Austin McCalley
Dr. Jennifer Parham-Mocello
Computer Science 160-020
26 October 2019

Assignment 5

Engineering Career Fair

The first company which I would like to visit at the career fair in the future is Hill AFB Civilian Engineering which employs civilian scientists and engineers to support the Air Force. I found this interesting because they are looking at employing cyber security, as well, as working with satellite systems, these two potential career paths extremely interest me. In the future I wish to work with communications through networking and cyber security.

Another company which I would like to visit at the career fair in the future is Aspen Capital which is an equity firm which utilizes programming to manage assets. I found this very interesting because investing in the stock market is something that I have been personally doing for a couple years now and then combining it with programming would be a lot of fun. The company atmosphere seems like a company that would suit me very well.

Understanding the Problem/Problem Analysis

The program is going to ask for several different inputs from the user. The first input that it is going to ask is what function it wants to do the integral approximation on, $f1(x) = 10x^2$, $f2(x) = 2x^2 - 5, f3(x) = x + 20$. We are also going to ask for what type of approximation we want to use; there are two different approximations that are going to be available to be selected, rectangle and trapezoidal. Then we are going to ask for the inputs of the starting and ending point to do the approximation for and then how many rectangle or trapezoids to approximate with. I am going to assume that the rectangle approximation is asking for right hand Riemann Sum approximation. The functional requirements for this problem are going to have each individual function have its own coded function to allow for any given x-input to allow for the answer to be returned. As well, we need an input checker to ensure that the function the user wants to be integrated can be programmed.

Program Design/Pseudo-Code

```
FUNCTION pickFunction():
    OUTPUT 'f1: f1(x) <- 10x^2'
    OUTPUT 'f2: f2(x) <- 2x^2 - 5'
    OUTPUT 'f3: f3(x) <- x + 20'
    function_names <- ['f1','f2','f3']
    user_in <- input 'Please pick a function from above to run an integration
on:’
    IF user_in not in function_names:
        OUTPUT 'Please try again. The function you picked was not in the pre-
approved list'
        RETURN pickFunction()
    ELSE:
        RETURN user_in
    ENDIF
ENDFUNCTION

FUNCTION pickApproximation():
    OUTPUT 'We are going to use trapezoidal(T) OR rectangle(R) method.'
    approximation_names <- ['t','r', 'T', 'R']
    user_in <- input ‘Please pick which approximation you would like to use:
'
    IF user_in not in approximation_names:
        OUTPUT 'Please try again. You did not pick a valid approximation
name.'
        RETURN pickApproximation()
    ELSE:
        RETURN user_in.lower()
    ENDIF
ENDFUNCTION

FUNCTION function1(x):
    ans <- 10.0*x**2.0
    RETURN ans
ENDFUNCTION

FUNCTION function2(x):
    RETURN 2.0*x**2.0 - 5.0
ENDFUNCTION

FUNCTION function3(x):
    RETURN x + 20.0
ENDFUNCTION

FUNCTION rectangleApproximation(func):
    try:
        number_rectangles <- input How many rectangles would you like to
approximate with? '
        a_point <- input 'Which x-value do you wish to start at? '
        b_point <- input 'Which x-value do you wish to end at? '
        diff <- b_point - a_point
         ENDIF
        interval <- diff / number_rectangles
                    ENDIF
        IF func = 'f1':
            # f1(x) <- 10x^2
            summation <- 0
            for x in range 1 to number_rectangles + 1:
                eval_point <- a_point+(interval*x)
                summation += function1(eval_point)
            ENDFOR
            RETURN interval * summation
        ELSEIF func = 'f2':
            # f2(x) <- 2x^2 - 5
            summation <- 0
      for x in range 1 to number_rectangles + 1:
         eval_point <- a_point+(interval*x)
                summation += function2(eval_point)
            ENDFOR
            RETURN interval * summation
        ELSEIF func = 'f3':
            # f3(x) <- x + 20
            summation <- 0
            for x in range 1 to number_rectangles + 1:
                eval_point <- a_point+(interval*x)
                summation += function3(eval_point)
            ENDFOR
            RETURN interval * summation
        ELSE:
            OUTPUT 'What did you do? We are going to try this again!'
            RETURN rectangleApproximation(func)
        ENDIF
    except Exception as e:
        OUTPUT 'We got an exception: %s! Lets try this again' % e
        RETURN rectangleApproximation(func)
ENDFUNCTION

FUNCTION trapezoidalApproximation(func):
    try:
        number_trapezoids <- input 'How many trapezoids would you like to
approximate with? '
        a_point <- input 'Which x-value do you wish to start at? '
        b_point <- input 'Which x-value do you wish to end with? '
        diff <- b_point - a_point
         ENDIF
        interval <- diff / number_trapezoids
                    ENDIF
        IF func = 'f1':
            # f1(x) <- 10x^2
            summation <- 0
            for x in range 1 to number_trapezoids + 1:
                eval_point <- a_point+(interval*x)
                IF x = 0 OR x = number_trapezoids:
                    summation += function1(eval_point)
                ELSE:
                    summation += 2.0*function1(eval_point)
                ENDIF
            ENDFOR
            RETURN (interval/2) * summation
        ELSEIF func = 'f2':
            # f2(x) <- 2x^2 - 5
            summation <- 0
            for x in range 1 to number_trapezoids + 1:
                eval_point <- a_point+(interval*x)
                IF x = 0 OR x = number_trapezoids:
                    summation += function2(eval_point)
                ELSE:
                    summation += 2.0*function2(eval_point)
                ENDIF
            ENDFOR
            RETURN (interval/2) * summation
        ELSEIF func = 'f3':
            # f3(x) <- x + 20
            summation <- 0
            for x in range 1 to number_trapezoids + 1:
                eval_point <- a_point+(interval*x)
                IF x = 0 OR x = number_trapezoids:
                    summation += function3(eval_points)
                ELSE:
                    summation += 2.0*function3(eval_points)
                ENDIF
            ENDFOR
         ENDIF

    except Exception as e:
        OUTPUT e
        OUTPUT 'We got an exception: %s! Let’s try this again!' % e
        RETURN trapezoidalApproximation(func)
ENDFUNCTION

function <- pickFunction()
approximation <- pickApproximation()
WHILE TRUE:
      IF approximation = 'r':
          approx <- rectangleApproximation(function)
          OUTPUT 'We got an approximation of %s for the function you
      requested!' % approx

      ELSEIF approximation = 't':
          approx <- trapezoidalApproximation(function)
          OUTPUT 'We got an approximation of %s for the function you
      requested!' % approx

      ELSEIF approximation = 'c':
            BREAK WHILE
      ENDIF
```

```
END WHILE
```

Testing Plan

The testing plans which I plan to implement is going to generate random numbers for the start and end points for each function and for each approximation. Then get the results and compare them to a computational calculator to ensure that the code is correct. An edge case which I must be aware of is when the start and end point are both zero which will result in zero as the product. With function three if you did not return zero then it will return the incorrect value because it will subtract from zero resulting in a negative number for no-area under the curve. Another edge case I have to be aware of is if the starting value is greater than the ending value which will not make the computation work correctly resulting in an incorrect answer. If I have time I am going to add an automated testing plan into this to ensure that my functions are working correctly.