Project 02

APSC 607 Fall 2017

Seth Goodman

October 25, 2017

1 Introduction

This project explored methods for calculating the integral roots of functions. Each functions was examined in the range between zero and two, using the Composite Trapezoidal Rule, Composite Midpoint Rule, Composite Simpson's Rule, as well as an adaptive implementation of the Composite Simpson's Rule. The behavior and characteristics of these methods are reviewed by examining the effectiveness of the resulting value for the integral given a range of values for N.

All computations were performed using MATLAB using the code accompanying this report (in a zip file). The following *Methods* section will present the methods used in MATLAB to explore functions, as well as the outputs and results. The *Results* section of this report contains the outputs for each function along with related observations and discussion. All figures and tables found in this report are available in the output subdirectory of the accompanying zip file. Additionally, all code and figures found in the zip file can be accessed via GitHub[1].

[1]`https://github.com/sgoodm/apsc607/tree/master/project_02`

2 Methods

Equations 1 and 2 define the two function, hereafter referred to as Function **A** and **B**, respectively, which are examined in this project. Three different composite numerical integration approaches - Trapezoidal, Midpoint, and Simpson's - will be tested to examine the effectiveness of each method. A fourth adaptive approach based on Simpson's Rule will also be tested for comparison. Integration will be restricted to between zero and two for testing, but integration over an expanded range and the potential utility of adaptive approaches in such a scenario will be discussed in the *Results* section.

$$f(x) = e^{2x} * sin(3x) \tag{1}$$

$$f(x) = \frac{1}{x+4} \tag{2}$$

To establish a baseline, the true value for the integral of each function is first calculated using built in MATLAB tools. The integral is calculated both using the symbolic toolkit function **int** as well as the numerical function **integral**. The resulting values can be seen in Table 1.

Function	**Symbolic**	**Numeric**
A	-14.2139771298625	-14.2139771298625
B	0.405465108108164	0.405465108108164

Table 1: True values of integrals between zero and two

The true value for Functions A and B will be compared to the results of integration using the Trapezoidal, Midpoint, and Simpson's Rules for a range of subintervals defined as **n**. The range of **n** will vary with each function and rule tested, in order to achieve accuracy within a tolerance of $1e^{-8}$.

The value of **n** required to reach the specified tolerance is dependent on the function itself as well as the error term associated with each rule. The error terms for the Trapezoidal, Midpoint, and Simpson's Rules are defined in Equation 3, 4, and 5 respective (Burden and Faires, 2010). As the Trapezoidal and Midpoint Rules have second order error terms, it is expected that they will require a greater value of **n** to produce results comparable to Simpson's Rule, which has a fourth order error term.

$$\frac{b-a}{6} h^2 f''(u) \tag{3}$$

$$\frac{b-a}{12} h^2 f''(u) \tag{4}$$

$$\frac{h^5}{90} f^{(4)}(\xi_j) \tag{5}$$

All values of **n** tested will be positive and even, as it is required for the Midpoint and Simpson's Rules. Although the Trapezoidal Rule can be performed using odd intervals, using only even intervals will provide sufficient sample points for analysis. In order to test **n** over a sufficient range for the Trapezoidal and Midpoint Rules, a binary expansion was used to generate values of **n** at which to sample the full range, rather than test at every possible step. Starting with two sub intervals ($n = 2$), if every even sub interval value up to 100,000 was tested it would require 50,000 points. By incorporating a binary expansion based approach, a sample of only 20 different **n** values can generated which are capable of sufficiently demonstrating the behavior of the integration rules.

Due to the higher order error term, Simpson's Rule can reach the desired tolerance much quicker and a simple range of $2 : 2 : nmax$ can be used. Finally, no **n** value needs to be specified for the adaptive approach because, as seen in Subsection 2.4, this approach iteratively checks that a specified tolerance has been reached before producing an output rather than using a set **n** value.

The remainder of this section will introduce the three composite numerical integration approaches (Trapezoidal, Midpoint, and Simpson's) as well as an adaptive approach using Simpson's Rule.

2.1 Composite Trapezoidal Rule

TEXT

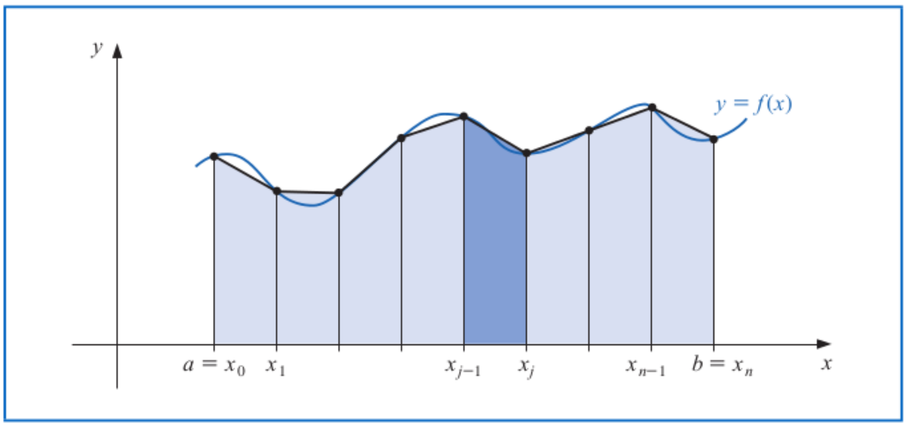

Figure 1: Trapezoidal Figure (Burden and Faires, 2010)

The Composite Trapezoidal Rule for n intervals, as seen in Figure 1, can be defined by Equation 6, given $h = (b - a)/n$ and $x_j = a + jh$, for each $j = 0, 1, \ldots, n$ (Burden and Faires, 2010).

$$\int_a^b f(x)dx = \frac{h}{2}\left[f(a) + 2\sum_{j=1}^{n-1} f(x_j) + f(b)\right] \tag{6}$$

Implementing this in a MATLAB function is extremely straightforward. The function accepts a function handle **f** defining $f(x)$, (e.g., $cos(x)$), the desired number of internval **n**, along with our bounds **rmin** and **rmax**. The value for **h** given **n** is calculated, as are the vectors **j** and x_j. Using these components and the summation function, the final integral for the input conditions can be calculated. The function then returns the integral value, along with the value of **h** used.

This function implementing the Trapezoidal Rule for integration (as well as subsequent Midpoint and Simpson's Rules) is called over a range of values for **n** using the **arrayfun** function in MATLAB. This function accepts another function, defining our integration rule, and a vector, and simply repeatedly calls the specified value while iterating over the values in the vector. The call to **arrayfun** then return a vector (or set of vectors in this case) containg the results from each call to the integration function.

The resulting vector of integral values across varying **n** can compared with the true integral value generated earlier, to produce an error vector. The error vector is used to identify the value of **n** (and thus **h**) at which the integration rule produced results that were accurate within a desired tolerance.

2.2 Composite Midpoint Rule

TEXT

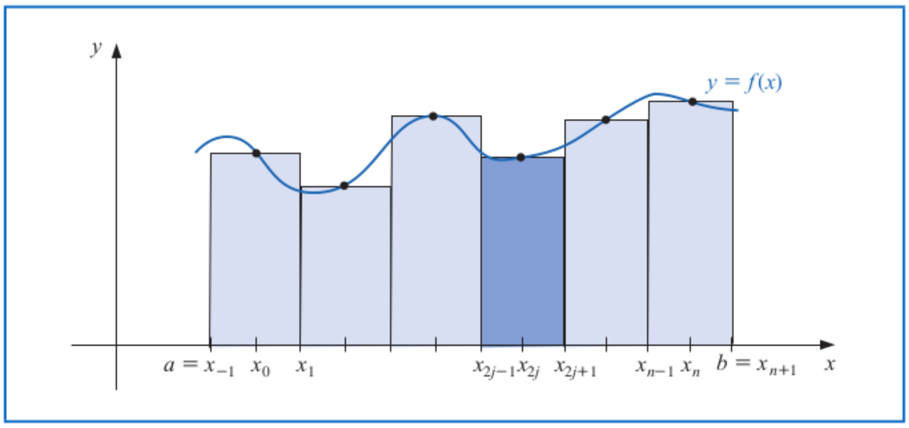

Figure 2: Midpoint Figure (Burden and Faires, 2010)

The Composite Midpoint Rule for $n+2$ intervals, as seen in Figure 2, can be defined by Equation 7, given $h = (b-a)/(n+2)$ and $x_j = a + (j+1)h$, for each $j = -1, 0, \ldots, n+1$ (Burden and Faires, 2010).

$$\int_a^b f(x)dx = 2h \sum_{j=0}^{n/2} f(x_2 j) \tag{7}$$

2.3 Composite Simpson's Rule

TEXT

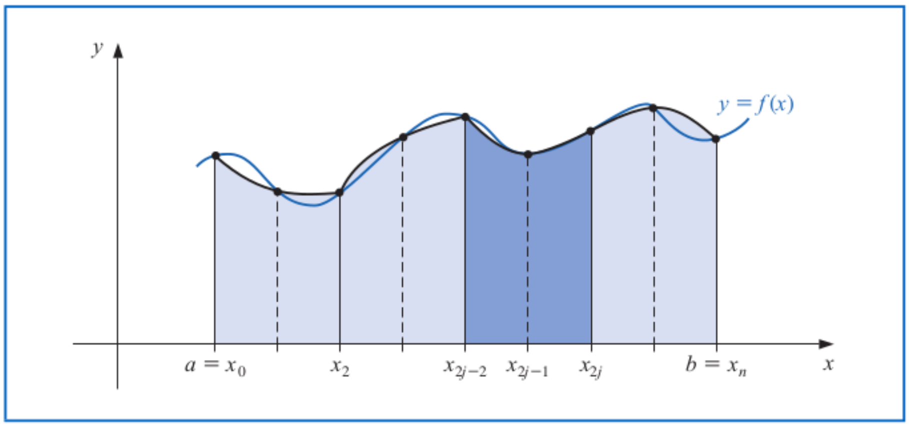

Figure 3: Simpson's Figure (Burden and Faires, 2010)

The Composite Simpson's Rule for n intervals, as seen in Figure 3, can be defined by Equation 8, given $h = (b-a)/n$ and $x0_j = a+jh$, $x1_j = a+jh+h/2$, $x2_j = a + jh + h$, for each $j = 0, 1, \dots, n-1$.

$$\int_a^b f(x)dx = \sum_{j=0}^{n-1} \left[\frac{h}{6} \left[f(x0_j) + 4f(x1_j) + f(x2_j) \right] \right] \tag{8}$$

2.4 Adaptive Simpson's Rule

TEXT

3 Results

TEXT

3.1 A

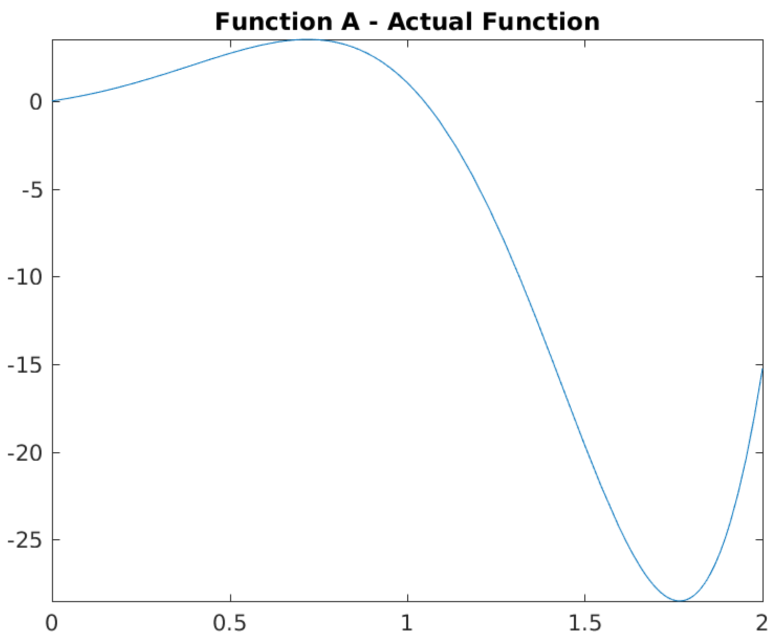

Figure 4: caption text a

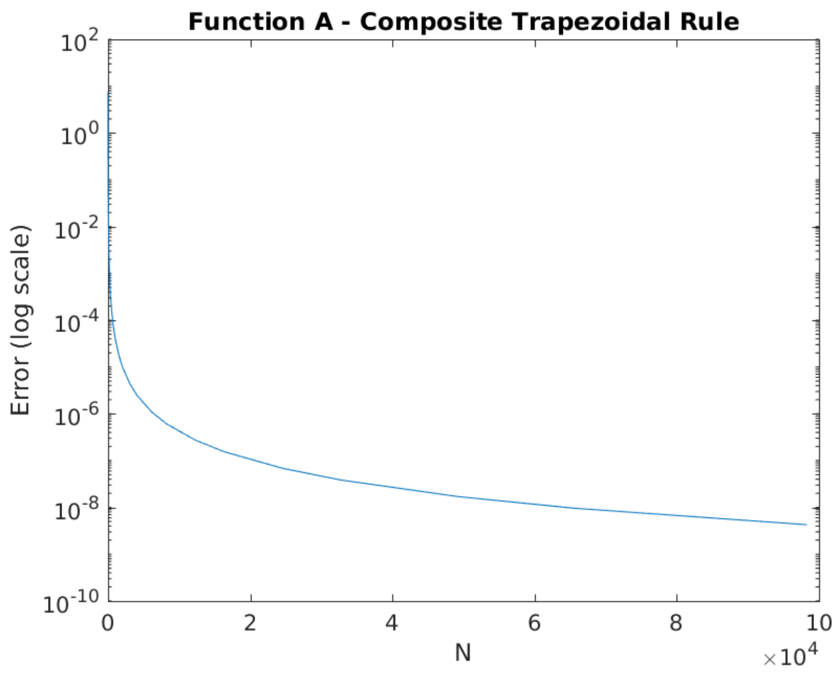

Figure 5: caption text a

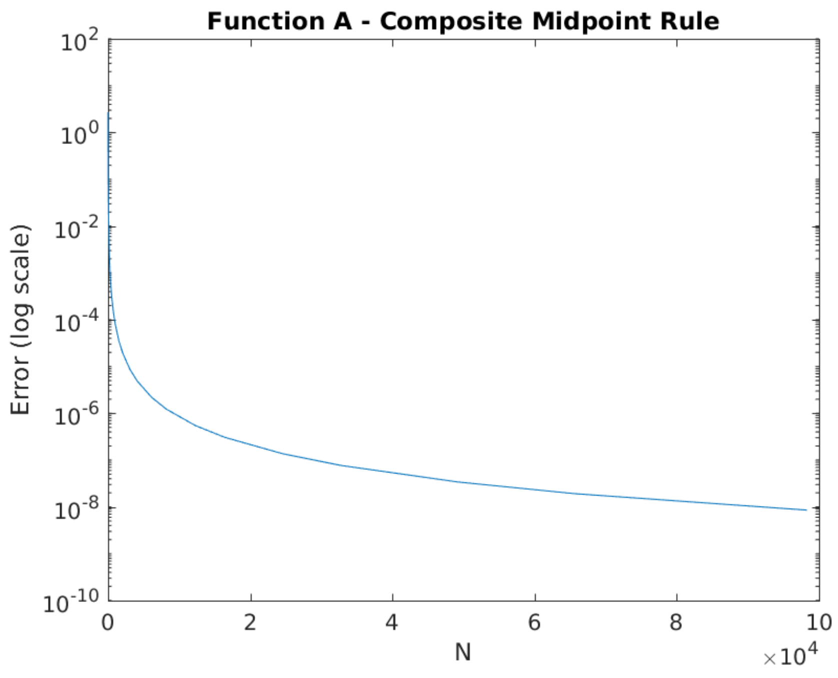

Figure 6: caption text a

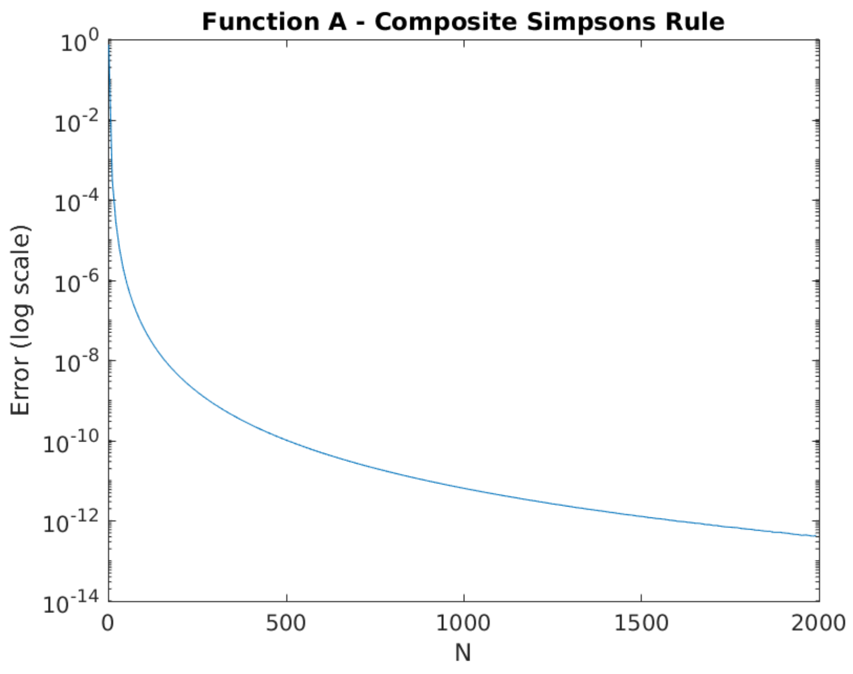

Figure 7: caption text a

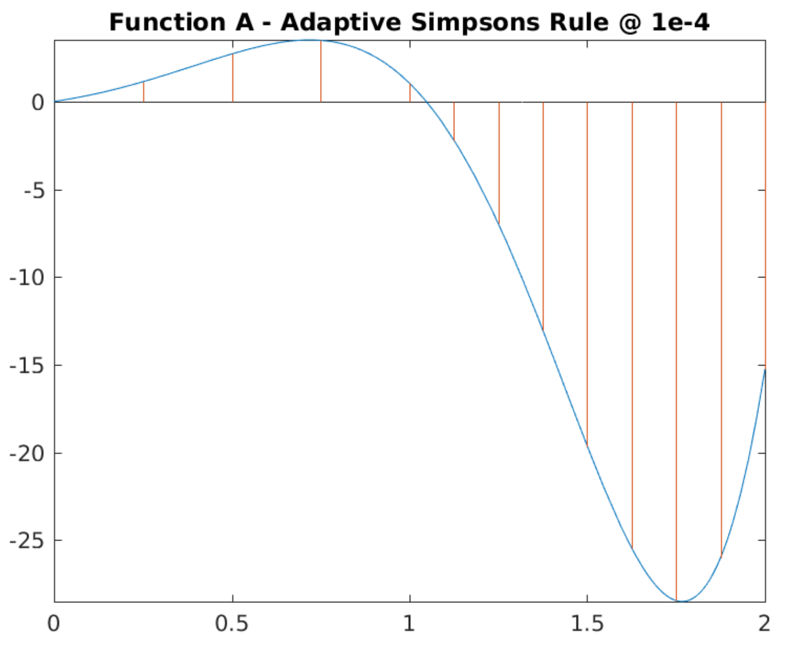

Figure 8: caption text a

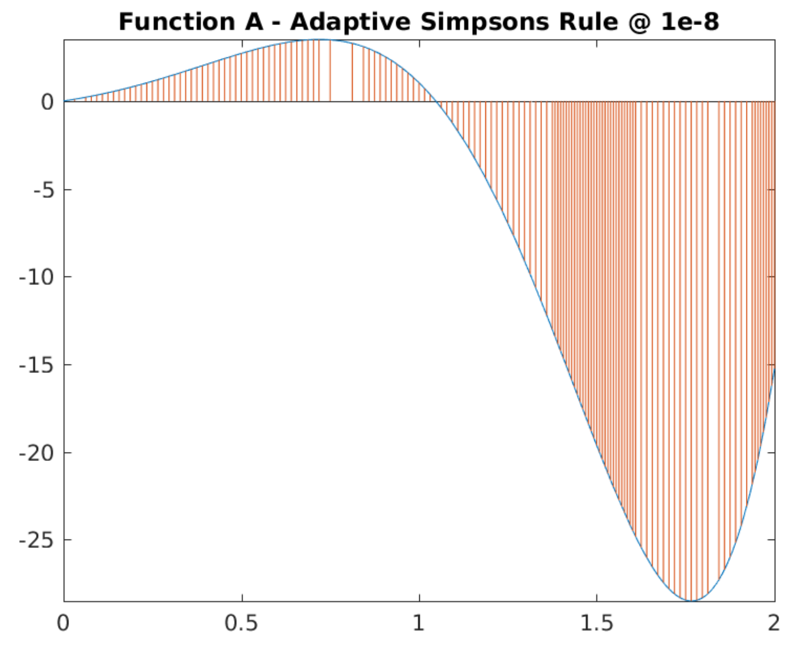

Figure 9: caption text a

3.2 B

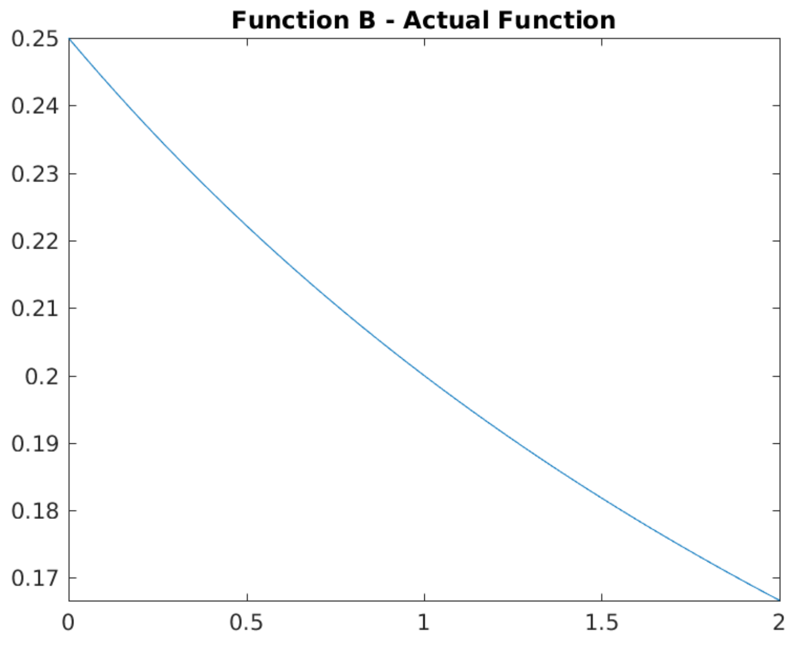

Figure 10: caption text a

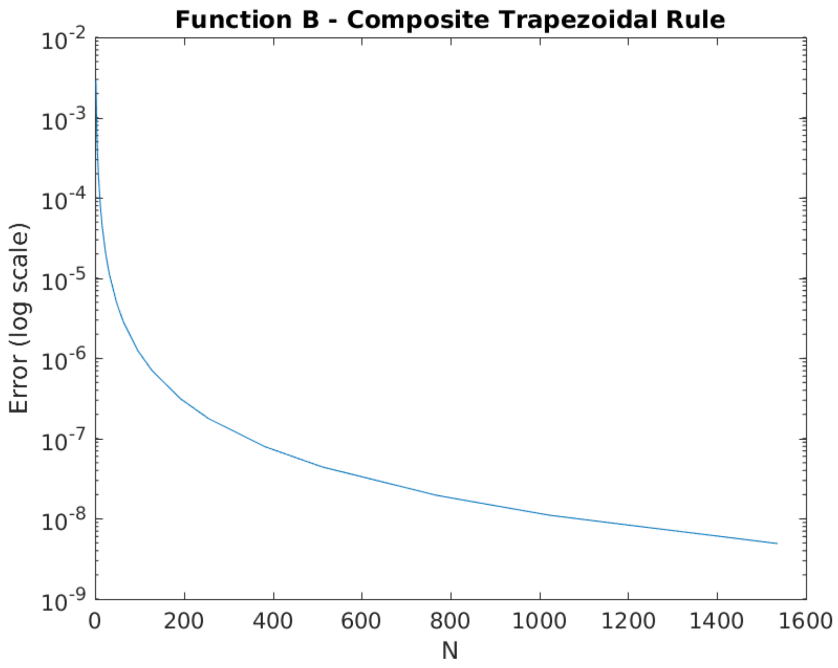

Figure 11: caption text a

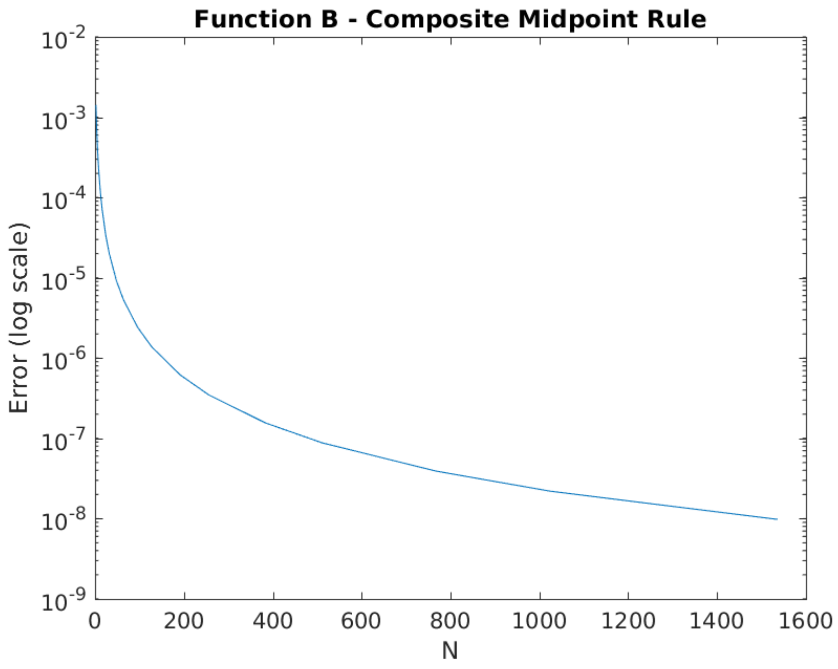

Figure 12: caption text a

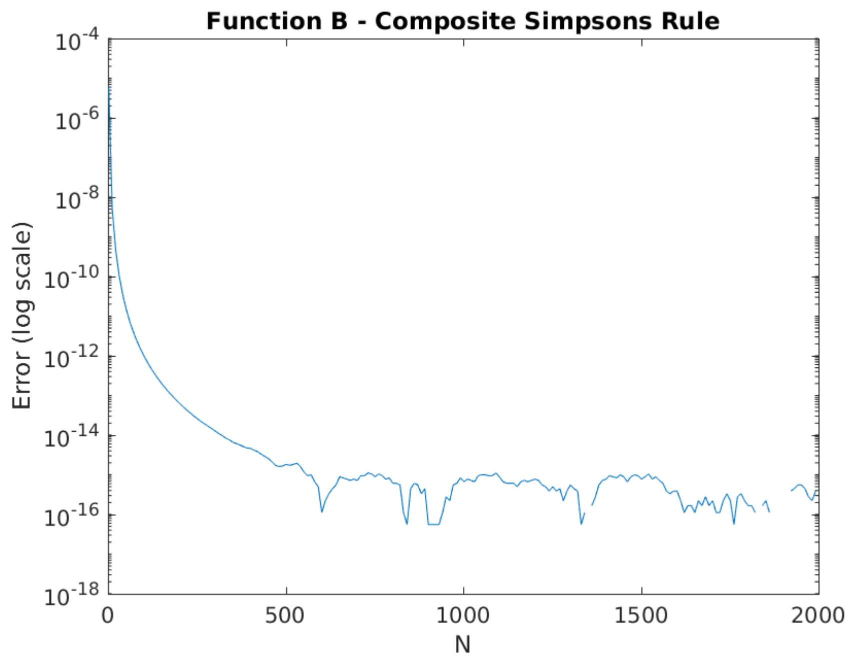

Figure 13: caption text a

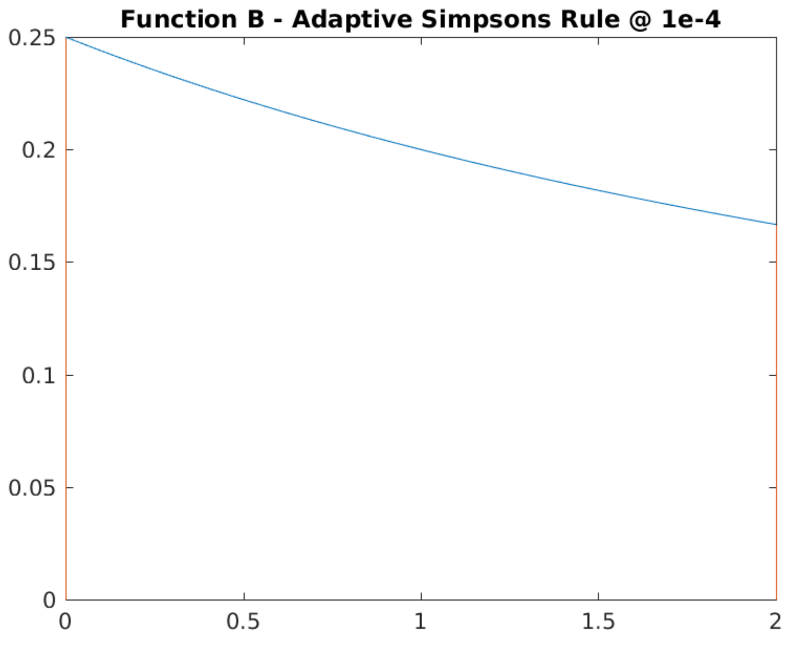

Figure 14: caption text a

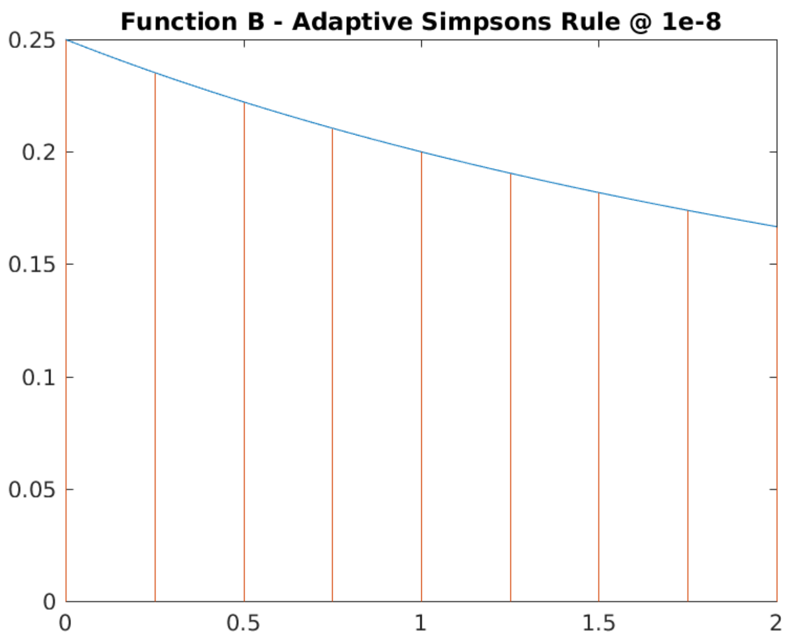

Figure 15: caption text a

3.3 Expanded Range

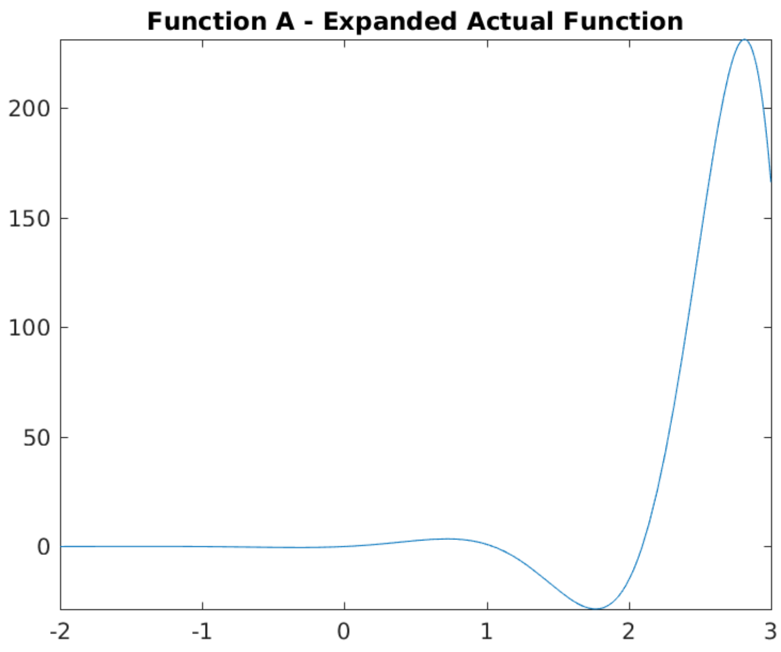

Figure 16: caption text a

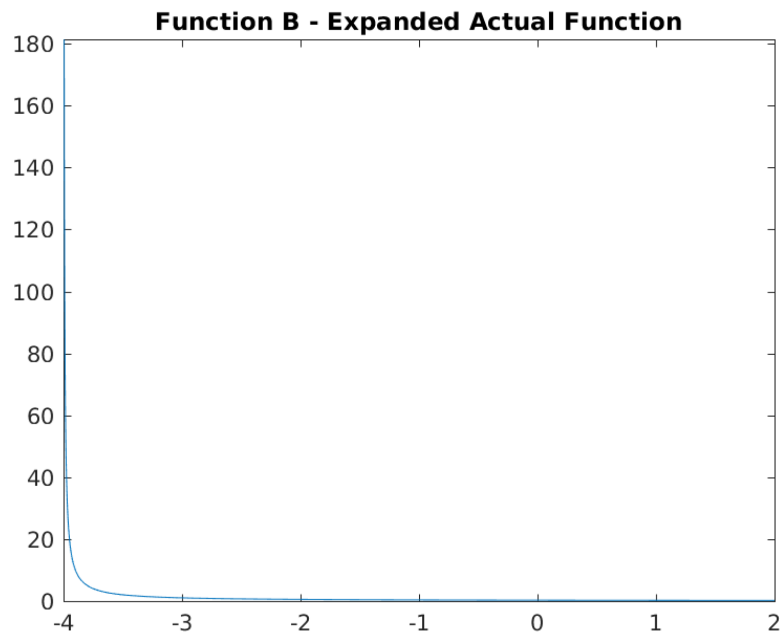

Figure 17: caption text a

Something about minimum error in footnote[2].

[2]See following link on MATLAB precision limitions (general limitations of floating point representations apply) https://www.mathworks.com/help/fixedpoint/ug/limitations-on-precision.html

References

R. Burden and J Faires. *Numerical Analysis*. Brooks/Cole, 9th edition edition, 2010.