Problem Set 5

(Due Mar. 6, 1:00 PM)

Instructions

1. Work on git. Fork the repository found at `https://github.com/minheeseo/PS5` and add your code, committing and pushing frequently. Use meaningful commit messages – these may affect your grade.

2. You will need to submit a complete github repository containing the package and a development file. The development file should walk through the entire process of building the package and include some example code showing how each function works.

3. You will be graded on:

   - Comments
   - Correct/frequent use of GitHub with lots of commits.
   - Elegance of code (e.g., apply rather than loops, speed of functionality, etc.)
   - Readability of the code/stability of naming conventions
   - Documentation/full completion of package structure

   how to record if the request goes directly to the agency

4. If you have any questions regarding the Problem Set, contact the TA or use her office hours.

Background

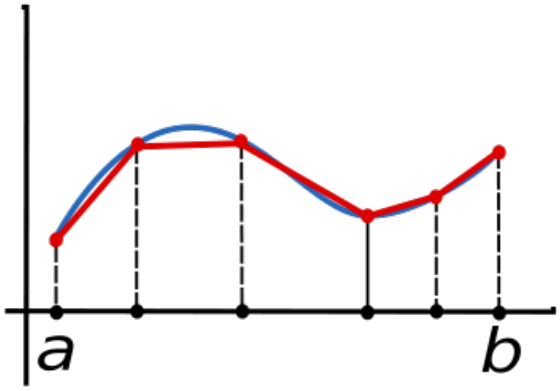
Trapezoidal

Trapezoid rule is a technique for approximating the definite integral, and it follows the mathematical expression below:

$$\int_a^b f(x)dx \approx T$$

$$T = \frac{h}{2}(f(x_0) + 2f(x_1) + ... + 2(f_{n-1}) + f(x_n))$$

$$\text{where } h = \frac{b-a}{n}$$



Simpsons Rule

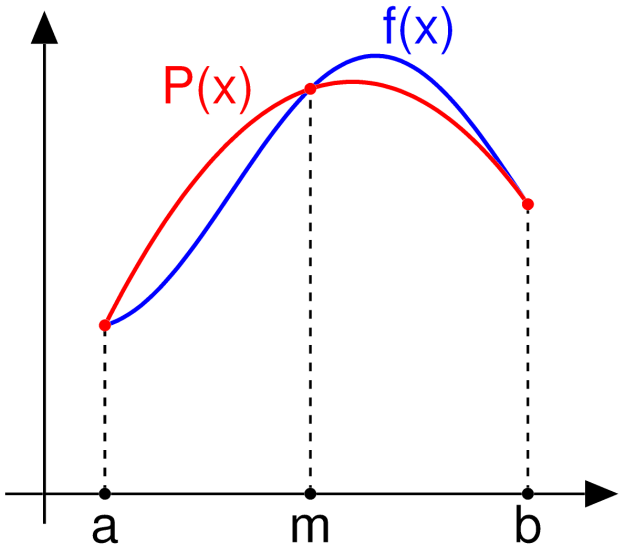
Simpson's rule is another technique for approximating a definite integrals, numerically. It follows the approximating below:

$$\int_a^b f(x)dx \approx S$$

$$S = \frac{h}{3}(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + ... + 4(f_{n-1}) + f(x_n))$$

$$\text{where } h = \frac{b-a}{h}$$

Image that you are drawing a parabola from the points (a, f(a)) to (b, f(b)) that also goes through the (m, f(m)). We can approximate the area under the parabola as being equal to $\int_a^b f(x)dx$.



For example, the parabola between any two points, (u, f(u)) and (w, f(w)) is drawn according to the formula (where $v = \frac{w-u}{2}$):

$$p(x) = f(u)\frac{(x-v)(x-w)}{(u-v)(u-w)} + f(v)\frac{(x-u)(x-w)}{(v-u)(v-w)} + f(w)\frac{(x-u)(x-v)}{(w-u)(w-v)}$$

Then, the integral under that parabola is:

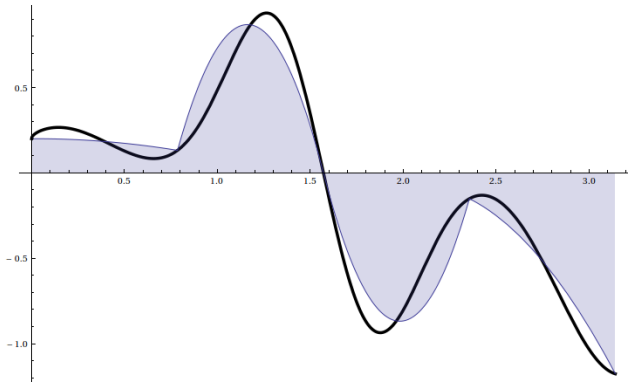
$$\int_u^w p(x)dx = \frac{h}{3}(f(u) + 4(f(v)) + f(w))$$

If we imagine carrying on that calculation many times between different points along the curve, we get

$$\int_a^b f(x)dx \approx S$$

$$S = \frac{h}{3}(f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3)... + 4f(x_{n-1}) + f(x_n))$$

where $h = \dfrac{b-a}{n}$



Create S4 Package

1. To prepare you for your midterm, your goal here is to use `devtools` to create an S4 R package named `integrateIt`. Background of Trapezoidal rule and Simpsons rule can be found above.

2. The package should include appropriate functions and appropriate documentation

3. The package should have two classes, `Trapezoid` and `Simpson`

4. The package should have one generic, `integrateIt`

5. The package should contain three methods, `integrateIt` and `print`

   (a) `integrateIt` method

      i. Takes four arguments: a vector of values (x), a vector of evaluated values ($f(x) = y$), starting/ending values (a,b), and a *Rule* argument that can be either "Trap" or "Simpson".

      ii. Have three outputs: an object of class `Trapezoid` or class `Simpson`, the values of x and y, and the result

      iii. Both classes should have validation methods that include a few appropriate tests

      iv. you will need to create an `initialize` function for each class, which will be used internally by `integrateIt`

   (b) `print` method

      i. A very simple print method for each class, which prints out just the integrated value (rather than all of the results)

      ii. EXTRA CREDIT if you accomplish this using a subclass scheme rather than writing separate functions for each class

6. You need to create an appropriate error message, and the error message should be intuitive. Test the error message on your development file.

7. GRADS ONLY: create an extra generic/method called `tolTest`.

   (a) `tolTest` method takes in five arguments

      i. A `fun` (function)

      ii. A `tolerance` argument

      iii. A `rule` argument that indicates whether the Trapezoidal or Simpson's

      iv. A `start` argument for the number of intervals it should start with

      v. A `correct` argument that provides the correct answer for the integral

   (b) `tolTest` operations

i. It should take in a function and increase the number of intervals n until the answer it provides using the specified approximation is within `tolerance` of the correct answer. (HINT: `integrate()`)

(c) `tolTest` output

i. All of the inputs

ii. The final n

iii. The absolute error of the estimate

8. Again, your development file should document the the entire process. This file should contain codes to check whether your functions work.