BPHS 4090 (Fall 2013) - Computer Assignment: Estimate Integrals Numerically

1. INSTRUCTIONS

The primary purpose of this lab is to understand how go about numerically estimating a definite integral. A second goal is for you to gain some additional experience with basics of computer algorithms and writing scripts (for Matlab, or some other language is you wish) that will perform computations based upon a paradigm you encode.

2. BACKGROUND

The purpose of this lab is to develop an explicit method for numerically estimating definite integrals. Our approach will involve what is generally referred to as the *Riemann sum*. From a historical point of view, calculus was independently developed by both Isaac Newton (British) and Gottfried Leibniz (German) in the mid–1600s. They both realized that integration essentially allowed you to go 'backwards' (with respect to differential calculus). This notion of integration was further refined by Augustin Louis Cauchy (French) in the early 1800s, who developed the theory of limits and formalized the notion of an integral. In the mid-1800s, Bernhard Riemann (German) presented the idea of filling the area underneath a curve (i.e., the 'curve' being the integrand) with small rectangles to estimate the area and then take the limit such that the rectangle width becomes infinitesimally small. Thus there was a conceptual connection between the integral and a geometric feature (i.e., an area).

The idea here is that we will start with a specific definite integral (that we may or may not be able to solve by hand). The goal is to set up a computer program (via Matlab) so to be able to estimate the integral using the method as pioneered by Riemann (i.e., make N small little rectangles and add up all their areas in order to get an estimate of the integral). We can express this relationship analytically as

$$\int_a^b f(x)\,dx = \lim_{n\to\infty}\left(\sum_{i=0}^{n-1} f(x_i)\Delta x\right) = \lim_{n\to\infty}\left(\sum_{i=1}^{n} f(x_i)\Delta x\right) \tag{1}$$

for $a \le x \le b$. The second expression (sans the limit) is called the left–hand sum and the third the right–hand sum. Computationally, we can not determine $\lim_{n\to\infty}$, but we can approximate

$$\int_a^b f(x)\,dx \approx \sum^{N} f(x_i)\Delta x \tag{2}$$

for suitably large N. Note that we summing 'areas' here as the product of a height $[f(x_i)]$ and a width $[\Delta x]$.

3. COMPUTER CODING 101

This exercise will entail using computer logic techniques that may or may not be familiar to you such as *for* loops and array building. This section is designed to provide a brief overview to get you started[1].

To write a Matlab script, open up Matlab and under 'File', choose to create a new M-file. This will bring up a new window where you can type commands (this is called the Editor). You need to save this file using a name such as *testcode.m*. You can then execute it at the command line in the main Matlab window by typing *testcode*. In the Editor, you can type various commands which are sequentially executed when the code is executed[2].

To start you off, below is an example code that creates an array of numbers starting at one hundred and going down to thirty nine in integer steps. This skeleton of a code will form a foundation for how you might want to think about setting up your code to solve an integral as outlined below. Think carefully about the syntax and make sure to ask questions if there is something you don't understand!

```
% testcode1.m                          My Name (date)

% code to generate an array y containing the integers from 100 to 39

startVAL= 100;       % initial value
endVAL= 39;        % ending value

for nn=1:abs(startVAL-endVAL)+1

% the for loop above basically tells the loop to go around 62 times,
% from nn=1 to nn=62; the reason why we didn't write for nn=1:62
% is because we wanted flexibility in case we chose to differ the
% initial of ending value and still get all the desired values

   % create array y, using nn as an index
   y(nn) = startVAL -nn+1;

% also use nn here as a means to subtract an integer value from
% the initial value; the +1 is so that we don't lose the initial value itself

end


% now I have a 62x1 array (i.e. a matrix containing 62 elements) called
% y that has the desired values
```

Another snippet of code is included below. This creates an array of time values, then evaluates those based upon a specified function (ending up with another array

[1] In addition, see the Matlab section on the course webpage for addition information (http://math.arizona.edu/ cbergevin/250Afall2009/overview.html).

[2] You can add comments to your code by starting a given line with the % symbol. Commenting your code is a very good idea so to make it clear to others (as well as yourself!) as to what the various commands do.

of the *mapped* values). A *for* loop is then used to manipulate each value in the function array individually.

```
% create an array of time values from 0 s to 10 s in 0.1 s steps (i.e., this is our domain)

time= linspace(0,10,101);

% now specify a function and 'map' the domain so to determine the corresponding range

x= 4*exp(-time/pi).*sin(time);

% create a for loop that goes through each value of x that shifts it upwards and scales it down

for nn=1:size(x,2)

   % create a new scaled array, x2
   x2(nn) = 0.5* x(nn) +0.2;


end

% plot the two to compare

plot(time,x)
hold on
plot(time,x2,'ro')
```

One useful piece of advice. When multiplying two arrays together, you may want to just multiply array elements of the same index together (instead of doing matrix multiplication per se). In order to do this (given two arrays, `A` and `B`, both of which have the same nx1 dimensions), Matlab syntax requires you add in a period after the first array name, i.e., type `A.*B` (you can see this early in the above code snippet, where the exponential gets multiplied to the sin).

4. QUESTIONS

Part I (*25 points*)

Consider the integral

$$\int_{-1.5}^{\pi} e^{-cx}\, dx$$

where c is a constant greater than zero.

(1) Make a sketch of the integrand and indicate the integral as an *area* to compute. Make sure to clearly label things![3]

(2) Can you find an explicit solution for this integral? If so, give it. If not, explain.

[3]You can scan your written work (or take a picture, just make sure it is clearly legible).

(3) Assume $c = 2$. Solve the definite integral numerically (by hand) using the left-hand rule (LEFT), right-hand rule (RIGHT), the midpoint rule (MID), the trapezoid rule (TRAP) and Simpson's rule (SIMP). Do this for both $N = 2$ and $N = 3$ (where N is the number of divisions over the interval of integration). How does your estimate compare to the exact answer? How does it change with N?

(4) Which of the values you computed are underestimates? Overestimates? Explain.

(5) Write a Matlab script (i.e., a .m file) that does the computations for you. In your code, you should be able to readily change N and c (as well as your end points). There are many ways to do this. One suggested approach is to create an array of values of the integrand for $x \in [-1.5, \pi]$ (discretized into $N+1$ evenly spaced steps). Then you can use a *for* loop to go through each array element and compute the corresponding area. Summing those up throughout the loop will lead to the Riemann sum estimate.

(6) Does your code return the same estimates as computed by hand for $N = 2$ and $N = 3$? If not, why? [Hint: it should] How does the estimate change for different values of N (e.g. 5,10, 100,10000)? As N increases, how does it compare to your exact answer?

(7) What is the effect of varying c? Explain in the contexts of both your analytical answer and numerical simulations. Do both agree?

(8) What happens when you change the end points (e.g., $\int_{-1.5}^{\pi} \rightarrow \int_{0}^{\pi/2}$)? Does this make sense? Explain.

Part II (*15 points*)

Consider the integral

$$\int_{-1.5}^{\pi} e^{-cx^2}\, dx$$

where c is a constant greater than zero.

(1) Make a sketch of the integrand and indicate the integral as an *area* to compute. How does this differ from that in Part I?

(2) Can you find an explicit solution for this integral? If so, give it. If not, explain.

(3) Modify your code from Part I to numerically estimate this integral (again assuming $c = 2$). How does your estimate vary with N?

(4) What is the effect of varying c? Explain in the contexts of both your analytical answer and numerical simulations. Do both agree?

(5) Let $c = 1$. Suppose you wanted to know $\int_{-\infty}^{\infty} e^{-x^2}\, dx$. How might you go about computing this? Given sufficiently large N, what value do you find? Using google or wikipedia (or any other source of information at your disposal), search *normal distribution* as well as *error function*. Based upon those resources, can you determine explicitly what the value of estimate approaches? (Hint: it has π in it)

Extra Credit (*15 points*)

- There will always be some degree of error associated with your numerical estimate. For the integral given in Part I, describe clearly how that error changes with N. Is that different for LEFT, RIGHT, TRAP, MID and SIMP? Explain why you think there is a difference.
- What happens when the limits of integration are flipped (i.e., $\int_{\pi}^{-1.5} e^{-cx}\, dx$)? How does your estimate change? Explain.
- Modify your code so to compute the following integral:

$$\int_{0.17}^{1.32\pi} \cos\left(\frac{\sin\left(x^2\right)}{e^{-x^3}}\right) dx$$