CMSI 186
PROGRAMMING LABORATORY
Spring 2008

Program 5: Riemann Integration

With this assignment, we move from arithmetic to calculus — specifically, Riemann integration, a technique for estimating the area under a curve.

Program to Write

Write a class called *math.Riemann* that estimates the area under a curve for a user-given range, using the algorithm described in class. The curve can be one of a "function library" which you will provide; at a minimum, your program should support polynomials of arbitrary degree, plus at least *three* other functions (examples/ideas on the right).

math.Riemann shall accept three sets of arguments, identified by the order in which they are given:

- The first argument is a name that uniquely identifies the function to use.
- Zero or more arguments after the function name supply any additional information needed by the function (and they thus depend on the function).
- The last two arguments of the command indicate the range over which the area over the function's curve is to be estimated. The range may be given with either the upper or lower bound first.

As always, *math.Riemann* should display a helpful error message if the given arguments are somehow invalid (e.g., unsupported function names, incorrect/missing additional information, non-numeric arguments, etc.).

The output of *math.Riemann* shall be a table showing one line for each iteration/refinement of your estimate, starting with a single rectangle (i.e., *deltaX* = *upperBound* – *lowerBound*) and continuing until the difference between the previous and succeeding estimates falls within ±0.01% of each other, after which the program should stop. The table should include columns for the iteration number, *deltaX*, the estimated area, and the % difference from the previous estimate.

When invoked with *--help* as the first argument (i.e., *java math.Riemann --help*), the program should display a message describing how it should be used, including a complete list of the functions that it supports, stating the names to use and the additional information that they require.

Examples

The following examples illustrate a number of possibilities which you may implement.

- You must start with a *poly* function, for polynomials of arbitrary degree. The example below estimates the area under $f(x) = 2x^4 - x^2 + 9x + 5.2$ from $x = 1.0$ to 3.0:

  java math.Riemann poly 2.0 0 –1.0 9.0 5.2 1.0 3.0
- To estimate the area under $f(x) = 100$ (yes, a horizontal line) from $x = -20.5$ to 30.75:

  java math.Riemann poly 100 30.75 –20.5
- To estimate the area under $f(x) = 9x^3 + 8x^2 + 7x + 6$ from $x = 0.5$ to 1.0:

  java math.Riemann poly 9 8 7 6 0.5 1.0
- To estimate the area under $f(x) = \sin x$ from $x = 0$ to 3.141592:

  java math.Riemann sin 0 3.141592
- Functions may support coefficients and accompanying terms; for example, estimating the area under $f(x) = (2 \ln x) + 0.5$ from $x = 1$ to 10 could look like this:

  java math.Riemann ln 2.0 0.5 1 10
- Feel free to create "combo" functions; for example, estimating the area under $f(x) = \cos x - \sin x$ from $x = -3$ to 3 could look like this:

  java math.Riemann cosMinusSin 3 –3

A Design Note, Extra Work, and a Gotcha

- No unit tests are explicitly required this time around; if you do implement some tests, use the *test* identifier to make the program run them.
- If you finish early, try to implement an optional *--threshold=n* argument that changes the program's default ±0.01% termination threshold.
- Speaking of termination, there may be other conditions under which stopping would make sense. If you suspect you've found such a condition, let me know and we'll discuss what to do.