# APS 105 — Computer Fundamentals

Lab 4: Loops and Functions

Winter 2019

**Important note:** You must use the submit command to electronically submit your solution by the end of your lab session.

---

This lab will give you experience writing software that makes use of a loop to *numerically* compute the integral of a polynomial function:

$$f(x) = 2x^2 - 7x - 2$$

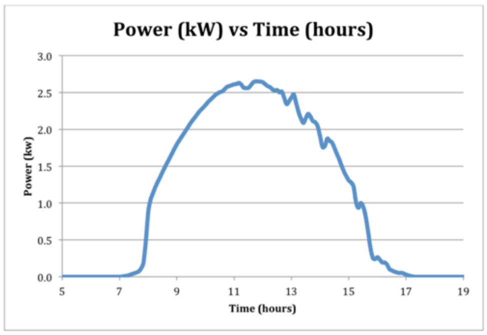
using an approach called the *Riemann* or *rectangle* method. You will also write your own C-language *functions*.
Your solution will be marked by your TA during your scheduled lab period, and will also be submitted electronically by the end of your scheduled lab period. Your TA will mark your solution based on its style, its use of functions, and your verbal answers to a few questions.

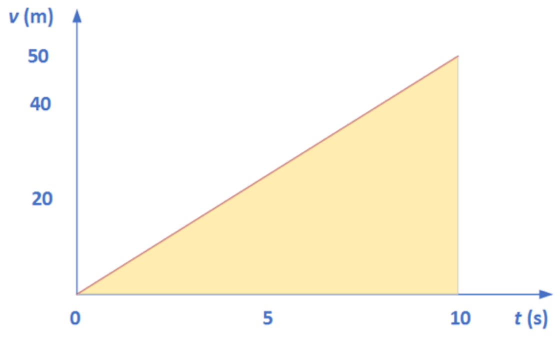
## Background: Using Software to Compute Integrals

As you have learned in your Calculus course last term, the integral of any mathematical function f(x) can be used to compute the area under the curve over a specific range of x values, assuming *f(x) is continuous in that range (it is definite)*.

However, in the real world, it is often not possible to determine an expression for the integration of various functions, so in practice engineers often approximate the area under a curve using computers.

For example, we use the area under the curve of *Power Vs. Time* produced by photovoltaic panels to approximate the amount of energy produced during certain period. Likewise, the area under the curve of *Velocity Vs. Time* gives us the distance traveled.





Consider the function $\boldsymbol{f(x) = 2x^2 - 7x - 2.}$

Using calculus, the exact integral of the f(x) in the range ***-2.4 to 1.6***, is equal to:

$$\int_{-2.4}^{1.6} 2x^2 - 7x - 2\,dx = \frac{2x^3}{3} - \frac{7x^2}{2} - 2x\ \Big|_{-2.4}^{1.6} \approx 15.1467$$

We can produce an estimate of this value, however, without using calculus. Figures 1 to 3, below shows the function f(x) over the range [−2.4, 1.6]. We can estimate the area under f(x) by creating 10 rectangles (or in general ***n*** rectangles) evenly spaced over the range of integration (i.e. -2.4 to 1.6). The height of each rectangle is determined by the value of f(x) *either* at the *intersect of the left corner, right cornet, or middle of the rectangles.*

For example, using the left corner intersect (Figure-1), the contribution of the left-most rectangle is:

**26.32 * 0.4 = 10.528**, *where the height is equal to f(-2.8)=26.32*

*the width is equal to (1.6-(-2.4))/10 = 0.4*

The total area under the curve is the sum of all individual rectangles (here 10). This total would be an approximation for the actual exact integral. Note that depending on the function, sections of the curve may fall below the x-axis. Clearly, the height and thus the area for those regions of the graph would result in a negative value. (these would be all rectangles shaded in yellow). Still, the total area would be calculated as the net sum of all sub areas (positive and negative). For example, in figure-1a, the total area in the range of 1 to 2, would be negative. This is in agreement with exact integral formula evaluation for the given curve (see above).

Using this example (Riemann's left side method, with 10 rectangles), the integral is approximated to be **22.24.** It is not the exact correct answer since the rectangles both over- and under-estimate the area depending on whether they are higher or lower than the curve at any point.

As you might imagine, if more rectangles are used (with larger n) each rectangle becomes narrower and the estimate becomes more precise. Figure-1b shows the rectangle approximation of the same integral, but with n = 20 rectangles. Observe how the rectangles are narrower and more closely fit the curve of the function. This means else error. This approximation gives a value of **18.64**, much closer to the true value!

**Note**: You are encouraged to work out this example on paper to arrive at this value.

In general, using mid-point approximation provides less error; however, the shape of the function also can influence which method (left, right or mid-point) is better suited for the application.

Figures 2 & 3, show the right and mid-point depiction of the Riemann's approximation.

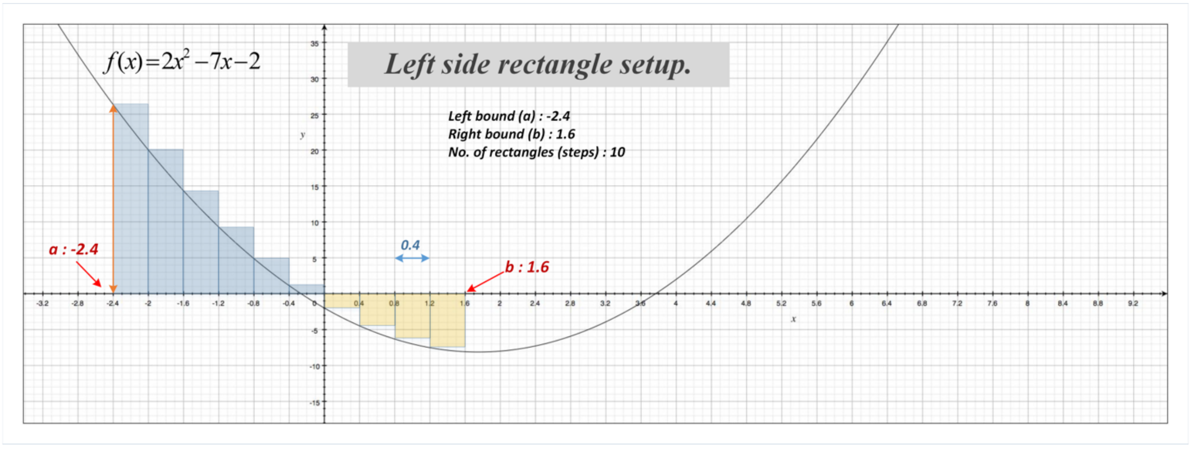


**Figure-1a** Riemann Left side (n=10) Integral approximation

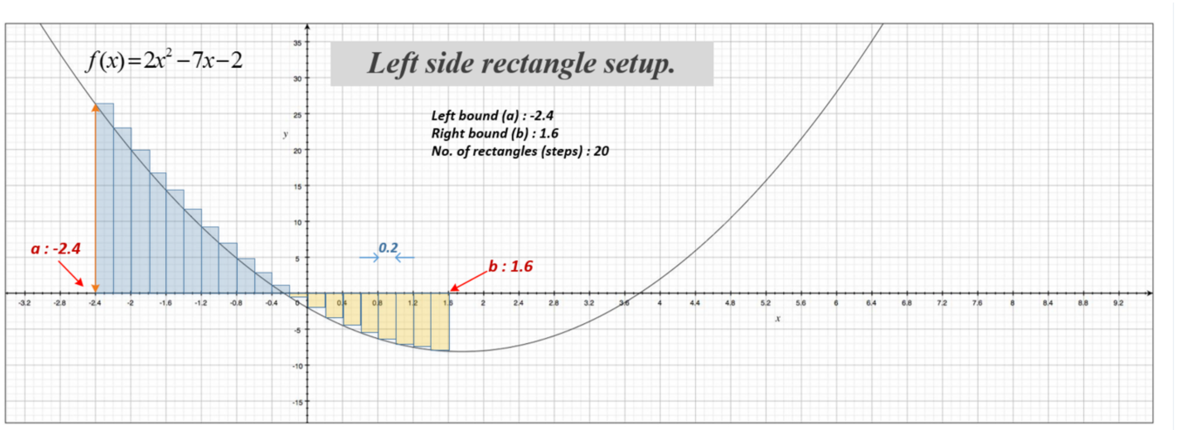


**Figure-1b** Riemann Left side (n=20) Integral approximation

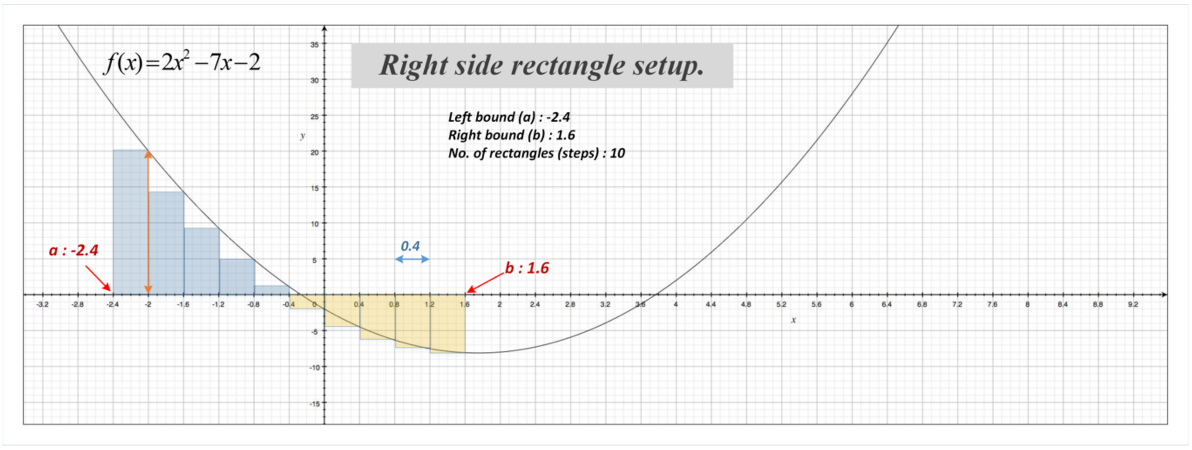


**Figure-2** Riemann Right side (n=10) Integral approximation

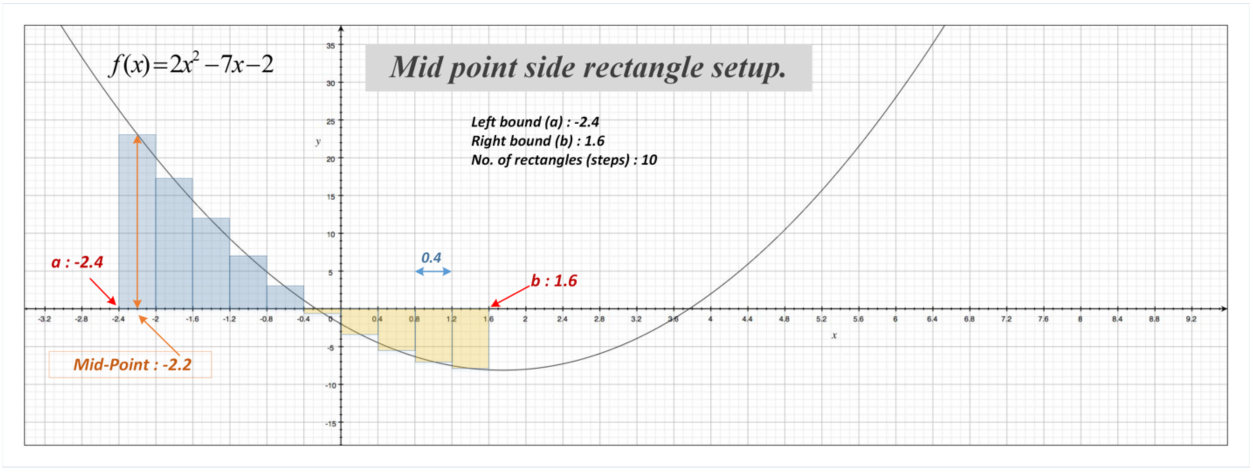


**Figure-3** Riemann Mid-point (n=10) Integral approximation

## Numerical Integration Program

You are to write a C program, in a file called *Lab4.c*, which estimates the integral of function:

$$f(x) = 2x^2 - 7x - 2$$

over the range [a,b], using n rectangles.

The C code should contain two functions: one to receive and evaluate inputs from user, and the second to evaluate the heights, or the actual function f(x) and return the result. The body of the ***`main(),`*** would contains all the necessary code to process the data received or initialized by these two functions, three (3) loops each to calculate the approximated area using left, right, and mid-point methods, and finally output the proper messages with the result as it will be shown later.

### Function to evaluate inputs:

Your code must have the provision to receive three inputs from the user: two double values to specify the range for integration, and one integer number to identify the number of rectangles used for estimation with following prototype:

```
bool validateInput(double left, double right, int n)
```

Note that the function only returns one value of type Boolean. It is used to evaluate the data entered. This function has three parameters which are passed to it from the main with the types indicated above.

The To avoid large integral calculation, the range should be limited to ***+/-10.0***. That is left boundary could go down to ***-10.0***, while the right one could reach ***10.0***. There is no limit on the integer number used to construct the rectangles, other than this number must be a nonzero positive integer.

To sum up if any of the following conditions is true, this function should print a message: **"invalid inputs..."**, and return a false value to the main code, where this function is called again.

```
 a<-10.0, or b>10.0, or b<=a, or number of rectangles <=0
```

### Function to evaluate f(x):

This is the module which evaluate the height of the rectangles. This is done simply by evaluating the function f(x) for various values of x. You should use `pow()` for evaluation. The function prototype is: `double evalFunc(double x)`

***NOTE:*** *Your TA will explicitly check to see that you used C-language functions, and your grade will be lowered if your code does not do so.*

## The main code:

1. The user will first enter **a** (the left side boundary), **b** (the right side boundary), and **n** (the number of triangles to be used). **a** & **b** are real numbers, while **n** is an integer. This is done by calling the function `GetInputs()`. This function evaluates the input data and as described above, would be recalled if the conditions set are not met.

```
Enter the Left boundary: a, Right boundary: b, and No. of rectangles to use.
To exit enter 0 0 0
3 -2 39
Invalid inputs...

Enter the Left boundary: a, Right boundary: b, and No. of rectangles to use.
To exit enter 0 0 0
-2 -2 51
Invalid inputs...

Enter the Left boundary: a, Right boundary: b, and No. of rectangles to use.
To exit enter 0 0 0
-2 2 0
Invalid inputs...

Enter the Left boundary: a, Right boundary: b, and No. of rectangles to use.
To exit enter 0 0 0
```

2. The main also should have a provision for the user to quit entirely if the user enter three zeros as follows:

```
Enter the Left boundary: a, Right boundary: b, and No. of rectangles to use.
To exit enter 0 0 0
0 0 0
Invalid inputs...

Terminating main!
```

3. The calculation of total area under the curve is done in independent loops for each of the left, right, and mid-point approximation. That is your code should have three loops for area calculation. These loops work in very similar manner; in that they should call the function evaluates the height or f(x), calculate and sum up the area.
4. Upon proper data entry, the code then provides the following sample outputs and exits.

***Note****: format the output numbers as shown below.*

```
Enter the Left boundary: a, Right boundary: b, and No. of rectangles to use.
To exit enter 0 0 0
-2.4 1.6 10
With Step Size: 0.4000
The approximate integral of the f(x) = 2(x^2)-7x-2
Bound between -2.40 and 1.60, using 10 rectangles is as follows

Mid point evaluation approximate: 15.0400
Left point evaluation approximate: 22.2400
Right point evaluation approximate: 8.4800



Enter the Left boundary: a, Right boundary: b, and No. of rectangles to use.
To exit enter 0 0 0
-2.4 1.6 20
With Step Size: 0.2000
The approximate integral of the f(x) = 2(x^2)-7x-2
Bound between -2.40 and 1.60, using 20 rectangles is as follows

Mid point evaluation approximate: 15.1200
Left point evaluation approximate: 18.6400
Right point evaluation approximate: 11.7600



Enter the Left boundary: a, Right boundary: b, and No. of rectangles to use.
To exit enter 0 0 0
-2.4 1.6 755
With Step Size: 0.0053
The approximate integral of the f(x) = 2(x^2)-7x-2
Bound between -2.40 and 1.60, using 755 rectangles is as follows

Mid point evaluation approximate: 15.1466
Left point evaluation approximate: 15.2378
Right point evaluation approximate: 15.0556



Enter the Left boundary: a, Right boundary: b, and No. of rectangles to use.
To exit enter 0 0 0
-6 7.6 81
With Step Size: 0.1679
The approximate integral of the f(x) = 2(x^2)-7x-2
Bound between -6.00 and 7.60, using 81 rectangles is as follows

Mid point evaluation approximate: 333.2268
Left point evaluation approximate: 337.7570
Right point evaluation approximate: 329.0799
```

## Submission Instructions

In a file called Lab4.c, write your solution to the problem. There are ***ten (10) marks*** available in this lab, marked in two different ways:

1. ***TA Grading (4 marks):*** Your solution will be marked by a Teaching Assistant during your

scheduled lab period for programming style and your understanding of the code. You can discuss programming style with your TA and on Piazza. Here are some quick guidelines:

- Good choices for variable names that indicate their purpose.
- A consistent naming convention. Use camelCase for normal variable names.
- Comments that explain code that is difficult to understand.
- Proper indentation.
- Appropriate white space between lines for better readability.

The TA will also ask you some questions to be sure that you understand the underlying concepts being exercised in this lab.

***2. Automated Grading (6 marks):***

Your solution must be submitted electronically by the end of your scheduled lab period. Submission of the lab requires you to use a terminal. In the terminal, you must:

1. Go to the directory that contains `Lab4.c` (i.e., use the cd command in the terminal).
2. Type in the following command: `/share/copy/aps105s/lab4/submit`

This command will run an exercise program that will check to make sure everything looks okay. If it finds a problem, it will ask you if you are sure that you want to submit.

Note that you may submit your work as many times as you want prior to the deadline; only the most recent submission is marked. You can also run the exerciser on your own with the following command:

```
/share/copy/aps105s/lab4/exercise
```

Finally, you can also check to see if what you think you have submitted is actually there, for peace of mind, using the following command:

```
/share/copy/aps105s/lab4/viewsubmitted
```

**You must submit your lab by the end of your assigned lab period. Late submissions will not be accepted, and you will receive a grade of zero.**

## Obtaining Your Automated Grade

After all lab sections have finished, a short time later, you will be able to run the auto- marker to determine the auto-marked fraction of your grade on the code you have submit- ted. To do so run the following command:

```
/share/copy/aps105s/lab4/marker
```

This command will compile and run your code and test it with all the test cases used to determine the auto-mark grade. You will be able to see those test cases output and what went right or wrong.