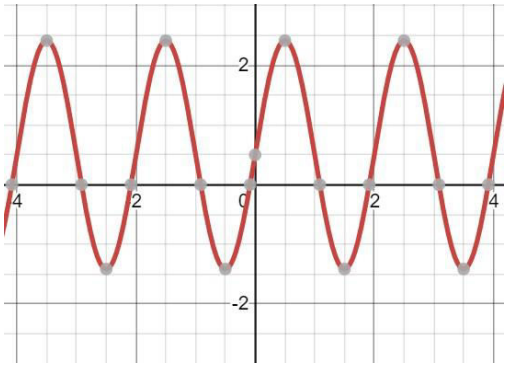
For function

$$f(x) = \begin{cases} -1 & -1 < x < 0 \\ 2 & 0 < x < 1 \end{cases}$$

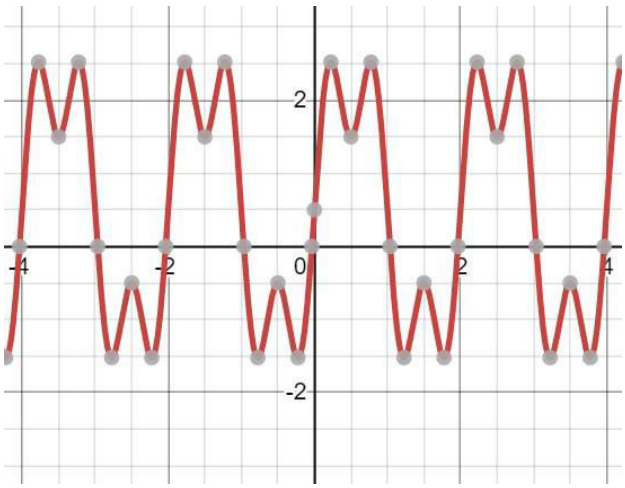



The following figure shows the approximation

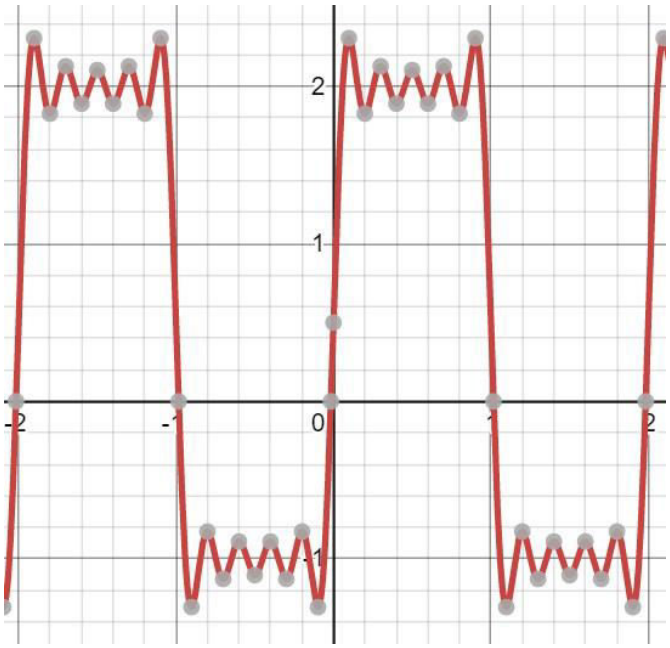
**n = 2**



**n = 4**

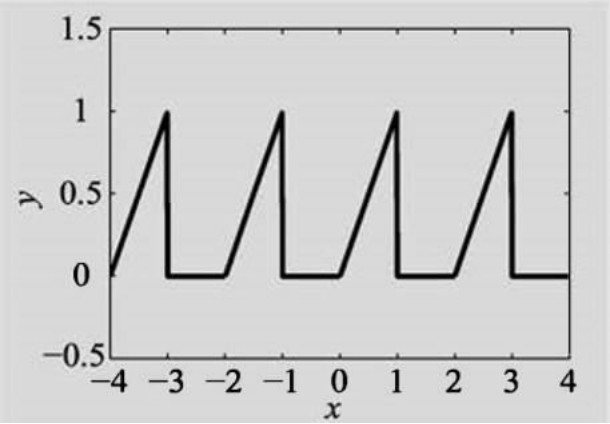



**n = 10**



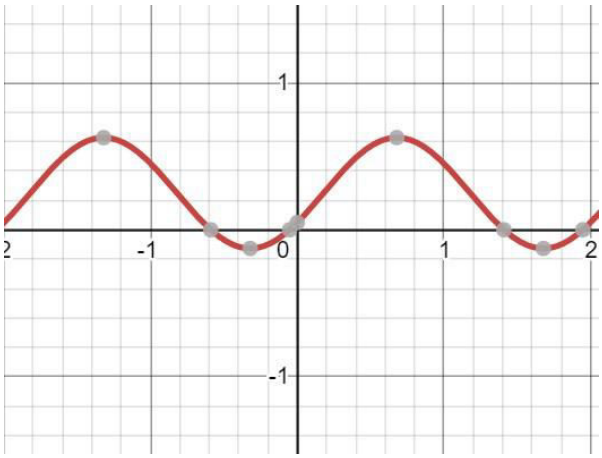
For function

$$f(x) = \begin{cases} x & 0 < x < 1 \\ 0 & 1 < x < 2 \end{cases}$$

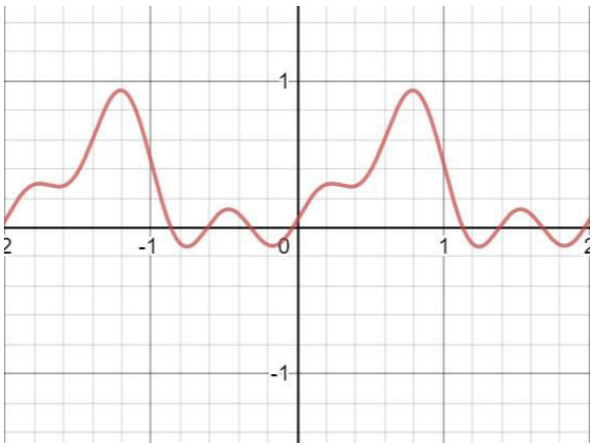



The following figures show the process of approximation

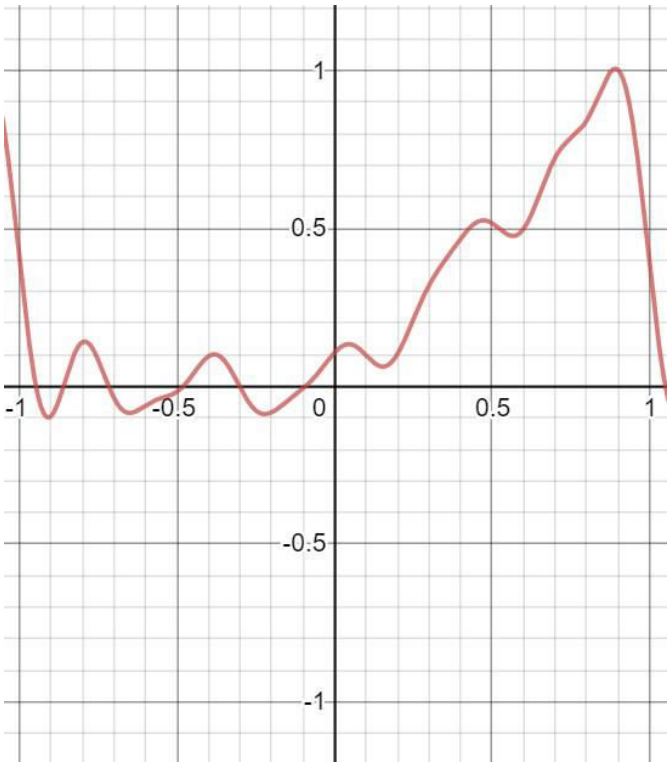
**n = 2**




**n = 4**



**n = 10**




## 3. Practical uses

- Heat equation
- Jpeg compression
- All areas of signal processing

## 4. Conclusions

In this method we used composite rectangular method for numerical integration, since the other integration methods did not give a good approximation.
As for the Fourier method, we have found it fairly easy to understand and implement.

## 5. Bibliography

1. Applied Numerical Analysis 7th edition, Gerald Wheatley
2. Numerical Methods for Engineers and Scientists 3rd edition, Amos Gilat