Princeton University
Department of Operations Research
and Financial Engineering

ORF 201
Computer Methods in Problem Solving

Lab 4: Numerical Integration

Due Sunday, Feb 27, 11:59 pm

---

1 Introduction

This assignment involves the numerical evaluation of an integral using two different methods. The primary purpose of numerical integration is the evaluation of integrals which are either impossible or else very difficult to evaluate analytically. Analytical closed form expressions for integrals have many advantages over numerical evaluations which include better accuracy, more generality and the possibility of evaluating the effects of varying any parameters involved. Nevertheless, numerical integration is indispensable in many cases, since it can make the difference between getting a fairly accurate answer and having no answer at all. The two methods that we are going to use are the rectangle and trapezoidal rules.

2 Rectangle Rule

Consider a function $f(x)$ on the interval $\alpha \leq x \leq \beta$. We wish to evaluate the integral

$$I = \int_{\alpha}^{\beta} f(x)dx.$$

We divide the interval $\alpha \leq x \leq \beta$ into n equal subintervals each of width Δx , where

$$\Delta x = \frac{\beta - \alpha}{n}.$$

Δx will be referred to as the *step length* and each of the subintervals as a *panel*. We denote the i^{th} panel by $[x_{i-1}, x_i]$. By convention, we number our panels from left to right: hence, $x_0 = \alpha$ and

$x_n = \beta$. Let $x^{(i)}$ be the midpoint of the i^{th} panel and let $f^{(i)}$ be the value of the function $f(x)$ at the point $x^{(i)}$. We can approximate the function $f(x)$ on this panel by the constant value $f^{(i)}$. The area under this approximation to the function $f(x)$ on the i^{th} panel is just the area of a rectangle:

$$\int_{x_{i-1}}^{x_i} f(x)dx \approx (\Delta x) f^{(i)}$$

By extending the rectangle rule to the integral across the entire interval we get the following:

$$\begin{aligned} \int_{\alpha}^{\beta} f(x)dx &\approx (\Delta x)(f^{(1)} + f^{(2)} + ... + f^{(n)}) \\ &\approx (\Delta x)(\sum_{i=1}^{n} f^{(i)}). \end{aligned}$$

3 Trapezoidal Rule

Let $f_{i-1} = f(x_{i-1})$ and $f_i = f(x_i)$ be the values of the function $f(x)$ at the points x_{i-1} and x_i. Then we can approximate the function $f(x)$ on the i^{th} panel by using the straight line connecting f_{i-1} at x_{i-1} to f_i at x_i. The area under this approximation is simply the area of a trapezoid:

$$\int_{x_{i-1}}^{x_i} f(x)dx \approx \frac{f_{i-1} + f_i}{2} \Delta x.$$

The integral across two panels is given by

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx = \int_{x_{i-1}}^{x_i} f(x)dx + \int_{x_i}^{x_{i+1}} f(x)dx.$$

This integral can be approximated by

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx \approx \frac{\Delta x}{2}(f_{i-1} + 2f_i + f_{i+1}).$$

By extending the trapezoidal approximation rule to the integral across the entire interval we get the following:

$$\begin{aligned} \int_{\alpha}^{\beta} f(x)dx &\approx \frac{\Delta x}{2}(f_0 + 2f_1 + +2f_2 + ... + 2f_{n-1} + f_n) \\ &\approx \frac{\Delta x}{2}(f_0 + f_n + 2\sum_{i=1}^{n-1} f_i). \end{aligned}$$

Of course, the more subintervals we have, the better the approximation we get.

4 Programming Assignment

Your assignment is to write a program to evaluate the integral of the function

$$f(x) = ax^k e^{-\frac{x^2}{b}},$$

using the rectangle and trapezoidal rules. Also, plot the function $f(x)$ together with the approximations used in the rectangle and trapezoidal rules. These three plots should be drawn on a single coordinate system using different colors to distinguish the three plots. The axes for the coordinate system should also be shown (in yet another color).

To draw the "exact" depiction of $f(x)$, set n equal to a large number (say 1000) and draw a "connect-the-dots" approximation using this value of n.

5 Getting Started

First, you need to create some directories and copy some files. Begin like this:

```
cd public_html/JAVA/ORF201
mkdir integra
cd integra
```

Then copy the following files:

```
cp /u/orf201/public_html/JAVA/ORF201/integra/index.html .
cp /u/orf201/public_html/JAVA/ORF201/integra/integra.html .
cp /u/orf201/public_html/JAVA/ORF201/integra/Integra.java .
```

6 Compiling

First, compile the java code that you just copied over:

```
javac Integra.java
```

Then use appletviewer to see what the code currently does:

```
appletviewer integra.html
```

An applet window should pop up with some text fields, an *integrate* button, and an empty drawing canvas below. At the moment, if you push the integrate button not much happens. That is because the main part of the code has been stripped out. It is your job to fill it in according to the instructions given below.

7 Programming Notes

The Java code you are given as a starting point contains the "user interface" part but has the "core" part of the code deleted. It is your job to recreate the core part. The code that has been deleted comes from two methods, *integrate()* and *paint()*, in the IntegrationCanvas class. In addition to filling out these two methods, you must also write three new methods; one to evaluate the function $f(x)$, one to evaluate the integral using the rectangle rule, and the third to evaluate the integral using the trapezoidal rule.

The exponential function e^x is called Math.exp(). It takes a double as input and returns a double.

There is no closed form expression for most of the integrals you are being asked to evaluate. We suggest that you first develop and debug your code using some simpler function for which you can compute the correct answer and then when all the bugs are out of that version of the code change the integrand to the one requested for this assignment.

7.1 Graphics

Your plots are going to be framed by a rectangle, which will be defined by the function

```
GL.ortho2(lowx-marginx, highx+marginx, lowy-marginy, highy+marginy);
```

where the margins can be set equal to 10% of the length of the window in each direction. You will need to compute appropriate values for lowy, highy, marginx, and marginy.

In addition to the graphics methods you learned last week you may want/need to use a few more, which we describe here. To clear the canvas and set its background to black, use

```
GL.color(Color.black);
GL.clear();
```

To output text onto the canvas, such as the computed area, set a color and then use

```
GL.drawString(str, x, y, HOR_POS, VER_POS);
```

Here, str is a string, x and y are the coordinates at which you want to place the string and HOR_POS and VER_POS specify where on the string the point (x,y) should be. Legitimate values for HOR_POS are ”LEFT”, ”CENTER”, and ”RIGHT”. Legitimate values for VER_POS are ”TOP”, ”CENTER”, and ”BOTTOM”. To append a new string of output text to some text that you’ve already put on the canvas, you can use

```
GL.drawString(str);
```

GL.drawString remembers from one invocation to the next where text should go.

8 Minimum Requirements (for a B)

At a minimum:

1. Your integrator must correctly implement both the rectangle rule and the trapezoidal rule.
2. It must graph the function, the rectangular approximation, and the trapezoidal approximation.
3. It must display on the canvas the areas computed.
4. Implement three new methods; one for computing $f(x)$, a second for computing the integral using the rectangle rule, and a third for computing the integral using the trapezoidal rule.