Experiment - 5
Numerical Integration

Author: Anıl Karatay
anilkaratay@iyte.edu.tr
(Duration: 105 minutes)

Purpose: The aim of this lab is to find an area in a given interval using rectangular and trapezoidal integration methods.

Introduction

Numerical integration is the approximate computation of an integral using numerical techniques. There are a wide range of methods available for numerical integration. Two of these are the rectangular and the trapezoidal methods. The rectangular method is one of the simplest integration techniques that approximates the integral of a function with a rectangle. It uses rectangles to approximate the area under the curve, see Figure 1. The width of the rectangle is determined by the integration range. The integration interval can be divided into n smaller intervals of equal length and n rectangles are used to approximate the integral. In a fixed integration range, the more rectangles are used, the more accurate the approach; each rectangle narrows, and the height of the rectangle better captures the values of the function in that range.

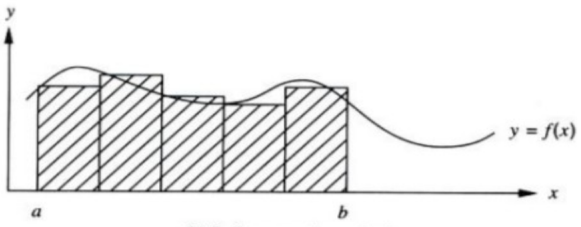

Figure 1: Left rectangular method

Trapezoidal method is another numerical techniques to approximate a definite integral. In this method, region under the graph is calculated by regarding it as a combination of many trapezoid panels. As number of panels increases, approximation error decreases. Region under the function f(x) between a and b (assuming no singularity exists in that range) approximated by the trapezoids and can be found by calculating the area of these trapezoids.

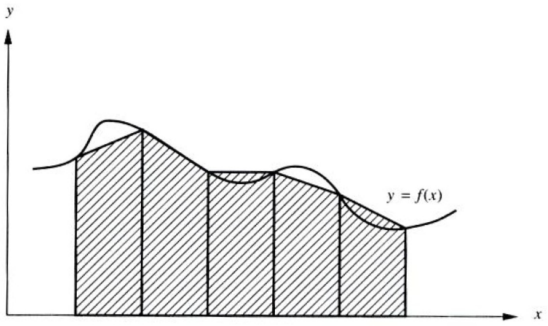

Figure 2: Trapezoidal method

Problem Statement

In this lab, integration of two different functions and their differences over a certain interval will be addressed. With the help of rectangular and trapezoidal panels, these integrals will be calculated and the performances will be compared. What is expected from the program you will write is to calculate the integrals of the functions given in Figure 3 using rectangular and trapezoidal methods for both functions between 0-30 and calculate the corresponding area by finding the differences of these definite integrals. By repeating this process with different number of panels, we will observe which method is more accurate.

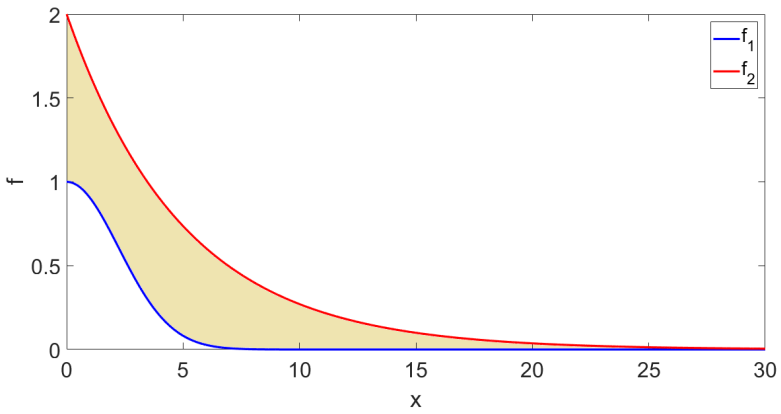

Figure 3: Graphical illustration of the functions

The mathematical expression of functions f_1 and f_2 is as in Eq. 1 and 2.

$$f_1(x) = e^{-0.1x^2} \tag{1}$$

$$f_2(x) = 2e^{-0.2x} \tag{2}$$

What is expected from you during the lab can be summarized as follows.

- Define interval and the functions.
- Write a function that calculates a definite integral with rectangular method.
- Write another function that calculates a definite integral with trapezoidal method.
- Generate number of panels from 50 to 1000 using dynamic memory allocation in a function.
- Calculate the integral of f_1, f_2 and $f_2 - f_1$ with both rectangular and trapezoidal methods. There must be 6 results for each panel number.
- Print the results in main().

Lab Procedure

- Define start and end points of the interval with define statements (be careful about integer division). (2 pts.)
- Declare f_1 and f_2 as separate functions. You must follow the prototypes below. (8 pts.)

```
double f1(double x);//returns the value of f1
double f2(double x);//returns the value of f2
```

- Write a function that calculates a definite integral of any function, f, with left rectangular method. Follow the prototype below. (25 pts.)

```
double rectangular(double (*f)(double), int n)//f is function, n is number of panels
```

- Write a function that calculates a definite integral of any function, f, with trapezoidal method. Follow the prototype below. (25 pts.)

```
double trapezoidal(double (*f)(double), int n)//f is function, n is number of panels
```

- Write a function that generates the number of panels in a dynamic array and it returns a pointer integer. The number of panels should increase by 50 from 50 to 1000. Note that, you have to use **dynamic memory allocation** in this function. (25 pts.)

```
int *number_of_panels_generator()//from 50 to 1000
```

It is imperative to follow the prototype above. If you cannot complete this function, you can do this operation in main() just to continue, provided that you forfeit the score of this section.

- Print the results in main() for each panel numbers. Using printf() anywhere other than main() is strictly forbidden. (15 pts.)

	Rect. f1	Trap. f1	Rect. f2	Trap. f2	Rect. area	Trap. area
Number of panels:50	3.102496	2.802496	10.585693	9.987180	7.483197	7.184684
Number of panels:100	2.952496	2.802496	10.277461	9.978205	7.324966	7.175709
Number of panels:150	2.902496	2.802496	10.176047	9.976542	7.273551	7.174047
Number of panels:200	2.877496	2.802496	10.125589	9.975961	7.248093	7.173465
Number of panels:250	2.862496	2.802496	10.095394	9.975691	7.232898	7.173196
Number of panels:300	2.852496	2.802496	10.075297	9.975545	7.222801	7.173049
Number of panels:350	2.845353	2.802496	10.060959	9.975457	7.215606	7.172961
Number of panels:400	2.839996	2.802496	10.050214	9.975400	7.210218	7.172904
Number of panels:450	2.835829	2.802496	10.041862	9.975360	7.206033	7.172865
Number of panels:500	2.832496	2.802496	10.035183	9.975332	7.202688	7.172837
Number of panels:550	2.829768	2.802496	10.029722	9.975311	7.199953	7.172816
Number of panels:600	2.827496	2.802496	10.025172	9.975296	7.197676	7.172800
Number of panels:650	2.825573	2.802496	10.021323	9.975283	7.195750	7.172788
Number of panels:700	2.823924	2.802496	10.018024	9.975274	7.194100	7.172778
Number of panels:750	2.822496	2.802496	10.015167	9.975266	7.192671	7.172770
Number of panels:800	2.821246	2.802496	10.012666	9.975259	7.191421	7.172764
Number of panels:850	2.820143	2.802496	10.010461	9.975254	7.190318	7.172758
Number of panels:900	2.819162	2.802496	10.008500	9.975249	7.189338	7.172754
Number of panels:950	2.818285	2.802496	10.006746	9.975246	7.188461	7.172750
Number of panels:1000	2.817496	2.802496	10.005168	9.975242	7.187672	7.172747

Figure 4: Screenshot of the output