Numerical Integration Using Trapezoid Rule and Simpson's Rule

Kevin Juan (kj89)

September 12th 2018

The Numerical Methods and Algorithms

The objective of this assignment was to implement two different algorithms for numerically evaluating integrals. The first method that was tested was the Trapezoid Rule. This method requires the selection of N points over the domain of integration, $[a, b]$, for $f(x)$. These N points are used to subdivide $f(x)$ into $N-1$ equally spaced intervals of length $h = \frac{b-a}{N-1}$ with area $A = \frac{h}{2}[f(x_i) + f(x_{i+1})]$ where $x_i = a + ih$ and $i = 0, 1, 2...N-1$, which is the area of a trapezoid. The integral $\int_a^b f(x)dx$ can then be estimated as the sum of the area of all the trapezoids. The Trapezoid Rule has an error of $\mathcal{O}(h^3 f''')$. Thus, the Trapezoid Rule can be written as:

$$I_{Total} = \sum_{i=1}^{N} I_i = \frac{h}{2}[f(x_0) + f(x_1) + f(x_1) + f(x_2) + f(x_2) + ...] \tag{1}$$

Equation 1 can be further simplified as such:

$$I_{Total} = h\left[\frac{1}{2}f(x_0) + f(x_1) + f(x_2) + ... + f(x_{N-2}) + \frac{1}{2}f(x_{N-1})\right] + \mathcal{O}(h^2 f''') \tag{2}$$

The Trapezoid Rule derivation and properties can be found in Press et al[1] §4.1.1 and Landau et al[2] §6.2.1.

The other method that was implemented was Simpson's Rule, which approximates $f(x)$ as a quadratic function over the range $x_{i-1} \leq x_i \leq x_{i+1}$. Contrary to the Trapezoid Rule, Simpson's Rule utilizes 3 points equally spaced points of size h in the range over interest in order to capture the curvature of $f(x)$. This allows us to set $f(x) \sim \alpha x^2 + \beta x + \gamma$ and use the following substitutions:

$$u = \frac{x - x_i}{h},\ du = \frac{dx}{h},\ x_{i-1} \to u = -1,\ x_i \to u = 0,\ x_{i+1} \to u = +1 \tag{3}$$

Substituting into the quadratic equation gives the following equations:

$$f(u) = \alpha u^2 + \beta u + \gamma \tag{4}$$
$$f(u = -1) = f_{-1} = \alpha + \beta + \gamma \tag{5}$$
$$f(u = 0) = f_0 = \gamma \tag{6}$$
$$f(u = +1) = f_{+1} = \alpha + \beta + \gamma \tag{7}$$

The three equations can then give values of α, β, and γ in terms of f:

$$\alpha = \frac{f_{-1} + f_{+1} - 2f_0}{2} \tag{8}$$
$$\beta = \frac{f_{+1} - f_{-1}}{2} \tag{9}$$
$$\gamma = f_0 \tag{10}$$

For the region $x_{i-1} \leq x_i \leq x_{i+1}$, the integral I_i can be evaluated as follows:

$$I_i = \int_{-1}^{+1} (\alpha x^2 + \beta x + \gamma) h du = \frac{h}{3}[f_{-1} + 4f_0 + f_{+1}] + \mathcal{O}(h^5 f^{(4)}) \tag{11}$$

This shows that Simpson's rule is 2 orders in h better than the Trapezoid Rule for a 1 order higher approximation. Similarly, a composite form can be written as:

$$I_{Total} = \sum_{i=1}^{N} I_i = \frac{h}{3}[f_0 + 4f_1 + 2f_2 + 4f_3 + ... + 4f_{N-2} + f_{N-1}] + \mathcal{O}(h^4) \tag{12}$$

More information detailing Simpson's Rule can be found in Press et al[1] §4.1.1 and Landau et al §6.2.2.

Implementation and Results

The following equation was used to test the Trapezoid Rule and Simpson's Rule for $x = 0.5$ and $x = 0.9999$:

$$K(x) = \int_0^{\frac{\pi}{2}} \frac{1}{\sqrt{1 - x \sin^2 \theta}} d\theta \tag{13}$$

The following recursive algorithm was used to evaluate the Trapezoid Rule to a given level of tolerance:

```
$n \leftarrow 1$ (n is the number of intervals)
$S \leftarrow 0.5[f(x_{min}) + f(x_{max})]$
$I_{New} \leftarrow (x_{max} - x_{min})S$
Repeat
    $n \leftarrow 2n$
    $\Delta x \leftarrow (x_{max} - x_{min})/n$
    $I_{Old} \leftarrow I_{New}$
    $S \leftarrow S + \sum_{i=1,3,5,...n-1} f(x_{min} + i\Delta x)$
    $I_{New} \leftarrow S\Delta x$
Until $|I_{New} - I_{Old}| < |tol|$ and $n > n_{min} \sim 8$
$I_{New}$ is the new integral
```

A similar recursive algorithm was used to evaluate Simpson's Rule to a specified level of tolerance:

```
$n \leftarrow 2$ (n is the number of intervals)
$S_1 \leftarrow f(x_{min}) + f(x_{max})$
$S_2 \leftarrow 0$
$S_4 \leftarrow f(0.5(x_{min} + x_{max}))$
$I_{New} \leftarrow 0.5(x_{max} - x_{min})(S_1 + 4S_4)/3$
Repeat
    $n \leftarrow 2n$
    $\Delta x \leftarrow (x_{max} - x_{min})/n$
    $S_2 \leftarrow S_2 + S_4$
    $I_{Old} \leftarrow I_{New}$
    $S_4 \leftarrow \sum_{i=1,3,5,...n-1} f(x_{min} + i\Delta x)$
    $I_{New} \leftarrow \Delta x(S_1 + 2S_2 + 4S_4)/3$
Until $|I_{New} - I_{Old}| < |tol|$ and $n > n_{min} \sim 8$
$I_{New}$ is the new integral
```

The data shown in Table 1 displays the results for the integration of Equation 13 using the Trapezoid Rule. At $x = 0.5$, the Trapezoid Rule converged to a value using a tolerance of 1×10^{-9} after $N = 8$. For $x = 0.9999$, the convergence required more intervals, and was not achieved for the same level of tolerance until $N = 512$.

N	$K(x = 0.5)$	$K(x = 0.9999)$
2	1.85495913	40.7732725
4	1.85407523	21.8375602
8	1.85407468	12.7136836
16	1.85407468	8.49387261
32	1.85407468	6.71466565
64	1.85407468	6.11464253
128	1.85407468	5.99808944
256	1.85407468	5.99161698
512	1.85407468	5.99158934
1024	1.85407468	5.99158934
2048	1.85407468	5.99158934
4096	1.85407468	5.99158934

Table 1: Tabulated data using the Trapezoid Rule for $K(x)$ to 9 digits of precision.

Table 2 displays the values for numerical integration of Equation 13 using Simpson's method. When $x = 0.5$, Simpson's Rule converged after $N = 16$ intervals were utilized for a tolerance of 1×10^{-9}. With the same tolerance level, but $x = 0.9999$, Simpson's Rule converged to a value after $N = 1024$ intervals. It is noted that to 9 digits of precision, both methods converged to the same value for $x = 0.5$ and $x = 0.9999$.

N	$K(x = 0.5)$	$K(x = 0.9999)$
2	1.84123921	27.9226251
4	1.85378059	15.5256562
8	1.85407449	9.67239144
16	1.85407468	7.08726894
32	1.85407468	6.12159667
64	1.85407468	5.91463482
128	1.85407468	5.95923841
256	1.85407468	5.98945950
512	1.85407468	5.99158013
1024	1.85407468	5.99158934
2048	1.85407468	5.99158934
4096	1.85407468	5.99158934

Table 2: Tabulated data using the Simpson's Rule for $K(x)$ to 9 digits of precision.

The script shown in the next section was used to generate 100 data points for $K(x)$ on the domain $0 \leq x \leq 0.9999$ using both the Trapezoid Rule and Simpson's Rule. The `for` loops within `int main()` calls the `trapezoid` and `simpson` functions for the 100 equally spaced values of x. The `while` loops nested within the `for` loops are intended to check the integration value against the last value calculated to determine at which N the integration values for the two methods fall below the set tolerance of 1×10^{-9}. The following plots were generated using both methods. As expected, the plots are identical for both methods.

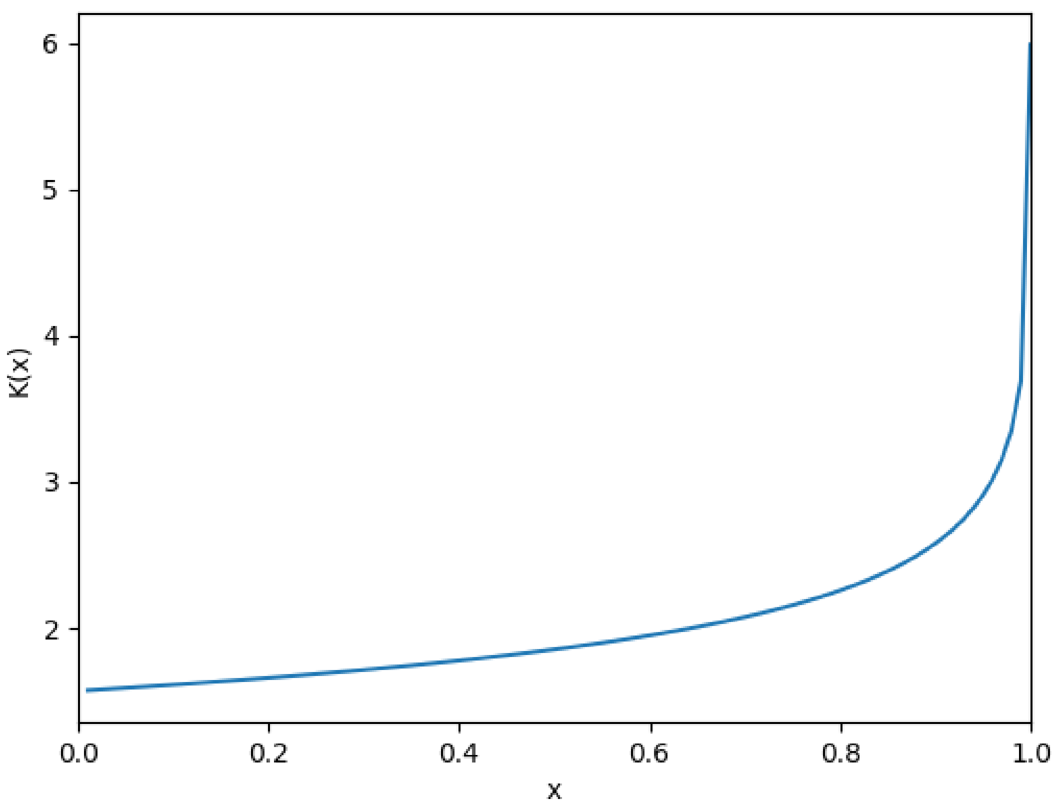

Figure 1: Plot of $K(x)$ for the domain $0 \leq x \leq 0.9999$ using the Trapezoid Method.

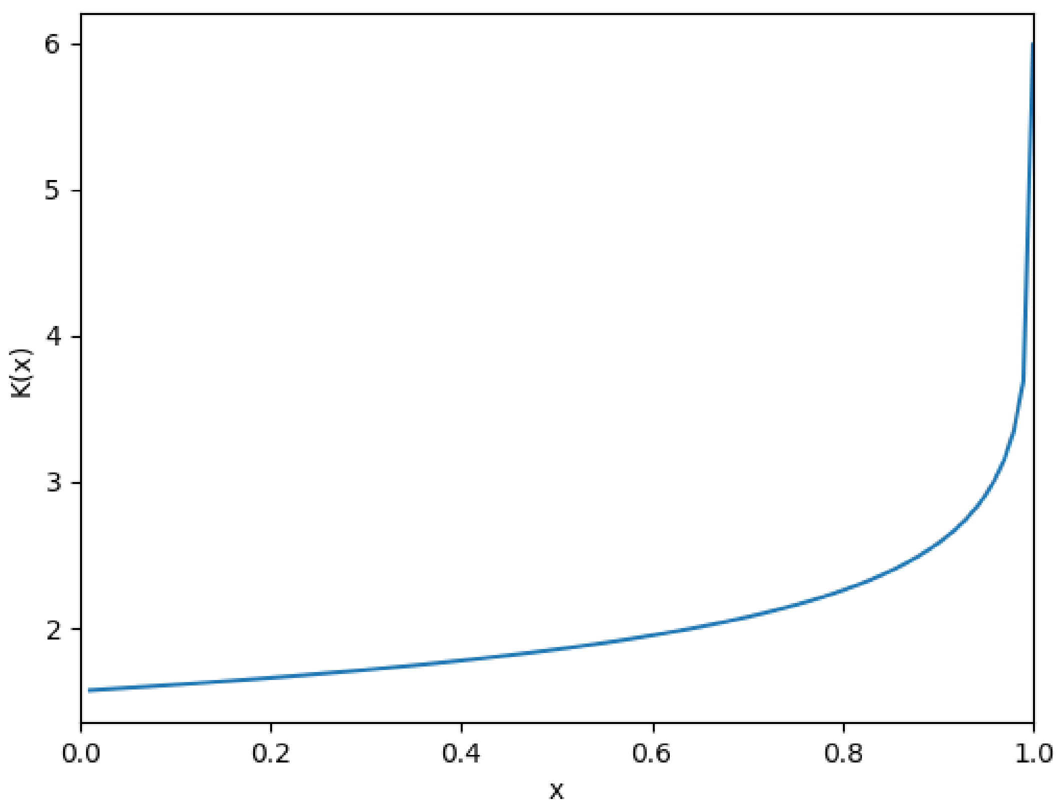

Figure 2: Plot of $K(x)$ for the domain $0 \leq x \leq 0.9999$ using Simpson's Method.

Source Code

```
/* AEP 4380 Homework #1

Test numerical integration
Trapezoid and Simpson's methods are tested

Run on a core i7 using clang 902.0.39.2

Kevin Juan 12 September 2018
*/

#include <cstdlib> // plain C
#include <cmath>
#include <assert.h>
```

```
#include <iostream> // stream IO
#include <fstream>  // stream file IO
#include <iomanip>  // to format the output

using namespace std;

/* Function K(x) to be integrated */
double Kx(double x, double theta) {
  return (1 / sqrt(1 - x * sin(theta) * sin(theta)));
}

/* Returns the value of the integral using Trapezoid Rule from a to b
using N intervals for the function *func that takes in 2 paramaters one
of which is x. Precondition: N must be positive
*/
double trapezoid(double (*func)(double, double), double a, double b, int N,
double x) {
  double h, S = 0.5 * (func(x, a) + func(x, b));
  double IOld, INew = (b - a) * S;
  int i, NInt = 1; // NInt is the minimum number of allowable intervals

  assert(N > 0); // check that the number of intervals is positive

  while (NInt <= N) {
    NInt = 2 * NInt;
    h = (b - a) / NInt;
    IOld = INew;
    for (i = 1; i <= NInt - 1; i += 2) {
      S = S + func(x, a + i * h);
    }
    INew = S * h;
  }
  return INew;
}

/* Returns the value of the integral using Simpson's Rule from a to b using
N intervals for the function *func that takes in 2 paramaters. Precondition:
N must be greater than 1.
*/
double simpson(double (*func)(double, double), double a, double b, int N,
double x) {
  double h, S1 = func(x, a) + func(x, b), S2 = 0.0;
  double S4 = func(x, 0.5 * (a + b)), IOld;
  double INew = 0.5 * (b - a) * (S1 + 4 * S4) / 3;
  int i, NInt = 2; // NInt is the minimum number of allowable intervals
  assert(N > 1);  // check that number of intervals is greater than 1

  while (NInt <= N) {
    NInt = 2 * NInt;
    h = (b - a) / NInt;
    S2 = S2 + S4;
    IOld = INew;
    S4 = 0; // set S4 to 0 to reset its value
    for (i = 1; i <= NInt - 1; i += 2) {
      // sum f(thetaA + i * dTheta) over 1, 3, 5, ... N - 1
      S4 = func(x, a + i * h) + S4;
    }
    INew = h * (S1 + 2 * S2 + 4 * S4) / 3;
  }
  return INew;
}

int main() {
  double thetaB = 4.0 * atan(1.0) / 2, thetaA = 0.0, tol = pow(10, -9);
  double xmin = 0.0, xmax = 0.9999, x;
  int i, j, NMax = 4096, pts = 100;

  ofstream fp;             // output file using streams
  fp.open("hw2.dat"); // open new file for output
  fp.precision(9);         // select 9 digits

  if (fp.fail()) {
    // or fp.bad()
    cout << "cannot open file" << endl;
    return (EXIT_SUCCESS);
  }
  fp << setw(45) << "Trapezoid Rule Values" << endl;
  fp << setw(20) << "x" << setw(20) << "Integral Value" << setw(20) <<
  "N" << endl;

  /* This loop evaluates the integral for 100 equally spaced values of x
  for K(x) using the Trapezoid method. The inner loop uses only enough
  intervals to calculate K(x) to a tolerance of 10^-9.
  */
  for (x = xmin; x <= xmax; x += (xmax - xmin) / pts) {
    i = 2; // second fewest number of intervals to check against tol
    while (i <= NMax && abs(trapezoid(Kx, thetaA, thetaB, i, x) -
    trapezoid(Kx, thetaA, thetaB, i / 2, x)) > tol) {
      i = 2 * i;
    }
    fp << setw(20) << x << setw(20) << trapezoid(Kx, thetaA, thetaB, i, x)
    << setw(20) << i << endl;
  }

  fp << setw(45) << "Simpson's Rule Values" << endl;

  /* This loop evaluates the integral for 100 equally spaced values of x for
  K(x) using the Simpson's method. The inner loop uses only enough intervals
  to calculate K(x) to a tolerance of 10^-9.
  */
  for (x = xmin; x <= xmax; x += (xmax - xmin) / pts) {
    j = 4; // second fewest number of intervals to check against tol
    while (j <= NMax && abs(simpson(Kx, thetaA, thetaB, j, x) - simpson(Kx,
    thetaA, thetaB, j / 2, x)) > tol) {
      j = 2 * j;
    }
    fp << setw(20) << x << setw(20) << simpson(Kx, thetaA, thetaB, j, x)
    << setw(20) << j << endl;
  }

  fp.close();
  return (EXIT_SUCCESS);
}
```

References

[1] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, editors. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, UK ; New York, 3rd ed edition, 2007. OCLC: ocn123285342.

[2] Rubin H. Landau, Jose Paez, and Christian C. Bordeianu. *A Survey of Computational Physics: Introductory Computational Science*. Princeton University Press, Princeton, 2008.