Estimating π via Numerical Integration

CS 330

1 Introduction

For this program, your will implement four numerical methods for approximating definite integrals and use these to estimate the value of π. By divvying up the interval of integration into smaller pieces, you should obtain more accurate estimates. The output of your program will tabulate the errors as described below.

2 π via Integration

One method for estimating π is to numerically compute the following definite integral:

$$\pi = 4 \cdot \arctan 1 = \int_0^1 \frac{4}{1+x^2}\, dx \tag{1}$$

Note that evaluating the integrand involves only simple operations (*i.e.*, addition, multiplication, and division). Figure 1 shows that the function is smooth and well-behaved on the interval $0 \leq x \leq 1$. This seems to indicate that our techniques for numerical integration should work well in this case.

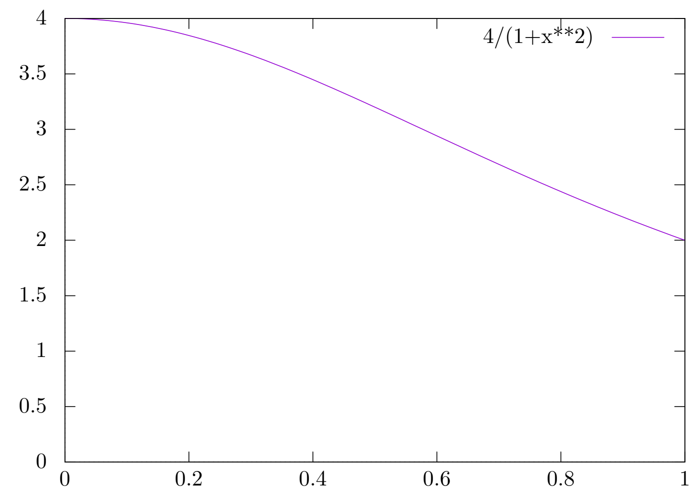

Figure 1: $f(x) = 4/(1+x^2)$ on the interval $0 \leq x \leq 1$

2.1 Four Numerical Methods

So far we have examined the *Trapezoid Rule* and *Simpson's* $\left[\frac{1}{3}\right]$ *Rule*. There are other Newton-Cotes closed formulas, including *Simpson's* $\frac{3}{8}$ *Rule* and *Boole's Rule*. In the same way that Simpson's $\frac{1}{3}$ Rule requires an even number of subintervals, Simpson's $\frac{3}{8}$ and Boole's Rules require multiples of 3 and 4 subintervals, respectively. Here are the simple forms of the all four rules. Note that $x_0 = a$ and $x_n = b$ for all of these, and $h = (b - a)/n$.

- Trapezoid Rule

$$\int_{x_0}^{x_1} f(x)\ dx \approx \frac{1}{2}h\left[f(x_0) + f(x_1)\right] \tag{2}$$

- Simpson's $\frac{1}{3}$ Rule

$$\int_{x_0}^{x_2} f(x)dx \approx \frac{1}{3}h\left[f(x_0) + 4f(x_1) + f(x_2)\right] \tag{3}$$

- Simpson's $\frac{3}{8}$ Rule

$$\int_{x_0}^{x_3} f(x)dx \approx \frac{3}{8}h\left[f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)\right] \tag{4}$$

- Boole's Rule

$$\int_{x_0}^{x_4} f(x)dx \approx \frac{2}{45}h\left[7f(x_0) + 32f(x_1) + 12f(x_2) + 32f(x_3) + 7f(x_4)\right] \tag{5}$$

You will be using the composite forms of all of these for this assignment. For example, Figure 2 illustrates how to construct the composite version of Boole's Rule for the case where $n = 16$.

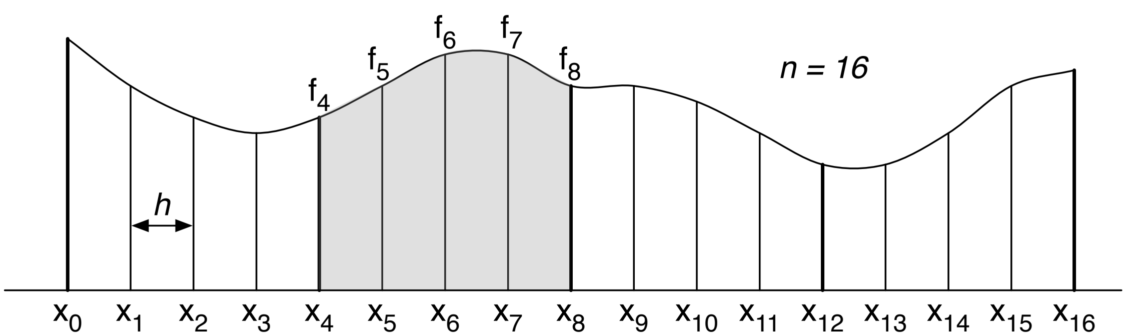

Figure 2: Division of interval $[x_0, x_{16}] = [a, b]$ for the composite version of Boole's Method using $n = 16$. In this case, the basic method is applied four times; the shaded region represents one of the four regions.

Applying Boole's Rule to approximate the shaded area in the figure we have

$$\int_{x_4}^{x_8} f(x)\ dx \approx \frac{2h}{45}\left[7f(x_4) + 32f(x_5) + 12f(x_6) + 32f(x_7) + 7f(x_8)\right] \tag{6}$$

To approximate the area over the entire interval we will apply this rule $n/4$ times. Avoiding redundant function evaluations at the points x_4, x_8, and x_{12}, the composite rule over the entire interval $[x_0, x_n] = [a, b]$ becomes

$$\begin{aligned}\int_{x_0=a}^{x_n=b} f(x)\,dx \approx \frac{2h}{45}[7(f_a + f_b) &+ \\ 14(f_4 + f_8 + f_{12} + \cdots + f_{n-8} + f_{n-4}) &+ \\ 32(f_1 + f_3 + f_5 + \cdots + f_{n-3} + f_{n-1}) &+ \\ 12(f_2 + f_6 + f_{10} + \cdots + f_{n-6} + f_{n-2})]&\end{aligned} \tag{7}$$

where $h = (b-a)/n$ and n is a multiple of 4. The composite form of Simpson's $\frac{3}{8}$ Rule can be similarly derived. We have covered composite forms for Simpson's $\frac{1}{3}$ Rule and the Trapezoid Rule in class.

3 Experiment

For this project you will implement the composite versions of the four methods mentioned above and use them to estimate π by approximating the integral in Equation 1. If n is a multiple of 12, then we satisfy the restrictions on n for all four methods. Your program must output (to `stdout`) the tabulated errors shown in Table 1 for values of $n = 12 \cdot 2^i$ $(i = 0, 1, 2, 3, \ldots, 16)$.

n	*Trap Error*	*Simp 1/3 Error*	*Simp 3/8 Error*	*Boole Error*
12				
24				
48				
96				
192				
384				
$\vdots$				
786432				

Table 1: Tabulated errors for estimates of π using our four methods.

The errors should be printed using exponential notation with at least 10 digits of precision (i.e. use format string `"%0.10Le"` with `printf`). There should be fives columns with values separated by whitespace. Here are the first four rows output by my solution:

```
12 1.1574067429e-03 1.3284413311e-08 5.9710615545e-08 4.4006875913e-08
24 2.8935184147e-04 2.0764483535e-10 9.3431488457e-10 6.6413984945e-10
48 7.2337962801e-05 3.2444045280e-12 1.4600162876e-11 1.0382410426e-11
96 1.8084490738e-05 5.0573260857e-14 2.2800858424e-13 1.6246856291e-13
```

3.1 Implementation Details

Create a function that performs the composite version of each rule and use the `long double` data type. For example, the function prototype for general purpose "trapezoid integrator" would look like this:

```
long double trapezoid(long double (*f)(long double),//function
                      long double a, long double b, //interval
                      int n);
```

The specific function we are using can be defined as follows:

```
long double f(long double x) {
  return 4.0L/(1.0L + x*x);
}
```

Use a high precision constant for the ground truth estimate of π. Usually the one provided by `math.h` is sufficient, but this is not always included depending on the system – I use the following snippet in my source:

```
#ifndef M_PI
#define M_PI 3.14159265358979323846264338327950288   /* pi */
#endif
```

4 Submission

As usual, create a zip file or tarball containing the source code for your solution. Your main source file must begin with a header identifying you and the project, and telling us how to build and run your program. Ideally, this should involve a Makefile, but this is not required.