Estimating π via Numerical Integration

CS 360

Due 11:59 pm Friday, October 18, 2013

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.

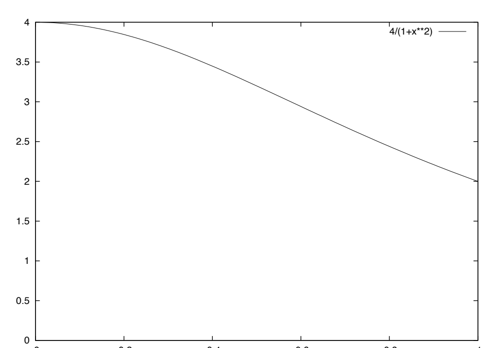

Figure 1: $f(x) = 4/(1+x^2)$ on the interval $0 \leq x \leq 1$.

2.1 Four Numerical Methods

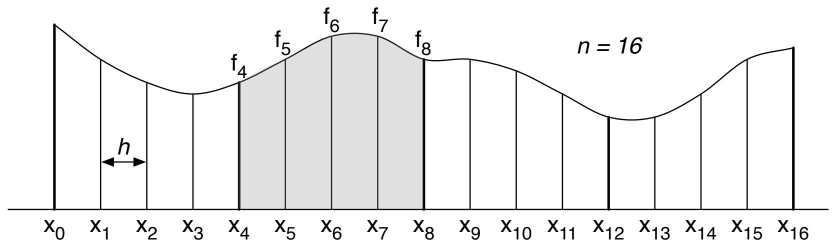

Figure 2: Division of interval $[x_0, x_{16}] = [a, b]$ for the composite version of Boole's Method using $n = 16$. In this case, the basic method is applied four times; the shaded region represents one of the four regions.

So far we have examined the *Trapezoid Rule* and *Simpson's* $\frac{1}{3}$ *Rule* listed on page 234 [1] of your text book. There are two other rules listed here as well: *Simpson's* $\frac{3}{8}$ *Rule* and *Boole's Rule*. The Composite version of Simpson's $\frac{1}{3}$ Rule requires us to divvy up the interval into an even number of pieces. Simpson's $\frac{3}{8}$ Rule and Boole's Rule require that the number of subintervals n is a multiple of 3 and 4 respectively. For example, Figure 2 illustrates how to construct the composite version of Boole's Rule for the case where $n = 16$. Using the notation $f_i = f(x_i)$, and applying Boole's Rule to approximate the shaded area in the figure we have

$$\int_{x_4}^{x_8} f(x)\ dx \approx \frac{2h}{45}\left[7f_4 + 32f_5 + 12f_6 + 32f_7 + 7f_8\right]. \tag{2}$$

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

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

where $h = (b - a)/n$ and n is a multiple of 4.

3 Error Analysis for the Trapezoid Method

The error E for the Composite Trapezoid Rule is bounded as follows:

$$|E| \le \left|\frac{(b-a)\cdot h^2}{12} f''(\xi)\right|. \tag{4}$$

[1] Page 225 of 6th edition

The first two derivates of our function $f(x)$ are

$$f'(x) = \frac{-8x}{(1+x^2)^2} \tag{5}$$

$$f''(x) = \frac{32x^2}{(1+x^2)^3} - \frac{8}{(1+x^2)^2}. \tag{6}$$

Since $|f''(x)| < 8$ on the interval $[0,1]$ the error bound is

$$|E| \leq \frac{h^2}{12}8 = \frac{2}{3}h^2. \tag{7}$$

Therefore if we want to guarantee that the error E is no larger than some prescribed tolerance ϵ, we need $2h^2/3 \leq \epsilon$ or

$$h \leq \sqrt{3\epsilon/2}. \tag{8}$$

Therefore, we need

$$n \geq (1-0)/h = \sqrt{\frac{2}{3\epsilon}}. \tag{9}$$

For example, if $\epsilon = 10^{-6}$ we need $n \geq 817$. In other words, if we use 817 or more trapezoids we will obtain π to six decimal places (it will probably take less than this).

4 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 $n = 12 \cdot 2^i$ ($i = 0, 1, 2, 3, \ldots$). 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
```

4.1 Implementation Details

Create a subroutine 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:

n	*Trap Error*	*Simp 1/3 Error*	*Simp 3/8 Error*	*Boole Error*
12				
24				
48				
96				
192				
384				
768				
$\vdots$				

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

```
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
```

5 Submission

Archive your source code and `README` text file and submit electronically by the due date. Your `README` file should contain your contact information (name and email), a brief description of the project (an overview for the uninitiated reader), instructions on how to build and run your program (explaining any input and output), and a list of files in the archive.