UNIVERSITY OF CALIFORNIA, RIVERSIDE

BOURNS COLLEGE OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

EE 105 Lab 1 Solution

MATLAB as an Engineer's Problem Solving Tool

LUIS FERNANDO ENRIQUEZ-CONTRERAS

Contents

List of Figures

Listings

1 Introduction

This introductory MATLAB laboratory course equips students with fundamental skills in scientific computing. Through hands-on exercises, students will gain proficiency in:

- Matrix algebra: Manipulating and analyzing numerical data represented as matrices
- Function creation: Defining and implementing custom functions for specific math equations
- Data visualization: Generating informative plots and graphs to interpret results
- Computational optimization: Employing techniques to improve the efficiency of numerical algorithms

By applying these acquired skills, students will develop a technical report that clearly communicates the purpose, methodology, and outcomes of the MatLab program. This report will showcase his or her understanding of MATLAB and its capabilities in solving scientific and engineering problems.

2 Matrices and Arrays

Listing 1: Matrix Multiplication

```
clear;
clc;
% Matrix Multiplication
% Declare A Matrix (3 x 1 Matrix)
A = [sqrt(2); 1; exp(pi)];
% Declare B Matrix (1 x 3 Matrix)
B = [3;5;7];
% Convert A Matix from a column to a row vector and muliple with column
% vector b to get scalar value C
C = A.' * B;
display(C);
```

C = 171.2275

3 Scripts

Listing 2: Matrix Multiplication using a For Loop

```
clear;
clc;
% Matrix Multiplication
% Declare A Matrix (3 x 1 Matrix)
A = [sqrt(2); 1; exp(pi)];
% Declare B Matrix (1 x 3 Matrix)
B = [3;5;7];
% Set D to zero to start the for loop
D = 0;
% Assumes both vector A and B are of the same length
for i = 1:length(A)
    D = D + A(i)*B(i); % Add previous D value (sum of previous ith
        elemenents) to the cuurent ith elemement multiplication
end
display(D);
```

The for loop and the matrix multiplication arrive at the same result. However, matrix multiplication requires less code and is less computationally intensive than the for loop. C = 171.2275

4 More Advanced Scripts

4.1 $f(x_i)$ Function

Listing 3: $f(x_i)$ Function

```
% Input column vector x and an output column vector y where the i-th element
    of the output vector is y(i) = fx(i)
function [y] = fx(x)
    y = cos(x) ./  (1 + exp(3 * x)); % Outcolumn is calculated use vector
        multiplication
end
```

4.2 Plot $f(x_i)$

Listing 4: Plot Function for $f(x_i)$

```
function plot_export(N, title_text, filename) % Input N for the number
   spacings between the elements, title_text for the name of the plots, and
```

```
the filename in order to export the plot. Note each plot must have a unique filename
x = linspace(0, 5, N + 1);
y = fx(x);
plot(x, y); % Creates the line plot
grid on;
title(title_text,'Interpreter','latex'); % Adds title to the plot. Adding $$ between the text, and adding 'Interperter', 'latex' to the matlab function, creates text with LaTeX formatting. Alternatively you may use title('title_name').
xlabel('$X$','Interpreter','latex'); % Adds xlabel to the plot. Adding $$ between the text, and adding 'Interperter', 'latex' to the matlab function, creates text with LaTeX formatting. Alternatively you may use xlabel('X').
ylabel('$f(x)$','Interpreter','latex'); % Adds ylabel to the plot. Adding $$ between the text, and adding 'Interperter', 'latex' to the matlab function, creates text with LaTeX formatting. Alternatively you may use ylabel('Y').
grid on;
fig = gcf; % Obtains current graphic in matlab
exportgraphics(fig, append(filename, '.pdf'),'ContentType','vector'); % Exports plot as a vector pdf image. (Requires R2020a or later)
end
```

Listing 5: Code to run function for $f(x_i)$

```
clear;
clc;
% This code automatically calculates, labels, and plots
N = [200, 5, 10]; % Declare the N that will be ran
for i = N % iterate through given N for example i = 200
    title_text = {['$f(x)$ vs $X$'] [append('$N = ', string(i), '$') ]}; % Create the the title for each i. Convert the integer to a string and add it to the title
    filename = append('Fig/fx_', string(i)); % Add number to the filename to ensure each plot has a unique filename
    plot_export(i, title_text, filename); % With i, title_text, and filename determined run and export documentex each plot. Vector pdf plots should be in the same folder as the code location ready to be inserted to a tex document
end
```

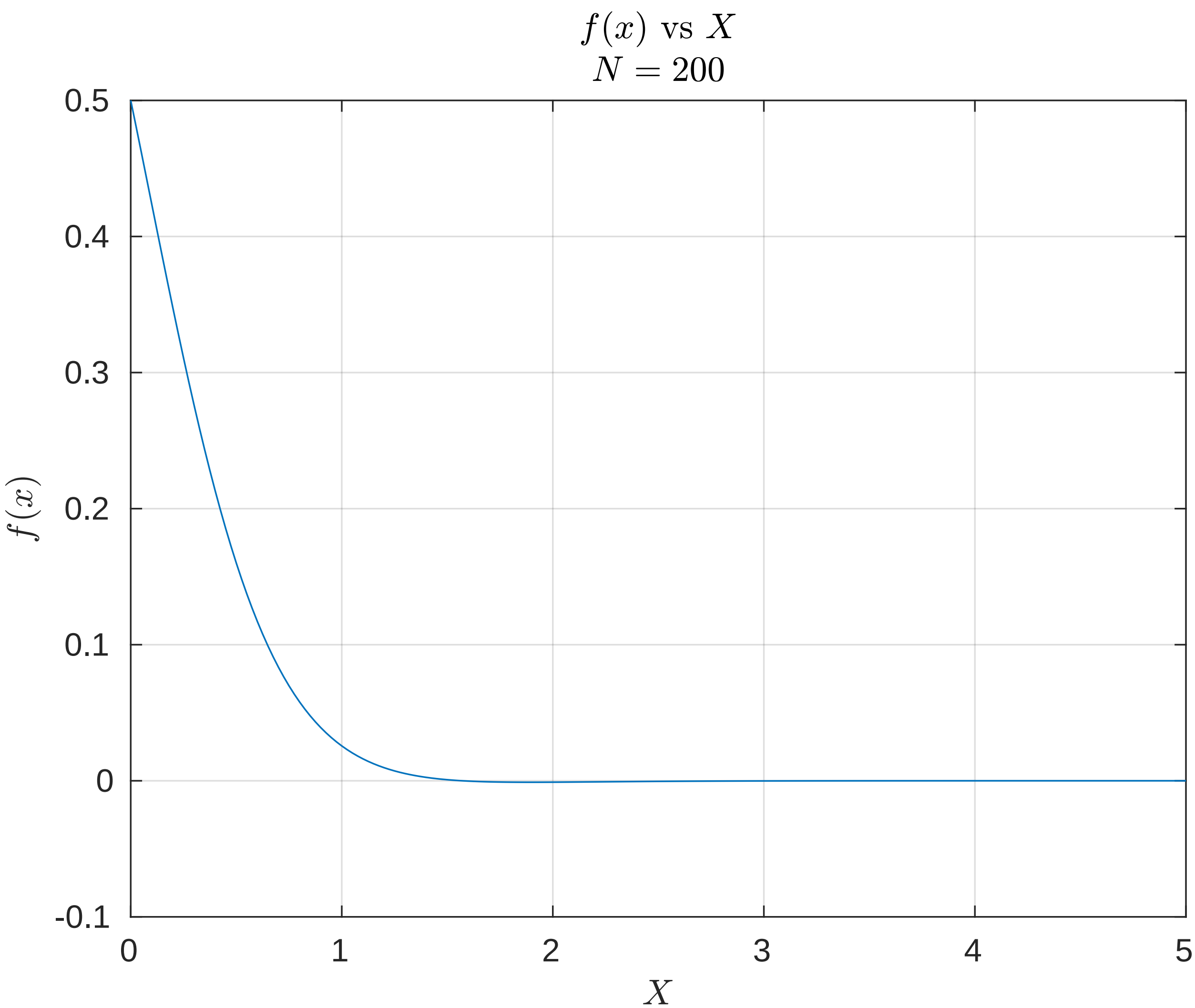

Figure 1: $f(x_i) = \frac{cos(x_i)}{1+e^{3x_i}}$ for $N = 200$

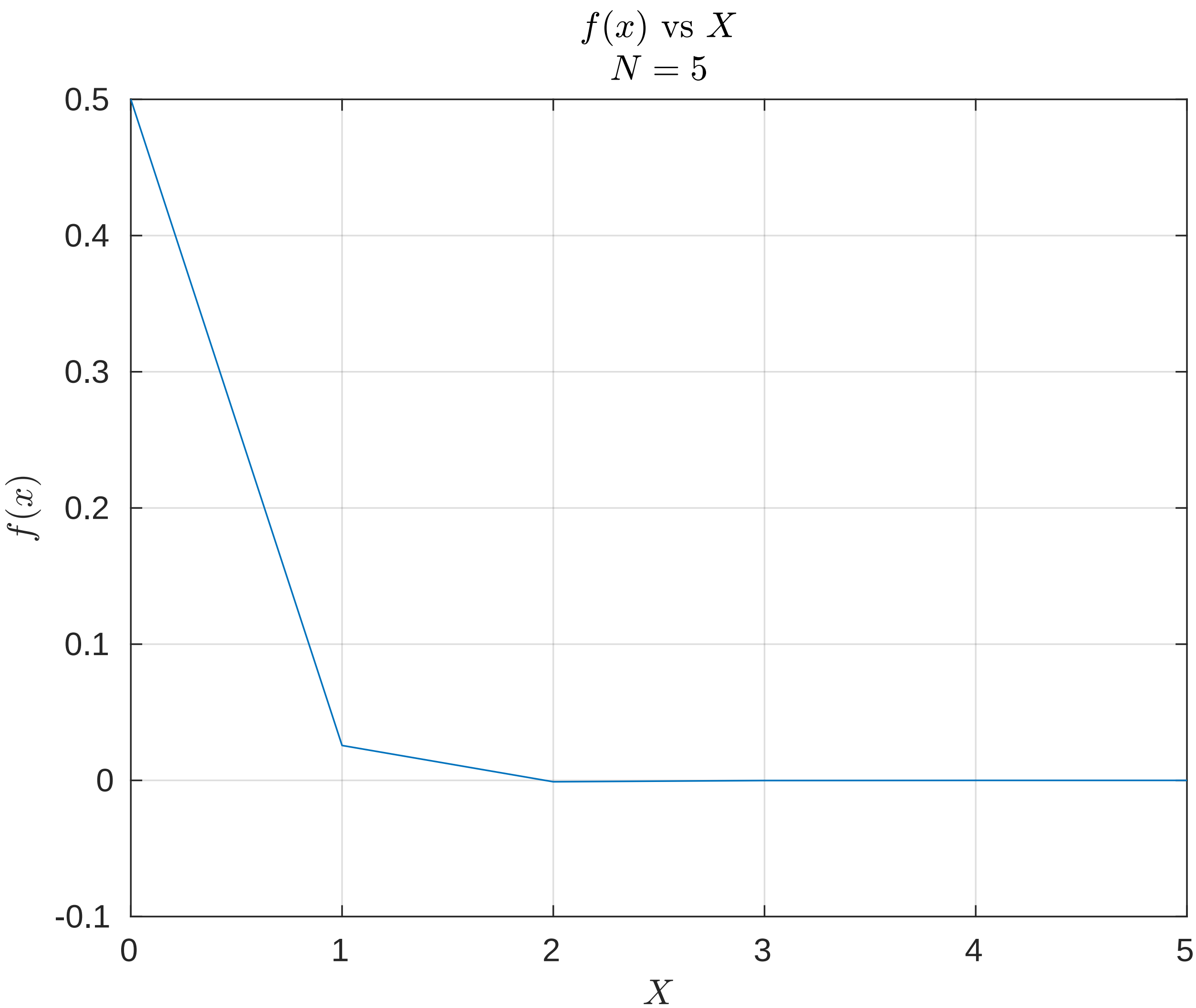

Figure 2: $f(x_i) = \frac{cos(x_i)}{1+e^{3x_i}}$ for $N = 5$

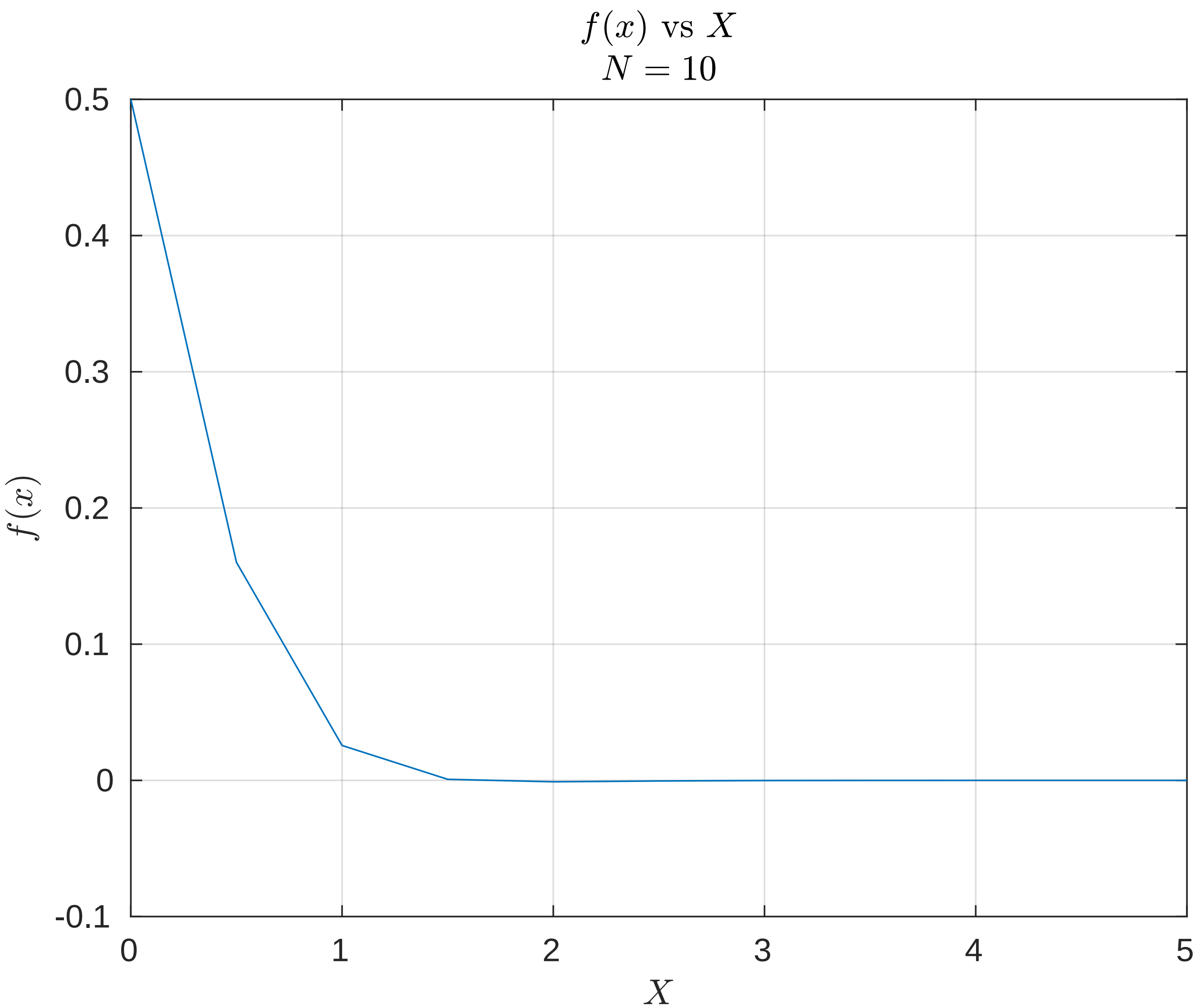

Figure 3: $f(x_i) = \frac{cos(x_i)}{1+e^{3x_i}}$ for $N = 10$

Increasing the number of points (N+1) within the fixed sized region [0,5] for x decreases the space between points, which enhances the resolution. Figure 1 exhibits a high domain resolution, resulting in a smooth and continuous plot devoid of discernible edges. Figures 2 and 3 conversely, demonstrate the effects of a lower domain resolution, characterized by visually apparent edges and a less refined plot appearance.

5 Area under the Curve

5.1 Standard Integration

Listing 6: $\int_0^5 \frac{cos(x)}{1+e^{3x}} dx$

```
clear;
clc;
% Find the integral using quad
Q = quad(@fx, 0, 5); % Function, lower bound, upper bound
```

$$\int_0^5 \frac{cos(x)}{1+e^{3x}} dx \approx 0.201$$

5.2 Riemann Integral Approximation Equations

5.2.1 Right Rectangular Approximation

Listing 7: $\int_0^5 \frac{cos(x)}{1+e^{3x}} dx$

```
clear;
clc;
for i = [25, 250]
    N = i; % 5 recantangles for every integer value
    dx = 5/N; % divide by the range
    x_bar = dx:dx:5; % x domain for bar plot.
    y_bar = fx(x_bar); % range for bar plot
    x_line = 0:1/1000:5; % Add a high resolution domain
    y_line = fx(x_line); % range for scatter plot
    bar(x_bar-dx/2, y_bar, 1, 'green'); % Creates the bar plot
    hold on; % Add a second line to the plot
    plot(x_line, y_line, 'red'); % Creates the scatter plot
    title({['Right rectangular approximation $Q_{2}$'] [append('$N = ',
        string(i), '$')]},'Interpreter','latex'); % Adds title to the plot.
        Adding $$ between the text, and adding 'Interperter', 'latex' to the
        matlab function, creates text with LaTeX formatting
    xlabel('$X$','Interpreter','latex'); % Adds xlabel to the plot. Adding
        $$ between the text, and adding 'Interperter', 'latex' to the matlab
        function, creates text with LaTeX formatting
```

```
    ylabel('$y$','Interpreter','latex'); % Adds ylabel to the plot. Adding $$ between the text, and adding 'Interperter', 'latex' to the matlab function, creates text with LaTeX formatting
    legend('$Q_{2}$', '$f(x)$','Interpreter','latex'); % Adds legend to the plot. Adding $$ between the text, and adding 'Interperter', 'latex' to the matlab function, creates text with LaTeX formatting
    grid on;
    fig = gcf; % Obtains current graphic in matlab
    exportgraphics(fig, append('Fig/q2_bar_plot_', string(i) ,'.pdf'),'ContentType','vector'); % Exports plot as a vector pdf image. (Requires R2020a or later)
end
```

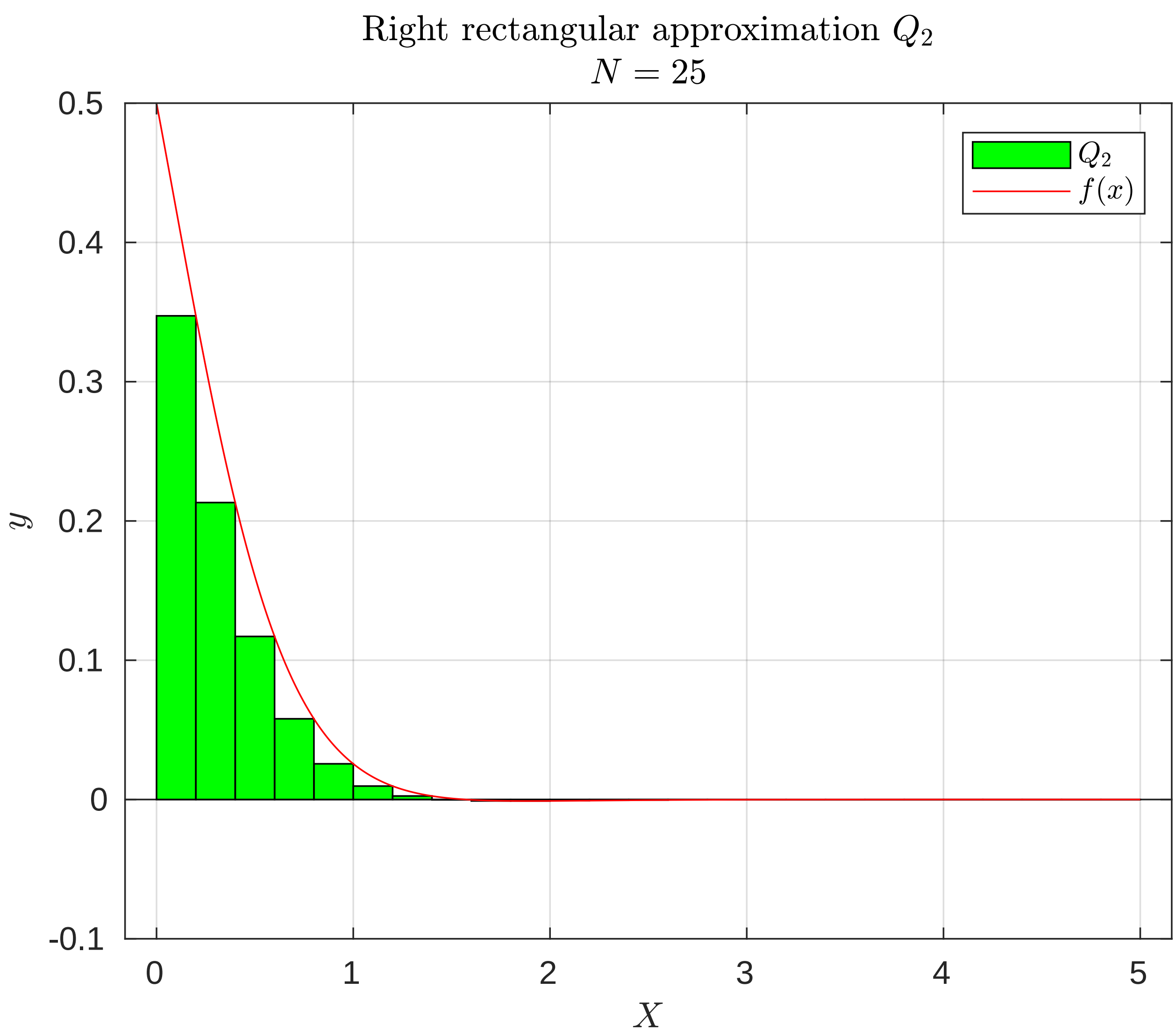

Figure 4: Right rectangular approximation for Q_2 for $N = 25$

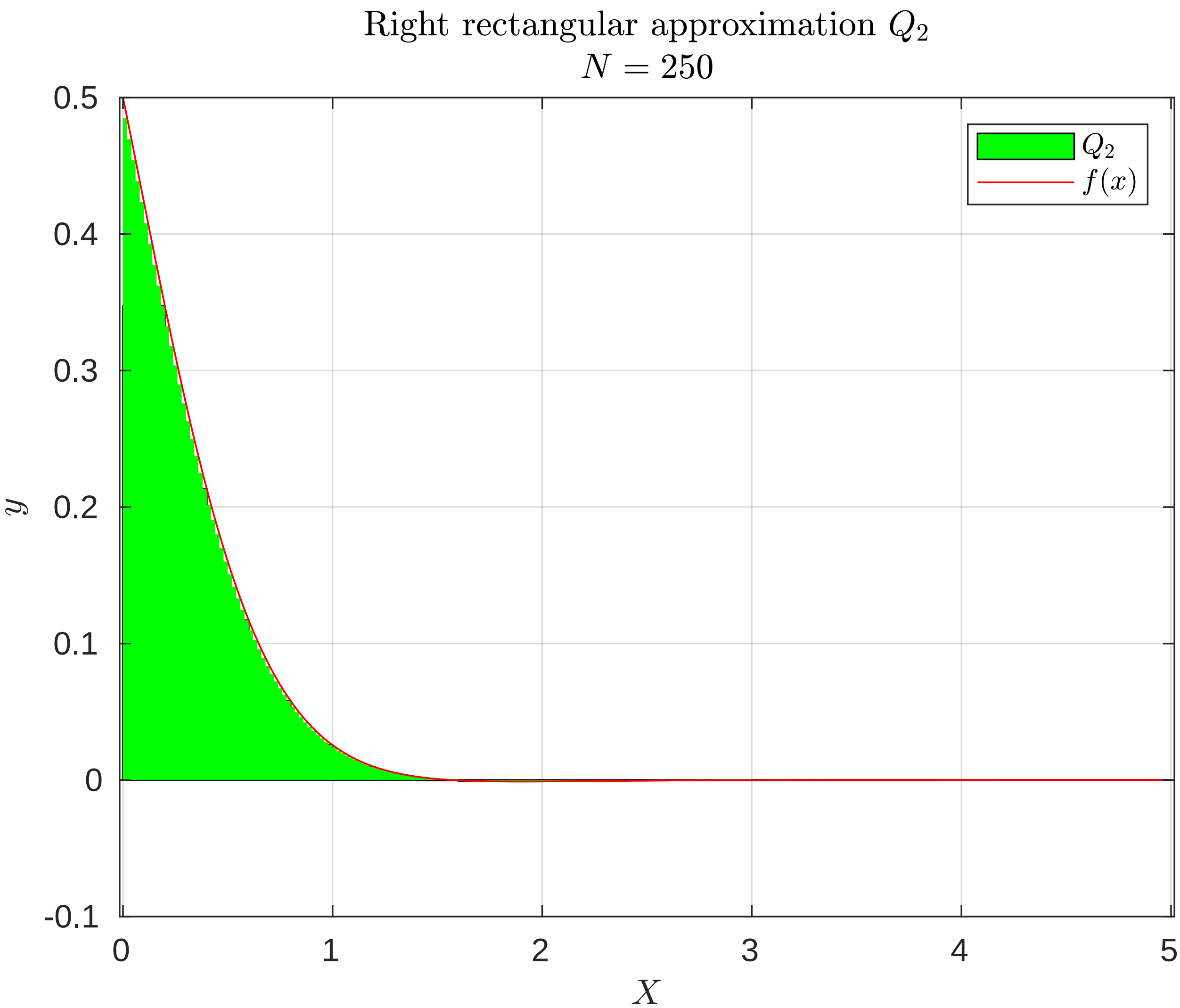

Figure 5: Right rectangular approximation for Q_2 for $N = 250$

5.2.2 Tradeoffs

Deceasing dx increases the resolution of the domain and the accuracy of the approximation. However, a higher resolution require more computational power. This is akin to higher resolution images requiring more space.

5.2.3 Trapazoidal approximation derivation

Given that the area of the first rectangle can be calculated:

$$f(x_0)\,dx_0 + 0.5\left(f(x_1) - f(x_0)\right)dx_0$$
$$f(x_0)\,dx_0 + \frac{f(x_1)}{2}dx_0 - \frac{f(x_0)}{2}dx_0$$

This means that the next area can be approximated:

$$f(x_1)\,dx_1 + \frac{f(x_2)}{2}dx_1 - \frac{f(x_1)}{2}dx_1$$

This can continue so forth until the N-th area can be approximated as:

$$f(x_{N-1})\,dx_{N-1} + \frac{f(x_N)}{2}dx_{N-1} - \frac{f(x_{N-1})}{2}dx_{N-1}$$

Summing these terms up yields the equation Q_3

$$Q_3 = f(x_0)\frac{dx_0}{2} + \sum_{i=1}^{N-1} f(x_i)\,dx_i + f(x_N)\frac{dx_{N-1}}{2}$$

5.2.4 $Q_3 = \frac{Q_1+Q_2}{2}$

Given that:

$$Q_1 = \sum_{i=1}^{N-1} f(x_i)\,dx_i$$

$$Q_2 = \sum_{i=2}^{N} f(x_i)\,dx_{i-1}$$

$$Q_3 = f(x_1)\frac{dx_1}{2} + \sum_{i=2}^{N-1} f(x_i)\,dx_i + f(x_N)\frac{dx_{N-1}}{2}$$

All dx_i are equal to each other. Substituting in Q_1 and Q_2 in to Q_3 gives:

$$Q_3 = \frac{1}{2}\left(\sum_{i=1}^{N-1} f(x_i)\,dx_i + \sum_{i=2}^{N} f(x_i)\,dx_{i-1}\right)$$
$$Q_3 = f(x_1)\frac{dx_1}{2} + \sum_{i=2}^{N-1} f(x_i)\,dx_i + f(x_N)\frac{dx_{N-1}}{2}$$

5.2.5 Q_1 and Q vs N

Listing 8: Q_1 Function

```
function Q_1 = q1_sum(N)
dx = (5-0)/(N-1);
x = 0:dx:5;
y = fx(x);
A = [ones(N-1, 1); 0];
Q_1 = y*A*dx;
end
```

Listing 9: Plot Q_1 and Q vs N

```
clear;
clc;
% N starts at 2 and ends at 200
N  = 2:1:200;
% Domain for Q1
Q_1N = zeros(1, length(N));
% Populate each ith value with the future q1 N calculation
for i = N
    Q_1N(i-1) = q1_sum(i);
end
% Create the domain for for the integral calculation
Q_N = ones(1, length(N));
% Find the integral using quad
Q = quad(@fx, 0, 5); % Function, lower bound, upper bound
% Create a vector of size N with the same Q value for each
% value
Q_N = Q_N*Q;
% Create the domain for for error calculation
Q_1error = zeros(1, length(N));
% Subtract the summation from the actual value to calculate the error
Q_1error = Q_1N - Q_N;
% Declare the figure
figure;
% Declare the subplot
% First subplot
subplot(2, 1, 1);
% Plot the approximation
```

```
plot(N, Q_1N, 'blue');
hold on;
% Plot the integral line
plot(N, Q_N, 'red');
% Add title
title('$Q_{1}$ and $Q$ vs $N$','Interpreter','latex');
% Add x axis label
xlabel('$N$' ,'Interpreter','latex');
% Add y axis label
ylabel('$Q_{1}$ and $Q$','Interpreter','latex');
grid on;
% Add legend
legend('$Q_{1}$', '$Q$', 'Interpreter','latex' );
% Second subplot
subplot(2, 1, 2);
% Plot the error line in a seprate subplot
plot(N, Q_1error, 'blue');
title('$Q_{1}$ Error vs $N$','Interpreter','latex');
% Add x axis label
xlabel('$N$' ,'Interpreter','latex');
% Add y axis label
ylabel('Error','Interpreter','latex');
grid on;
fig = gcf; % Obtains current graphic in matlab
exportgraphics(fig, 'Fig/q1_sum_error_plot.pdf','ContentType','vector'); %
    Exports plot as a vector pdf image. (Requires R2020a or later)
```

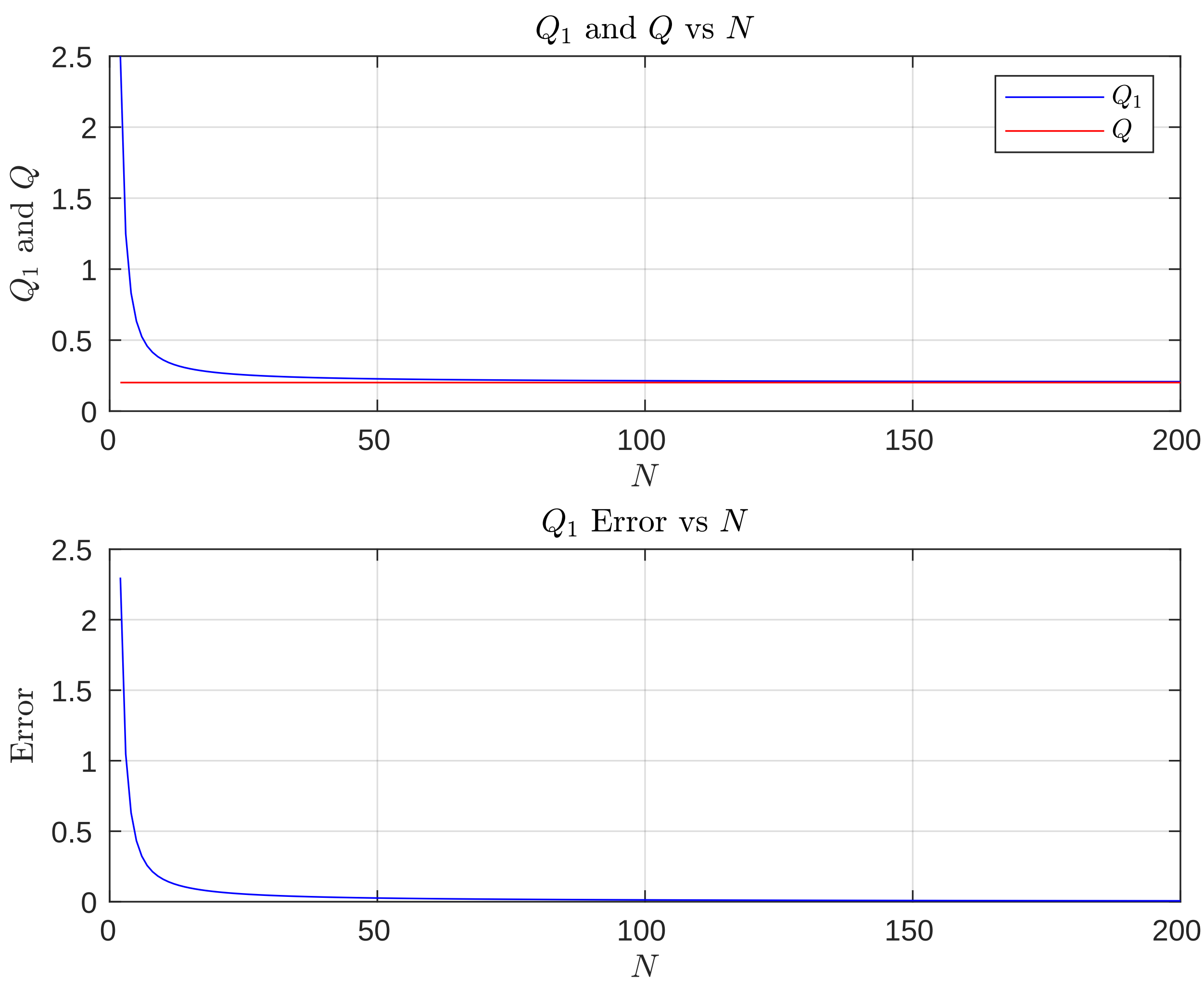

Figure 6: Q_1 and Q vs N and the % Error for Q_1

Q_1 in Figure 6 converges to Q at around $N \approx 75$

5.2.6 Q_3 and Q vs N

Listing 10: Q_3 Function

```
function Q_3 = q3_sum(N)
dx = (5-0)/(N-1);
x = 0:dx:5;
y = fx(x);
A = [0.5; ones(N-2, 1); 0.5];
Q_3 = y*A*dx;
end
```

Listing 11: Plot Q_3 and Q vs N

```
clear;
clc;
% N starts at 2 and ends at 200
N  = 2:1:200;
% Domain for Q1
Q_3N = zeros(1, length(N));
% Populate each ith value with the furutre q1 N calculation
for i = N
    Q_3N(i-1) = q2_sum(i);
end
% Create the domain for for the integral calcualtion
Q_N = ones(1, length(N));
% Find the integral using quad
Q = quad(@fx, 0, 5); % Function, lower bound, upper bound
% Convert the integral value from a scalar value to line of a constant
% value
Q_N = Q_N*Q;
% Create the domain for for error calculation
Q_3error = zeros(1, length(N));
% Subtract the summation from the actual value to calculate the error
Q_3error = Q_3N - Q_N;
% Declare the first figure
figure(1);
% Declare the subplot
% First subplot
subplot(2, 1, 1);
% Plot the approximation
```

```matlab
plot(N, Q_3N, 'blue');
hold on;
% Plot the integral line
plot(N, Q_N, 'red');
% Add title
title('$Q_{3}$ and $Q$ vs $N$','Interpreter','latex');
% Add x axis label
xlabel('$N$' ,'Interpreter','latex');
% Add y axis label
ylabel('$Q_{3}$ and $Q$','Interpreter','latex');
% Add legend
legend('$Q_{3}$', '$Q$', 'Interpreter','latex' );
grid on;
% Second subplot
subplot(2, 1, 2);
% Plot the error line in a seprate subplot
plot(N, Q_3error, 'blue');
title('$Q_{3}$ Error vs $N$','Interpreter','latex');
% Add x axis label
xlabel('$N$' ,'Interpreter','latex');
% Add y axis label
ylabel('Error','Interpreter','latex');
grid on;
fig = gcf; % Obtains current graphic in matlab
exportgraphics(fig, 'Fig/q3_sum_error_plot.pdf','ContentType','vector'); %
    Exports plot as a vector pdf image. (Requires R2020a or later)

%Plot Q1 and Q3 Error
% Declare the second figure
figure(2);
% Domain for Q1
Q_1N = zeros(1, length(N));
% Populate each ith value with the future q1 N calculation
for i = N
    Q_1N(i-1) = q1_sum(i);
end
% Create the domain for for the integral calculation
Q_N = ones(1, length(N));
% Find the integral using quad
Q = quad(@fx, 0, 5); % Function, lower bound, upper bound
```

```
% Create a vector of size N with the same Q value for each
% value
Q_N = Q_N*Q;
% Create the domain for for error calculation
Q_1error = zeros(1, length(N));
% Subtract the summation from the actual value to calculate the error
Q_1error = Q_1N - Q_N;
% Plot the lines
plot(N, Q_1error, 'red', N, Q_3error, 'blue');
% Add title
title('$Q_{1}$ and $Q_{3}$','Interpreter','latex');
% Add x axis label
xlabel('$N$' ,'Interpreter','latex');
% Add y axis label
ylabel('$Q_{1}$ and $Q_{3}$','Interpreter','latex');
% Add legend
legend('$Q_{1}$', '$Q_{3}$', 'Interpreter','latex' );
grid on;
fig = gcf; % Obtains current graphic in matlab
exportgraphics(fig, 'Fig/q1_q3_error_plot.pdf','ContentType','vector'); % Exports plot as a vector pdf image. (Requires R2020a or later)
```

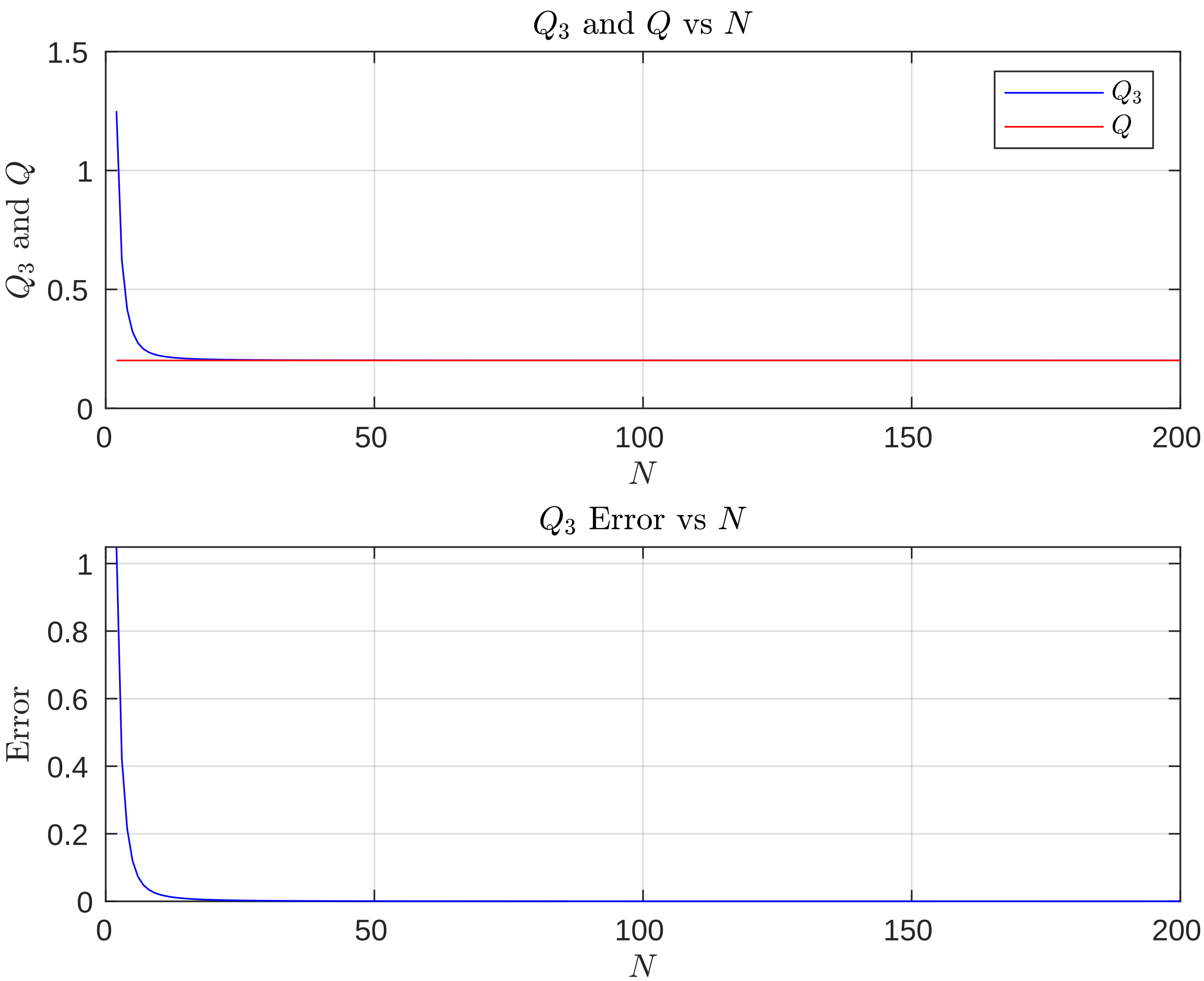

Figure 7: Q_3 and Q vs N and the % Error for Q_2

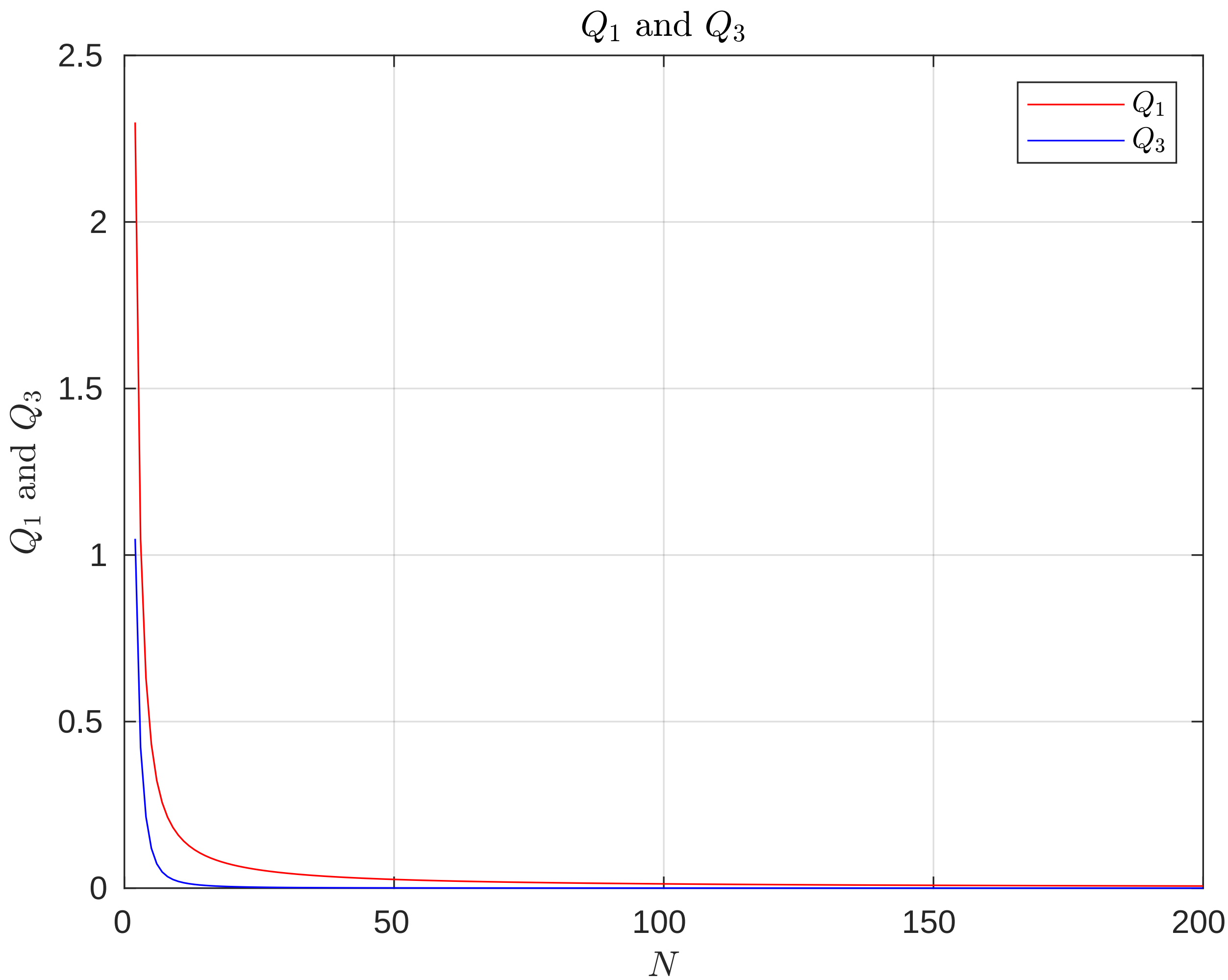

Figure 8: Error Comparison of Q_1 and Q_3

Q_3 in Figure 7 converges to Q at around $N = 25$. Q_3 converges to Q at a faster rate compared to Q_1 as shown in Figure 8.

6 Conclusion

This laboratory exercise showcases MATLAB's versatility across diverse scientific and engineering domains. It notably highlights the fundamental concept of trade-offs in engineering. Riemann summation serves as a poignant example, empowering students to grapple with the delicate balance between accuracy and computational efficiency: a fundamental skill for effective engineering design.