Zack Humphries

Homework 4

COMP 521
October 14, 2022

Problem

Approximate the integral of $f(x) = e^{-2x}sin(2\pi x)$ on the interval $x \in [0, 3.5]$ using:

1. The composite trapezoidal rule
2. The composite Simpson's rule
3. Adaptive quadrature

You have to use the Matlab code provided with this document.

For the composite rules, use the following number of integration subintervals $N = \{20, 40, 80, 160\}$. Use these solutions to determine the orders of accuracy. The exact integral is 0.1446445197. Use this value to calculate the absolute errors.

Compare the results from the quadratures. Use a table to show the results.

Discuss your results.

Deliverable: Submit a .PDF file with your report. Submit the main.m code file only with your modifications.

A table comparing the different approximation functions is shown below:

Subinterval	Actual	Trapeziodal: Estimated	Trapezoidal: Absolute Error	Simpsons: Estimated	Simpsons: Absolute Error	Quadrature: Estimated	Quadrature: Absolute Error
20	0.1446	0.1284	0.0163	0.1447	5.6311e-05	0.1446	6.8181e-07
40	0.1446	0.1406	0.0040	0.1446	3.5187e-06	0.1446	6.8181e-07
80	0.1446	0.1436	0.0010	0.1446	2.1983e-07	0.1446	6.8181e-07
160	0.1446	0.1444	2.5083e-04	0.1446	1.3713e-08	0.1446	6.8181e-07

Figure 1: Comparison Table

The adaptive quadrature does not have any integration subintervals. It is more used in the table to compare to the other integration techniques.

Out of the composite trapezoidal rule and composite Simpson's rule, the composite Simpson's rule faired much better as it has a smaller error degree than the composite trapezoidal rule $\mathcal{O}(h^5)$ for the Simpson's rule versus $\mathcal{O}(h^3)$ for the trapezoidal rule.

The adaptive quadrature was generally more accurate than the two composite rules, except for the 80 and 160 integration subinterval of the composite Simpson's rule. If the tolerance for the adaptive quadrature had been set at a much lower value than 10^{-4}, then the adaptive quadrature would have a higher accuracy. However, setting a lower tolerance would be at the expense of computing time.

```
%% Zack Humphries
% COMP 521
% HW4

% Use numerical integration to approximate the integral of a function f(x)
% over the interval x \in [a,b]
close all; clear all; clc;

a = 0;
b = 3.5;
n_list = [20 40 80 160];
length_n_list = length(n_list);
trapl_array = zeros(1,length_n_list);
simprl_array = zeros(1,length_n_list);

for n_index=1:length_n_list

    n = n_list(n_index);
    % Part 1
    % Apply the somposite trapezoidal rule to calculate the integral
    Itc = traprl(@Fx, a, b, n);
    fprintf("The result for the integral with n = %f using Trapezoidal is %.16
    f \n", n, Itc);


    % Part 2
    % Apply the somposite Simpson's Rule to calculate the integral
    Isc = simprl(@Fx, a, b, n);
    fprintf("The result for the integral with n = %f using Simpson's is %.16f
    \n", n, Isc);


    trapl_array(n_index) = Itc;
    simprl_array(n_index) = Isc;
end

% Part 3
% Apply adadptive quadrature to calculate the integral
tol = 1e-4;
[SRmat, Ias, err] = adapt(@Fx, a, b, tol);
x = SRmat(2,:);
fprintf("The result for the integral using Adaptive Simpson's is %.16f \n",
    Ias);

actual = 0.1446445197;

trapl_error = abs(actual - trapl_array);
simprl_error = abs(actual - simprl_array);
adapt_error = abs(actual-Ias);
actual_list = repmat(actual, 1, length_n_list);

adapt_list = repmat(Ias, 1, length_n_list);
adapt_error_list = repmat(adapt_error, 1, length_n_list);

var_names = ['Subinterval', "Actual", "Trapeziodal: Estimated", "Trapezoidal:
```

```
    Absolute Error", "Simpsons: Estimated", "Simpsons: Absolute Error", "
    Quadrature: Estimated", "Quadrature: Absolute Error"];
53
54 T = table(n_list.', actual_list.', trapl_array.', trapl_error.', simprl_array
    .', simprl_error.', adapt_list.', adapt_error_list.', VariableNames=
    var_names)
55 uitable('Data',T{:,:},'ColumnName',T.Properties.VariableNames,...
56     'RowName',T.Properties.RowNames,'Units', 'Normalized', 'Position',[0, 0,
    1, 1]);
57
58
59 function result = Fx(x)
60     part1 = exp(-2*x);
61     part2 = sin(2*pi*x);
62     result = part1.*part2;
63 end
```

Homework 4: main.m