Program 3

David Miller
MAD5403: Foundations of Computational Math I

January 31, 2018

1 Executive Summary

In this program we investigate the accuracy, refinement process, and difference amongst quadrature methods. Different error tolerances are picked and the amount of sub-intervals needed to satisfy this tolerance are computed. Execution if the code proves these are conservative bounds, but bounds nonetheless. Refinement is applied with this error tolerance and stops whenever an appropriate mesh is created. All of this is shown in this report.

2 Statement of Problem

Quadrature methods are pivotal in todays scientific community. They allow for efficient and accurate methods to obtain integral values when analytical methods can not be used. Among the myriad of quadrature methods we employ five: closed Newton-Cotes that uses two endpoints, closed Newton-Cotes that uses two endpoints and midpoint, open Newton-Cotes that uses the midpoint, open Newton-Cotes that uses points 1/2 and 2/3 in the interval, and gauss Legendre. We care to test these methods against different types of functions to verify that the correct execution of these methods are independent of the function but the error is dependent on the function. Using these functions we verify and analyze accuracy, error, and convergence of the quadrature methods.

3 Description of Mathematics

Quadrature methods provide numerical approximations to definite integrals. This is done by first discretizing our domain $[a, b]$ into a uniform mesh $X = [x_0, \ldots, x_n]$ where $a = x_0 < x_1 < \cdots < x_n = b$. We then care about the error, so using the simple composite errors we can derive lower and upper bounds $E_i = f(a, b, n, f^{(n)}(\eta))$ for our composite methods where n is the amount of sub-intervals and $\min f^{(n)}(\eta)$ and $\max f^{(n)}(\eta)$ are used to set a lower and upper bound, respectively. From E_i we can also determine the amount of sub-intervals needed to satisfy an arbitrary error tolerance.

3.1 Closed Newton-Cotes - End Points (Trapezoidal)

The simple composite closed Newton-Cotes quadrature that uses the two endpoints approximates the integral via

$$I_1 = \int_a^b f(x)\,dx \approx h\frac{f(b)-f(a)}{2}, \quad h = b-a \tag{1}$$

where the error is

$$E = -\frac{h^3}{12}f''(\eta), \quad \eta \in [a,b]. \tag{2}$$

For the composite method we have that

$$I_n = \int_a^b f(x)\,dx \approx \frac{h}{2}\bigg(f(a)+f(b)+2\sum_{i=1}^{n-1} f(x_i)\bigg), \quad h = \frac{b-a}{n} \tag{3}$$

from which we can derive the error via the single composite form

$$E_i = -\frac{h^3}{12}\sum_{i=0}^{n-1} f''(\eta_i), \quad \eta_i \in [x_i, x_{i+1}] \tag{4}$$

$$= -\frac{h^3}{12}nf''(\mu), \qquad \mu \in [a,b] \tag{5}$$

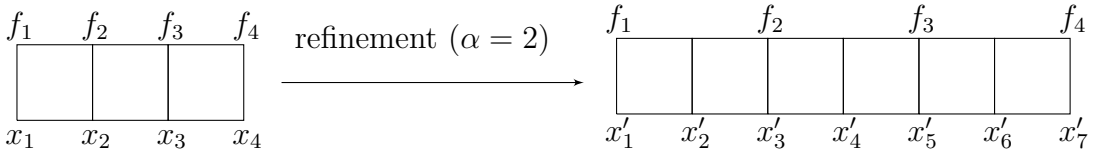
where we used the Mean Value Theorem for (5). We use this to bound our composite error

$$\min\bigg(-\frac{h^3n}{12}\min f''(\mu), -\frac{h^3n}{12}\max f''(\mu)\bigg) \le E_i \le \max\bigg(-\frac{h^3n}{12}\min f''(\mu), -\frac{h^3n}{12}\max f''(\mu)\bigg). \tag{6}$$

From (6) we can derive an expression to tell us how many sub-intervals we need to satisfy some prescribed error tolerance. Simple algebraic manipulation of (4) and (5) yields

$$n = \sqrt{\frac{-(b-a)^3}{12E_i}f''(\mu)} \tag{7}$$

where n will be between two values when we plug in $\min f''(\mu)$ and $\max f''(\mu)$ into (7). Regarding refinement, closed Newton-Cotes using the two endpoints has $\alpha = 2$ and therefore doubles the amount of sub-intervals at each refinement. This can be verified graphically.



where we can see reuse of evaluations can be used when refining.

3.2 Closed Newton Cotes - End Points & Midpoint (Simpson)

The simple composite closed Newton-Cotes quadrature that uses the two endpoints and midpoint approximates the integral via

$$I_1 = \int_a^b f(x)\,dx \approx \frac{h}{3}(f(a) + 4f(\frac{b+a}{2}) + f(b)), \quad h = (b-a)/2 \tag{8}$$

where the error is

$$E = -\frac{h^5}{90} f''''(\eta), \quad \eta \in [a, b]. \tag{9}$$

For the composite method we have that

$$I_n = \int_a^b f(x)\,dx \approx \frac{h}{3}\bigg(f(a) + f(b) + \sum_{i=1}^{n-1} 2f(x_i) + 4f(x_i + h)\bigg), \quad h = \frac{b-a}{2n}, \tag{10}$$

from which we can derive the error via the single composite form

$$E_i = -\frac{h^5}{90} \sum_{i=0}^{n-1} f''(\eta_i), \quad \eta_i \in [x_i, x_{i+1}] \tag{11}$$

$$= -\frac{h^5}{90} n f''(\mu), \qquad \mu \in [a, b] \tag{12}$$

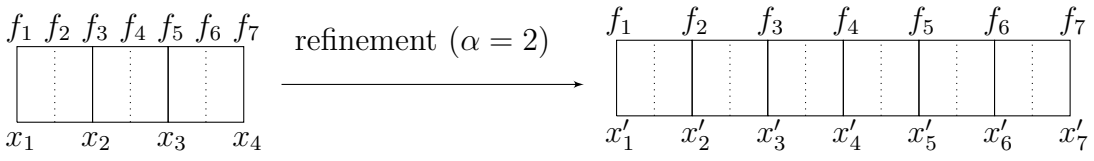
where we used the Mean Value Theorem for (12). We use this to bound our composite error

$$\min\bigg(-\frac{h^5 n}{90}\min f''''(\mu), -\frac{h^5 n}{90}\max f''''(\mu)\bigg) \le E_i \le \max\bigg(-\frac{h^5 n}{90}\min f''''(\mu), -\frac{h^5 n}{90}\max f''''(\mu)\bigg). \tag{13}$$

From (13) we can derive an expression to tell us how many sub-intervals we need to satisfy some prescribed error tolerance. Simple algebraic manipulation of (11) and (12) yields

$$n = \sqrt[4]{\frac{-(b-a)^5}{2880 E_i} f''''(\mu)} \tag{14}$$

where n will be between two values when we plug in $\min f''(\mu)$ and $\max f''(\mu)$ into (14). Regarding refinement, closed Newton-Cotes using the two endpoints has $\alpha = 2$ and therefore doubles the amount of sub-intervals at each refinement. This can be verified graphically.



where we can see reuse of evaluations can be used when refining. We just have to be careful with the weights of the evaluations when we reuse evaluations.

3.3 Open Newton-Cotes - Midpoint (Midpoint)

The simple composite open Newton-Cotes quadrature that uses the midpoint approximates the integral via

$$I_1 = \int_a^b f(x)\,dx \approx 2hf(\frac{a+b}{2}), \quad h = (b-a)/2 \tag{15}$$

where the error is

$$E = \frac{h^3}{3} f''(\eta), \quad \eta \in [a,b]. \tag{16}$$

For the composite method we have that

$$I_n = \int_a^b f(x)\,dx \approx 2h \sum_{i=0}^{n-1} f(\frac{x_i + x_{i+1}}{2}), \quad h = \frac{b-a}{2n} \tag{17}$$

from which we can derive the error via the single composite form

$$E_i = \frac{h^3}{3} \sum_{i=0}^{n-1} f''(\eta_i), \quad \eta_i \in [x_i, x_{i+1}] \tag{18}$$

$$= -\frac{h^3}{3} n f''(\mu), \qquad \mu \in [a,b] \tag{19}$$

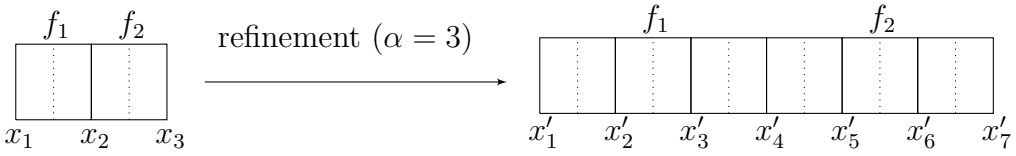
where we used the Mean Value Theorem for (19). We use this to bound our composite error

$$\min\left(\frac{h^3 n}{3}\min f''(\mu), \frac{h^3 n}{3}\max f''(\mu)\right) \leq E_i \leq \max\left(\frac{h^3 n}{3}\min f''(\mu), \frac{h^3 n}{3}\max f''(\mu)\right). \tag{20}$$

From (20) we can derive an expression to tell us how many sub-intervals we need to satisfy some prescribed error tolerance. Simple algebraic manipulation of (18) and (19) yields

$$n = \sqrt{\frac{(b-a)^3}{24E_i} f''(\mu)} \tag{21}$$

where n will be between two values when we plug in $\min f''(\mu)$ and $\max f''(\mu)$ into (21). Regarding refinement, closed Newton-Cotes using the two endpoints has $\alpha = 2$ and therefore doubles the amount of sub-intervals at each refinement. This can be verified graphically.



where we can see reuse of evaluations can be used when refining.

3.4 Open Newton-Cotes - 1/3 & 2/3 Points

The simple composite open Newton-Cotes quadrature that uses points at 1/3 and 2/3 the interval approximates the integral via

$$I_1 = \int_a^b f(x)\,dx \approx \frac{3}{2}h(f(\frac{a+b}{3}) + f(\frac{2(a+b)}{3})), \quad h = (b-a)/3 \tag{22}$$

where the error is

$$E = \frac{3}{4}h^3 f''(\eta), \quad \eta \in [a,b]. \tag{23}$$

For the composite method we have that

$$I_n = \int_a^b f(x)\,dx \approx \frac{3}{2}h\sum_{i=0}^{n-1} f(x_i + h) + f(x_i + 2h/3), \quad h = \frac{b-a}{3n}, \tag{24}$$

from which we can derive the error via the single composite form

$$E_i = \frac{3}{4}\sum_{i=0}^{n-1} f''(\eta_i), \quad \eta_i \in [x_i, x_{i+1}] \tag{25}$$

$$= \frac{3}{4}h^3 n f''(\mu), \qquad \mu \in [a,b] \tag{26}$$

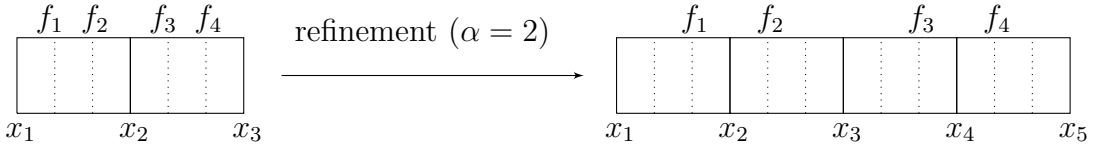
where we used the Mean Value Theorem for (26). We use this to bound our composite error

$$\min\left(\frac{3}{4}h^3 n \min f''(\mu), \frac{3}{4}h^3 n \max f''(\mu)\right) \leq E_i \leq \max\left(\frac{3}{4}h^3 n \min f''(\mu), \frac{3}{4}h^3 n \max f''(\mu)\right). \tag{27}$$

From (27) we can derive an expression to tell us how many sub-intervals we need to satisfy some prescribed error tolerance. Simple algebraic manipulation of (25) and (26) yields

$$n = \sqrt{\frac{(b-a)^3}{36E_i} f''(\mu)} \tag{28}$$

where n will be between two values when we plug in $\min f''(\mu)$ and $\max f''(\mu)$ into (28). Regarding refinement, closed Newton-Cotes using the two endpoints has $\alpha = 2$ and therefore doubles the amount of sub-intervals at each refinement. This can be verified graphically.



where we can see reuse of evaluations can be used when refining.

3.5 Two Point Gauss Legendre

The Gauss-Legendre quadrature (in our case) uses two function evaluations and weights to approximate an integral and thus has degree of exactness 3. Given the general form $\int_{-1}^{1} f(x)\,dx = \omega_1 f(x_1) + \omega_2 f(x_2)$, we have that

$$\int_{-1}^{1} 1dx = \omega_1 + \omega_2 = 2 \tag{29}$$

$$\int_{-1}^{1} x\,dx = \omega_1 x_1 + \omega_2 x_2 = 0 \tag{30}$$

$$\int_{-1}^{1} x^2\,dx = \omega_1 x_1^2 + \omega_2 x_2^2 = \frac{2}{3} \tag{31}$$

$$\int_{-1}^{1} x^3\,dx = \omega_1 x_1^3 + \omega_2 x_2^3 = 0 \tag{32}$$

Solving this system yields

$$\omega_1 = \omega_2 = 1, \quad x_1 = \sqrt{\frac{1}{3}}, \quad x_2 = -\sqrt{\frac{1}{3}} \tag{33}$$

where we can map into our interval $[a, b]$ by

$$x \mapsto \frac{a + b \pm \sqrt{\frac{1}{3}}(b-a)}{2}. \tag{34}$$

The errors for simple composite and composite are

$$E = \frac{2^5(2!)^4}{5(3!)^3} f''''(\eta) \tag{35}$$

$$E_i = \frac{(\frac{b-a}{n})^5(2!)^4}{5(3!)^3} \sum_{i=0}^{n-1} f(\eta_i), \quad \eta_i \in [x_i, x_{i+1}] \tag{36}$$

$$= \frac{(2!)^4}{5(3!)^3}(\frac{b-a}{n})^5 n f''''(\mu), \ \ \mu \in [a, b] \tag{37}$$

To get an estimate on the number of sub-intervals we algebraically manipulate (37) to get

$$n = \sqrt[4]{\frac{(b-a)^5}{4320E_i} f''''(\mu)} \tag{38}$$

It's important to note that there is no α for refinement so mesh doubling is applied during refinement. The function values must just be computed at the new values.

4 Description of the Algorithm and Implementation

NOTE: The current implementation of the code terminates refinement when a certain amount of refinements are done, not when a certain error tolerance is reached. However the code can easily be changed to check against error to terminate the refinement when we have a good enough mesh.

For the following algorithms we have

1. $f(x)$ with discretized domain $X = [x_0, \ldots, x_n]$ such that $a = x_0 < x_n = b$.
2. Error tolerance ϵ ($\epsilon = -1$ if we do not want to refine)
3. Analytic solution $I^* = \int_a^b f(x)\,dx$

where these are used to numerically compute the integral of $f(x)$.

4.1 Trapezoidal Algorithm

Inputs: Mesh X, Function $f(x)$, Max error tolerance ϵ, Analytic solution I^*
Outputs: Quadrature result I
Storage: Mesh $X = [x_0, \ldots, x_n]$ is $\mathcal{O}(n)$, Multiple temporary variables each $\mathcal{O}(1)$
Time: For Loop at line 3 is $\mathcal{O}(n)$, While loop at line 6 is $\mathcal{O}(2^r nr)$, Everything else is $\mathcal{O}(1)$

Algorithm 1 Trapezoidal Method

1: $h \leftarrow x_1 - x_0$, $refinement \leftarrow 0$, $error \leftarrow \infty$
2: $I \leftarrow \frac{1}{2}(f(x_0) + f(x_n))$
3: **for** $i = 1 : 1 : n - 1$ **do**
4: $\quad I \leftarrow I + f(x_i)$
5: **end for**
6: **while** error $> \epsilon$ **do**
7: $\quad h \leftarrow \frac{h}{2}$
8: $\quad$ **for** $i = 1 : 1 : 2^{refinement}(n-1)$ **do**
9: $\quad\quad I \leftarrow I + f(x_0 + (2i-1)h)$
10: $\quad$ **end for**
11: $\quad$ refinement $\leftarrow$ refinement $+1$
12: $\quad$ error $\leftarrow |I^* - I * h|$
13: **end while**
14: $I \leftarrow Ih$
15: **return** I

For the Trapezoidal algorithm the factor $2^r nr$ comes from the refinement. The $2^r n$ is the size of the new mesh at refinement iteration r. The code may seem to have a worst case of exponential but in fact it is just the size of a refined mesh, so it is still linear with respect to the size of the biggest mesh being evaluated.

4.2 Simpson Algorithm

Inputs: Mesh X, Function $f(x)$, Max error tolerance ϵ, Analytic solution I^*
Outputs: Quadrature result I
Storage: Mesh $X = [x_0, \ldots, x_n]$ is $\mathcal{O}(n)$, Multiple temporary variables each $\mathcal{O}(1)$
Time: For loop at line 3 is $\mathcal{O}(n)$, While loop at line 6 is $\mathcal{O}(2^r nr)$, For loop at line 13 is $\mathcal{O}(2^r n)$, Everything else is $\mathcal{O}(1)$

Algorithm 2 Simpson Algorithm

1: $h \leftarrow x_1 - x_0$, $refinement \leftarrow 0$, $error \leftarrow \infty$
2: $I \leftarrow f(x_0) + f(x_n) + f(x_1 + \frac{h}{2})$
3: **for** $i = 1 : 1 : n - 1$ **do**
4: $\quad I \leftarrow I + 2(f(x_i)_f(x_i + \frac{h}{2}))$
5: **end for**
6: **while** error $> \epsilon$ **do**
7: $\quad h \leftarrow \frac{h}{2}$
8: $\quad$ refinement $\leftarrow$ refinement $+1$
9: $\quad$ **for** $i = 1 : 2 : 2^{refinement}(n - 1)$ **do**
10: $\quad\quad I^* \leftarrow I^* + 2f(x_0 + i\frac{h}{2})$
11: $\quad$ **end for**
12: $\quad$ error $\leftarrow |I^* - I * h/6|$
13: **end while**
14: **for** $i = 1 : 2 : 2^{refinement+1}(n - 1)$ **do**
15: $\quad I \leftarrow I + 2f(x_0 + i\frac{h}{2})$
16: **end for**
17: $I \leftarrow I\frac{h}{6}$
18: **return** I

For the Simpson algorithm the factor $2^r nr$ comes from the refinement. The $2^r n$ is the size of the new mesh at refinement iteration r. The code may seem to have a worst case of exponential but in fact it is just the size of a refined mesh, so it is still linear with respect to the size of the biggest mesh being evaluated.

4.3 Midpoint Algorithm

Inputs: Mesh X, Function $f(x)$, Max error tolerance ϵ, Analytic solution I^*
Outputs: Quadrature result I
Storage: Mesh $X = [x_0, \ldots, x_n]$ is $\mathcal{O}(n)$, Multiple temporary variables each $\mathcal{O}(1)$
Time: For loop at line 2 is $\mathcal{O}(n)$, While loop at line 5 is $\mathcal{O}(3^r nr)$, Everything else is $\mathcal{O}(1)$

Algorithm 3 Midpoint Algorithm

1: $h \leftarrow x_1 - x_0$, $refinement \leftarrow 0$, $error \leftarrow \infty$
2: **for** $i = 0 : 1 : n - 1$ **do**
3: $\quad I \leftarrow I + f(x_i + \frac{h}{2})$
4: **end for**
5: **while** error $> \epsilon$ **do**
6: $\quad h \leftarrow \frac{h}{2}$
7: $\quad$ refinement $\leftarrow$ refinement $+1$
8: $\quad$ **for** $i = 0 : 1 : 3^{refinement}(n-1)$ **do**
9: $\quad\quad$ **if** $i + 2 \not\equiv 0 \pmod 2$ **then**
10: $\quad\quad\quad I \leftarrow I + f(x_0 + h(i + \frac{1}{2}))$
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: $\quad$ error $\leftarrow |I^* - I * h|$
14: **end while**
15: $I \leftarrow Ih$
16: **return** I

For the Midpoint algorithm the factor $3^r nr$ comes from the refinement. The $3^r n$ is the size of the new mesh at refinement iteration r. The code may seem to have a worst case of exponential but in fact it is just the size of a refined mesh, so it is still linear with respect to the size of the biggest mesh being evaluated.

4.4 Open Newton-Cotes Algorithm (1/3 & 2/3 Points)

Inputs: Mesh X, Function $f(x)$, Max error tolerance ϵ, Analytic solution I
Outputs: Quadrature result I
Storage: Mesh $X = [x_0, \ldots, x_n]$ is $\mathcal{O}(n)$, Multiple temporary variables each $\mathcal{O}(1)$
Time: For loop at line 2 is $\mathcal{O}(n)$, While loop at line 5 is $\mathcal{O}(2^r nr)$, Everything else is $\mathcal{O}(1)$

Algorithm 4 Open Newton-Cotes Algorithm

1: $h \leftarrow x_1 - x_0$, $refinement \leftarrow 0$, $error \leftarrow \infty$
2: **for** $i = 0 : 1 : n - 1$ **do**
3: $\quad I \leftarrow I + f(x_i + \frac{h}{3}) + f(x_i + \frac{2h}{3})$
4: **end for**
5: **while** error $> \epsilon$ **do**
6: $\quad h \leftarrow \frac{h}{2}$
7: $\quad$ refinement $\leftarrow$ refinement $+1$
8: $\quad$ **for** $i = 0 : 1 : 2^{refinement}(n-1)$ **do**
9: $\quad\quad I \leftarrow I + f(x_0 + h(2i + \frac{1}{3})) + f(x_0 + h(2i + \frac{5}{3}))$
10: $\quad$ **end for**
11: $\quad$ error $\leftarrow |I^* - I * h/2|$
12: **end while**
13: $I \leftarrow I\frac{h}{2}$
14: **return** I

For the Open Newton-Cotes (1/3 & 2/3) algorithm the factor $2^r nr$ comes from the refinement. The $2^r n$ is the size of the new mesh at refinement iteration r. The code may seem to have a worst case of exponential but in fact it is just the size of a refined mesh, so it is still linear with respect to the size of the biggest mesh being evaluated.

4.5 Gauss-Legendre Algorithm

Inputs: Mesh X, Function $f(x)$, Max error tolerance ϵ, Analytic solution I
Outputs: Quadrature result I^*
Storage: Mesh $X = [x_0, \ldots, x_n]$ is $\mathcal{O}(n)$, Multiple temporary variables each $\mathcal{O}(1)$
Time: For loop at line 2 is $\mathcal{O}(n)$, While loop at line 5 is $\mathcal{O}(2^r nr)$, Everything else is $\mathcal{O}(1)$

Algorithm 5 Gauss-Legendre Algorithm

1: $h \leftarrow x_1 - x_0$, $refinement \leftarrow 0, x_1 \leftarrow \sqrt{\frac{1}{3}}$, $x_2 \leftarrow -\sqrt{\frac{1}{3}}$, $error \leftarrow 0$
2: **for** $i = 1 : 1 : n - 1$ **do**
3: $\quad x_1 \leftarrow (x_{i+1} + x_i + (x_{i+1} - x_i)\sqrt{1/3})/2$
4: $\quad x_2 \leftarrow (x_{i+1} + x_i - (x_{i+1} - x_i)\sqrt{1/3})/2$
5: $\quad I \leftarrow I + f(x_1) + f(x_2)$
6: **end for**
7: **while** error $> \epsilon$ **do**
8: $\quad I \leftarrow 0$
9: $\quad h \leftarrow h/2$
10: $\quad$ **for** $i = 0 : 1 : 2^{refinement}(n - 1)$ **do**
11: $\quad\quad x_1 \leftarrow (2x_0 + ih + h\sqrt{1/3})/2$
12: $\quad\quad x_2 \leftarrow (2x_0 + ih - h\sqrt{1/3})/2$
13: $\quad\quad I \leftarrow I + f(x_1) + f(x_2)$
14: $\quad$ **end for**
15: **end while**
16: $I \leftarrow I * \frac{h}{2}$
17: **return** I

For the Gauss-Legendre algorithm the factor $2^r nr$ comes from the refinement. The $2^r n$ is the size of the new mesh at refinement iteration r. The code may seem to have a worst case of exponential but in fact it is just the size of a refined mesh, so it is still linear with respect to the size of the biggest mesh being evaluated.

5 Description of the Experiment Design and Results

For this program we use the following functions

1. $f_1(x) = e^x, \quad x \in [0, 3]$
2. $f_2(x) = e^{sin(2x)}cos(2x), \quad x \in [0.\frac{\pi}{3}]$
3. $f_3(x) = tanh(x), \quad x \in [-2, 1]$
4. $f_4(x) = xcos(2\pi x), \quad x \in [0, 3.5]$
5. $f_5(x) = x + \frac{1}{x}, \quad x \in [0.1, 2.5]$

Error analysis requires minimum and maximum values of the second and fourth derivatives of our functions so that we can give a bound. Below are the computed min and max using Wolfram|Alpha

	$\min f''(\mu)$	$\max f''(\mu)$	$\min f''''(\mu)$	$\max f''''(\mu)$
$f_1(x) = e^x$	1	20.0855	1	20.0855
$f_2(x) = e^{(sin(2x))}cos(2x)$	-16.2831	13.542	-396.975	396.975
$f_3(x) = tanh(x)$	-0.7698	-0.0524725	-4.0859	4.08589
$f_4(x) = xcos(2\pi x)$	-81.1495	138.174	-5454.91	3350.51
$f_5(x) = x + \frac{1}{x}$	0.128	2000	0.24576	2400000

Table 1: Min and max values of second and fourth derivatives for functions integrated

For this program we want to do the following

- Compute the interval size required for a relative error of 1% and an absolute error of 0.001 and verify numerically via simple composite
- Verify that refinement stops when error is within the tolerance
- Show that some methods are inherently better than others

The next section presents the results and small discussions which prove the above points.

6 Results

6.1 Function 1

For $f(x) = e^x$ we have that

$$I = \int_0^3 e^x \, dx = 19.085537 \tag{39}$$

where we use this to compare our quadrature results. If we want a relative error of 1%, then our quadrature result I_n must be in the interval $[.99I, 1.01I] \approx [18.894682, 19.276392]$ and if we want a relative error of 0.001 we want our quadrature I_n to be in the interval $[19.084537, 19.086537]$. Estimating amount of sub-intervals via equations (7), (14), (21), (28), and (38) for relative error of 1% and absolute error of 0.001 we get

	Relative error 1%	Absolute Error 0.001
n_{Trap}	16	213
$n_{Simpson}$	2	7
$n_{Midpoint}$	11	151
n_{Open2}	9	123
$n_{Gauss-Legendre}$	2	6

Table 2: Number of sub-intervals for relative error of 1% and absolute error of 0.001

where we use $\max|f''(\mu)|$ and $\max|f''''(\mu)|$ to get a conservative bound. We get the following results

I_{16}^{Trap}	I_{213}^{Trap}	$I_{2}^{Simpson}$	$I_{7}^{Simpson}$	$I_{11}^{Midpoint}$	$I_{151}^{Midpoint}$	I_{9}^{Open2}	I_{123}^{Open2}	I_{2}^{GL}	I_{6}^{GL}
.055863	.000363	.091463	.000263	.023037	.000337	.058737	.000337	.020837	.000237

Table 3: Errors of predicted n for relative and absolute error

It is easy to believe to see that these n values satisfy our error requirements. It is easy to believe this since we used $\max f''(\mu)$ and $\max f''''(\mu)$ to determine n which yields a conservative number.

	Trapezoidal	Simpson	Midpoint	Open 2	Gauss Legendre
$n = 1/1$	31.6283	19.5061	13.4451	15.161	18.8101
$n = 4/9$	19.9719	19.0876	18.9975	18.7913	19.0842
$n = 16/81$	19.1414	19.0855	19.0844	19.0669	19.0855
$n = 64/729$	19.089	19.0855	19.0855	19.0844	19.0855
$n = 256/6561$	19.0858	19.0855	19.0855	19.0855	19.0855

Table 4: Results of quadrature with refinement. First n is for methods with $\alpha = 1/2$ (and Gauss-Legendre) and second n is for methods with $\alpha = 1/3$.

6.2 Function 2

For $f(x) = e^{(sin(2x))}cos(2x)$ we have that

$$I = \int_0^{\pi/3} e^{(sin(2x))cos(2x)}\, dx = 0.68872134 \tag{40}$$

where we use this to compare our quadrature results. If we want a relative error of 1%, then our quadrature result I_n must be in the interval $[.99I, 1.01I] \approx [0.68183413, 0.69560855]$ and if we want a relative error of 0.001 we want our quadrature I_n to be in the interval $[0.68772134, 0.68972134]$. Estimating amount of sub-intervals via equations (7), (14), (21), (28), and (38) for relative error of 1% and absolute error of 0.001 we get

	Relative error 1%	Absolute Error 0.001
n_{Trap}	16	40
$n_{Simpson}$	3	4
$n_{Midpoint}$	11	28
n_{Open2}	9	23
$n_{Gauss-Legendre}$	8	4

Table 5: Number of sub-intervals for relative error of 1% and absolute error of 0.001

where we use $\max|f''(\mu)|$ and $\max|f''''(\mu)|$ to get a conservative bound. We get the following results

I_{16}^{Trap}	I_{213}^{Trap}	$I_{2}^{Simpson}$	$I_{7}^{Simpson}$	$I_{11}^{Midpoint}$	$I_{151}^{Midpoint}$	I_{9}^{Open2}	I_{123}^{Open2}	I_{8}^{GL}	I_{4}^{GL}
.0017603	.000281	.000195	.000065	.001866	.000288	.001859	.000284	.000002	.000042

Table 6: Errors of predicted n for relative and absolute error

It is easy to believe to see that these n values satisfy our error requirements. It is easy to believe this since we used $\max f''(\mu)$ and $\max f''''(\mu)$ to determine n which yields a conservative number.

	Trapezoidal	Simpson	Midpoint	Open 2	Gauss Legendre
$n = 1/1$	-0.0988143	0.796946	1.24483	1.009623	0.611318
$n = 4/9$	0.660309	0.688786	0.691511	0.69823	0.688679
$n = 16/81$	0.686961	0.688722	0.688756	0.689308	0.688721
$n = 64/729$	0.688611	0.688721	0.688722	0.688758	0.688721

Table 7: Results of quadrature with refinement. First n is for methods with $\alpha = 1/2$ (and Gauss-Legendre) and second n is for methods with $\alpha = 1/3$.

6.3 Function 3

For $f(x) = tanh(x)$ we have that

$$I = \int_{-2}^{1} tanh(x)\, dx = -0.89122192 \tag{41}$$

where we use this to compare our quadrature results. If we want a relative error of 1%, then our quadrature result I_n must be in the interval $[.99I, 1.01I] \approx [-0.90013414, -0.8823097]$ and if we want a relative error of 0.001 we want our quadrature I_n to be in the interval $[-0.89222192, -0.89022192]$. Estimating amount of sub-intervals via equations (7), (14), (21), (28), and (38) for relative error of 1% and absolute error of 0.001 we get

	Relative error 1%	Absolute Error 0.001
n_{Trap}	9	42
$n_{Simpson}$	3	5
$n_{Midpoint}$	10	30
n_{Open2}	9	24
$n_{Gauss-Legendre}$	3	4

Table 8: Number of sub-intervals for relative error of 1% and absolute error of 0.001

where we use $\max|f''(\mu)|$ and $\max|f''''(\mu)|$ to get a conservative bound. We get the following results

I_{16}^{Trap}	I_{213}^{Trap}	$I_{2}^{Simpson}$	$I_{7}^{Simpson}$	$I_{11}^{Midpoint}$	$I_{151}^{Midpoint}$	I_{9}^{Open2}	I_{123}^{Open2}	I_{3}^{GL}	I_{4}^{GL}
.003228	.000149	.000186	.000020	.001306	.000145	.001075	.000151	.000127	.000030

Table 9: Errors of predicted n for relative and absolute error

It is easy to believe to see that these n values satisfy our error requirements. It is easy to believe this since we used $\max f''(\mu)$ and $\max f''''(\mu)$ to determine n which yields a conservative number.

	Trapezoidal	Simpson	Midpoint	Open 2	Gauss Legendre
$n = 1/1$	-0.30365	-1.02545	-1.38635	-1.14239	-0.790909
$n = 4/9$	-0.875024	-0.891177	-0.892833	-0.896595	-0.891252
$n = 16/81$	-0.890199	-0.891222	-0.891242	-0.891563	-0.891222
$n = 64/729$	-0.891158	-0.891222	-0.891222	-0.891243	-0.891222

Table 10: Results of quadrature with refinement. First n is for methods with $\alpha = 1/2$ (and Gauss-Legendre) and second n is for methods with $\alpha = 1/3$.

6.4 Function 4

For $f(x) = xcos(2\pi x)$ we have that

$$I = \int_0^{3.5} xcos(2\pi x)\, dx = -0.05066059 \tag{42}$$

where we use this to compare our quadrature results. If we want a relative error of 1%, then our quadrature result I_n must be in the interval $[.99I, 1.01I] \approx [-0.0511672, -0.05015398]$ and if we want a relative error of 0.001 we want our quadrature I_n to be in the interval $[-0.05166059, -0.049966059]$. Estimating amount of sub-intervals via equations (7), (14), (21), (28), and (38) for relative error of 1% and absolute error of 0.001 we get

	Relative error 1%	Absolute Error 0.001
n_{Trap}	988	703
$n_{Simpson}$	38	32
$n_{Midpoint}$	699	497
n_{Open2}	570	406
$n_{Gauss-Legendre}$	28	29

Table 11: Number of sub-intervals for relative error of 1% and absolute error of 0.001

where we use $\max|f''(\mu)|$ and $\max|f''''(\mu)|$ to get a conservative bound. We get the following results

I_{16}^{Trap}	I_{213}^{Trap}	$I_{2}^{Simpson}$	$I_{7}^{Simpson}$	$I_{11}^{Midpoint}$	$I_{151}^{Midpoint}$	I_{9}^{Open2}	I_{123}^{Open2}	I_{28}^{GL}	I_{29}^{GL}
.000002	.000004	.000006	.000012	.000002	.000005	.000002	.000005	.000014	.000012

Table 12: Errors of predicted n for relative and absolute error

It is easy to believe to see that these n values satisfy our error requirements. It is easy to believe this since we used $\max f''(\mu)$ and $\max f''''(\mu)$ to determine n which yields a conservative number.

	Trapezoidal	Simpson	Midpoint	Open 2	Gauss Legendre
$n = 1/1$	-6.125	-2.04167	2.62534E-12	-1.02083	0.230093
$n = 4/9$	-2.61401	0.738683	-0.0292888	1.20564	-0.616779
$n = 16/81$	-0.0594496	-0.0504533	-0.050504	-0.0476094	-0.0507996
$n = 64/729$	-0.051162	-0.0506599	-0.0506587	-0.050493	-0.0506611
$n = 256/6561$	-0.0506918	-0.0506606	-0.0506606	-0.0506502	-0.0506606
$n = 1024/59049$	-0.0506625	-0.0506606	-0.0506606	-0.0506606	-0.0506606

Table 13: Results of quadrature with refinement. First n is for methods with $\alpha = 1/2$ (and Gauss-Legendre) and second n is for methods with $\alpha = 1/3$.

6.5 Function 5

For $f(x) = x + \frac{1}{x}$ we have that

$$I = \int_{0.1}^{2.5} x + \frac{1}{x}\, dx = 6.3388758 \tag{43}$$

where we use this to compare our quadrature results. If we want a relative error of 1%, then our quadrature result I_n must be in the interval $[.99I, 1.01I] \approx [6.27548704, 6.40226456]$ and if we want a relative error of 0.001 we want our quadrature I_n to be in the interval $[6.3378758, 6.3398758]$. Estimating amount of sub-intervals via equations (7), (14), (21), (28), and (38) for relative error of 1% and absolute error of 0.001 we get

	Relative error 1%	Absolute Error 0.001
n_{Trap}	191	1518
$n_{Simpson}$	32	91
$n_{Midpoint}$	135	1074
n_{Open2}	111	877
$n_{Gauss-Legendre}$	39	82

Table 14: Number of sub-intervals for relative error of 1% and absolute error of 0.001

where we use $\max|f''(\mu)|$ and $\max|f''''(\mu)|$ to get a conservative bound. We get the following results

I_{16}^{Trap}	I_{213}^{Trap}	$I_{2}^{Simpson}$	$I_{7}^{Simpson}$	$I_{11}^{Midpoint}$	$I_{151}^{Midpoint}$	I_{9}^{Open2}	I_{123}^{Open2}	I_{39}^{GL}	I_{82}^{GL}
.001314	.000024	.000514	.000014	.001306	.000016	.001286	.000016	.000166	.000006

Table 15: Errors of predicted n for relative and absolute error

It is easy to believe to see that these n values satisfy our error requirements. It is easy to believe this since we used $\max f''(\mu)$ and $\max f''''(\mu)$ to determine n which yields a conservative number.

	Trapezoidal	Simpson	Midpoint	Open 2	Gauss Legendre
$n = 1/1$	15.6	8.51077	4.96615	5.15922	5.69851
$n = 4/9$	7.87447	6.53664	6.16254	5.97582	6.23859
$n = 32/243$	6.38352	6.33939	6.33847	6.32431	6.33854
$n = 256/6561$	6.33961	6.33888	6.33888	6.33863	6.33888
$n = 1024/59049$	6.33892	6.33888	6.33888	6.33886	6.33888
$n = 2048/177147$	6.33889	6.33888	6.33888	6.33887	6.33888

Table 16: Results of quadrature with refinement. First n is for methods with $\alpha = 1/2$ (and Gauss-Legendre) and second n is for methods with $\alpha = 1/3$.

7 Conclusion

Quadrature methods provide efficient and accurate results for integration over finite domains. Proper error analysis can tell the user much about what to expect

1. How many sub-intervals are need for a certain error tolerance,
2. When to expect exact answers from quadrature,
3. What methods are better suited for the user.

In this program we proved that the five quadrature methods have a bounded error which can be derived, certain methods provided more accurate results than other, number of sub-intervals tell us how accurate we will be, and degree of exactness allows for exact integration. In addition to this, the results from the five functions gave insight into how each method works and how efficient reuse of function evaluation can be taken advantage of. This is important because sometimes it takes a great deal of processing power, and time, to compute function evaluations.