Final Exam

(Duration 180 mins)

Author: Mehmet Çalı
13.07.2020

Purpose: The aim of the program that will be implemented in the final laboratory is to calculate the integral of a function within regions which are separated by its roots.

Problem Statement

The function to calculate the integral is given in Equation 1.

$$f(x) = \sin\left(\frac{\pi}{2}x^2\right) + 0.2 \qquad 1$$

The integral will be calculated between an interval of (a,b). The interval boundaries, a and b, will be provided by the user with conditions of $a \geq 0$ and $b \leq 10$ (the user can enter any two numbers between 0 to 10). It is given that, within any interval inside (0,10) the minimum distance between successive two roots of function f(x) is 0.05. In other words, maximum one root can exist between $(\tau, \tau + 0.05)$, as long as the sub-interval $(\tau, \tau + 0.05)$ is within (a,b). Therefore, the program can sequentially search whether a root exist in $(\tau, \tau + 0.05)$. The following example demonstrates the integral calculation and roots for (a,b)=(1,2.2).

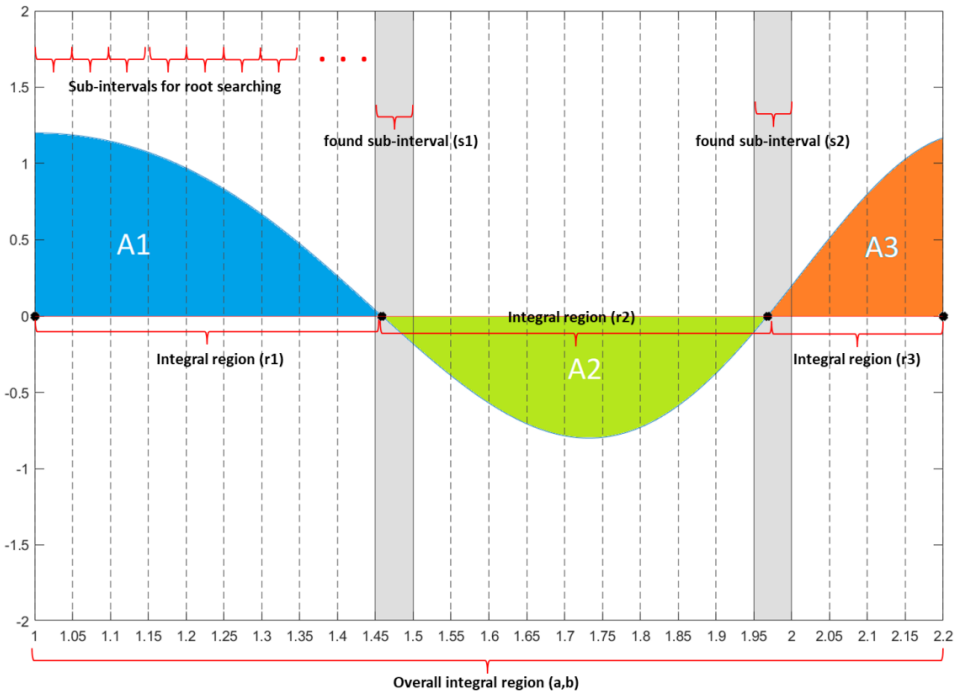

Figure 1 Integral regions ((1, 1.4588), (1.4588,1.96769) and (1.96769,2.2)) are specified between the interval initial point (a=1), roots and interval final point (b=2.2)

If a root exist in a sub-interval, the program should find the root using trisection method for that sub-interval $(\tau, \tau + 0.05)$. Trisection method is very similar to bisection method with only difference of dividing the search region into 3 equal spaces instead of 2. Sub-intervals that contain the roots are demonstrated with s1 and s2 in Figure 1. After finding precise location of the roots in these sub-intervals (1.4588 and 1.9677 for the given example), integral of the regions which are separated by the roots should be calculated. Note that there may be more or less than 2 roots depending on the interval selection.

The exam procedure clearly explains every step that should be taken. **You must strictly follow the methods described in the exam procedure to get any scores. For the functions that you have written, you must print one of the sample outputs displayed in each step at exam procedure. The print functions should be called in the main.**

Exam Procedure

1. Write a function to calculate the output of Equation 1 for any given x. **(5 pt)**

Sample Outputs:	`f(0.00)=0.200000` `f(2.20)=1.167617`

2. Write a function with following prototype to print elements in an array until a zero valued element is encountered. **(5 pt)**

```
void printArray(double inputArray[50]);
```

3. Write a function with the following prototype to notify whether a root is present in a specified sub-interval. The function output should be 1 or 0 according to the fact that root exist or not respectively. **(5pt)**

```
int isRootPresent(double t_low, double t_upp);
```

Note that, t_{low} and t_{upp} are boundaries of sub-intervals which are demonstrated as $(\tau, \tau + 0.05)$ in problem description. When calling this function, $t_{upp} - t_{low}$ should always be 0.05, so you may use one parameter as well (which is t_{low}). Also you may include function pointer to the arguments.

HINT: Remember bisection method conditions.

Sample Outputs:	`isRootPresent(1.95,2.00)=1`
	`isRootPresent(2.05,2.10)=0`
	`isRootPresent(3.75,3.80)=1`

4. Write a function named "`findLowerBoundaries`" that searches every sub-interval, and stores the lower boundary of the sub-intervals to an array if the corresponding sub-intervals include a root. The search interval is bounded by (a,b) and sub-interval length is 0.05 as described before. You can use the following prototype: **(15pt)**

```
void findLowerBoundaries(double lower_boundaries[50],double a,double b);
```

The array may be fixed size with a length of 50 and should be declared in the main. At the beginning of the function, you must initialize every element in the array "lower_boundaries" with zero; so that, you may process this array until you meet a zero in the next steps.
You should print the resulting array (lower_boundaries) by calling the function "printArray" in the main.

Sample Outputs:

```
For (a,b)=(0.00,2.20), lower boundaries are:
1.450000
1.950000
```

```
For (a,b)=(2.00,3.60), lower boundaries are:
2.450000
2.800000
3.150000
3.400000
```

If you failed to implement Step 3 and/or Step 4 you can continue the next steps by storing 1.45 and 1.95 to the array named "lower_boundaries" (rest of the elements are 0).

5. Write a function named "trisection" that finds a root inside a given interval using trisection method. Trisection method divides the input region (t_{low}, t_{upp}) into three regions $(t_{low}, y_1), (y_1, y_2)$ and (y_2, t_{upp}). Then, iteratively keeps dividing the one that includes the root. Exit condition should be set as $|y_1 - y_2| < \epsilon$. Output can be selected as y_1. Epsilon should be set to 0.00001 in the main. **(20 pt)**

Sample Outputs:

```
trisection(f,2.45,2.5,0.00001)=2.475522
trisection(f,1.95,2.0,0.00001)=1.967680
```

If you failed to implement Step 5 you can continue the next steps by replacing trisection with bisection method that exists in the lecture notes.

6. Write a function named "findRoots" that finds the roots of the sub-intervals whose lower bounds are stored at array named "lower_boundaries". (Hint: Loop through the array "lower_boundaries" until you encounter a zero) Remember that, upper boundaries equal to $t_{low} + 0.05$. To find the roots, call the root finding function that you implemented in the previous step. The function named "findRoots" should output another array (double roots[50];) that stores the roots of the function. The elements of array named "roots" should also be initialized as 0. In the next steps you will process the array until you meet a zero. You may use the following prototype: **(15 pt)**

```
void findRoots(double (*f)(double), double lower_boundaries[50],
               double roots[50], double epsilon));
```

Sample Outputs:

```
For (a,b)=(0.00,2.20), roots are:
1.458832
1.967680
```

```
For (a,b)=(2.00,3.60), roots are:
2.475522
2.805677
3.182472
3.445549
```

If you failed to implement Step 5 and/or Step 6 you can continue the next steps by storing 1.4588 and 1.9677 to the array named `roots` *(rest of the elements are 0).*

7. Write a function named "`leftRectangle`" that calculates the integral of the curve bounded by two parameters. Use left adjust rectangular (left rectangle) numeric integration method. You **must** use following prototype: **(15 pt)**

```
double leftRectangle(double (*f)(double)), double x1, double x2, double step);
```

where f is function name, (x1,x2) is function boundary, step is the width of the rectangle. Use the step size of 0.00001.

Sample Outputs:

```
leftRectangle(f,1.4588,1.9677,0.00001)=-0.261080
leftRectangle(f,2.8056,3.1824,0.00001)=0.283444
```

8. Write a function named "`saveIntegrals`". The function "`saveIntegrals`" must call "`leftRectangle`" for each region separated by the roots. For example, regions are (1, 1.4588), (1.4588,1.9677) and (1.9677,2.2) for the case given in Figure 1. (Be sure not to exclude outermost regions). The function should get the integrals and saved in array (`double integrals[50];`) The elements of array named `integrals` should also be initialized as 0 at the beginning of the function. **(20 pt)**

```
void saveIntegrals(double roots[50], double integrals[50], double a,
                   double b, double step)
```

Sample Outputs:

```
For (a,b)=(0.00,2.20), integral of the regions are:
1.001286
-0.261080
0.155494
```

```
For (a,b)=(2.00,3.60), integral of the regions are:
0.378021
-0.170046
0.283444
-0.135559
0.113040
```

Bonus: Write a function named "`getMaxIntegral`" that outputs the region boundaries that includes maximum integral **(15 pt).** The function should also output total integral over (a,b) **(5 pt)**

Sample Outputs:

```
Total integral over (a,b)=(0.00,2.20) is: 0.895700
Maximum integral region inside (a,b)=(0.00,2.20) is: (0.000000,1.458832)
Total integral over (a,b)=(2.00,3.60) is: 0.468025
Maximum integral region inside (a,b)=(2.00,3.60) is: (2.000000,2.476139)
```