Simple integration assignment

For this assignment, you will need to do some numerical integrations.

1. Make a program that can do integration of a function by both Simpson's rule and the rectangular rule method.
2. Integrate out the Gaussian probability function using this code, from a particular value to "close enough" to infinity.
3. Compare with the value of the error function, erfc that is standard in C (with math.h included).

You should get your answers to match within 1%. You should calculate the probability of getting a 3σ event. That is, the probability that an event more than 3 standard deviations from the mean of a Gaussian occurs by chance.

Some additional mathematical formulae you will need

The Gaussian probability density function is defined as:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-} \frac{(x-\mu)^2}{2\sigma^2}, \tag{1}$$

where μ is the mean of the distribution, and σ is the standard deviation of the distribution, and the normalization is set so that $\int_{-\infty}^{+\infty} f(x)dx = 1$.

Also, a hint to think about for how to figure out how far is far enough in integrating out (since you cannot integrate all the way to infinity), in order to have less than a 1% error: The value of $e^{\frac{-x^2}{2}}$ will always be less than the value of $e^{-\frac{x}{2}}$ for large positive values of x. The latter is something you should know how to integrate quite easily on paper.

Some challenge problem suggestions

Here are a few challenge problem suggestions.

1. Compare the amount of computer time it takes to get values to converge to better than 1% of the correct answer for the different ways you have of doing things. You will need to use the `clock` command for this. The `clock` command returns something typically in microseconds, but often with a resolution of many milliseconds, so you may actually need to run a loop that repeats your procedures over and over to get a really good answer of what the CPU time used is.
2. Investigate the numerical errors versus number of steps for both methods. See if you can find a point where the answer gets worse as you add more steps.
3. Write your code in such a way that you can pass an arbitary function by function name to it, and you can pass the parameters of that function as an array. This may be quite hard, but it is also something that you will then find very easy to adapt for nearly any numerical integral you wish to do in your career as a scientist. This will be an exception to the rule that you need to come up with your own idea and execute it very well to get an A+.