AE370: Homework 5

C. Latham

February 19, 2018

Time to Complete: 7 hours

1 Problem 1

1.1 Parts A, B, C, D

Using code from various previous homeworks, error plots for the equispaced Lagrange, Chebyshev Lagrange, and composite trapezoid rule approximations were found. All three of these were plotted together on a semi-log plot. The plot is seen in 1. The code is seen in 1.2, 1.3, 1.4, 1.5, and 1.6.

There are a few reasons that composite trapezoid integration is the best form of integration. The uniform Lagrange function takes a long time (0.8702 seconds), and the error actually increased as n grew. This was likely due to erratic endpoint behavior that has been observed previously using equispaced points. The Chebyshev Lagrange function took even longer (2.3800 seconds), but the error is respectable, following what looked like a Δx trend compared to n. The composite trapezoid integration method is incredibly fast (0.002387 seconds) and follows a Δx^2 trend compared to n. Time measurement was done using tic-toc and the error is seen below in 1.

	$n = 2$	$n = 8$	$n = 16$	$n = 20$
Uniform Lagrange	4.0481	1.2463	8.9882	29.5964
Chebyshev Lagrange	4.0481	0.2760	0.0110	0.0022
Composite Trapezoid	2.4455	0.0377	6.8992e-04	5.9337e-04

Table 1: Errors for the different integration schemes

1.2 General Code

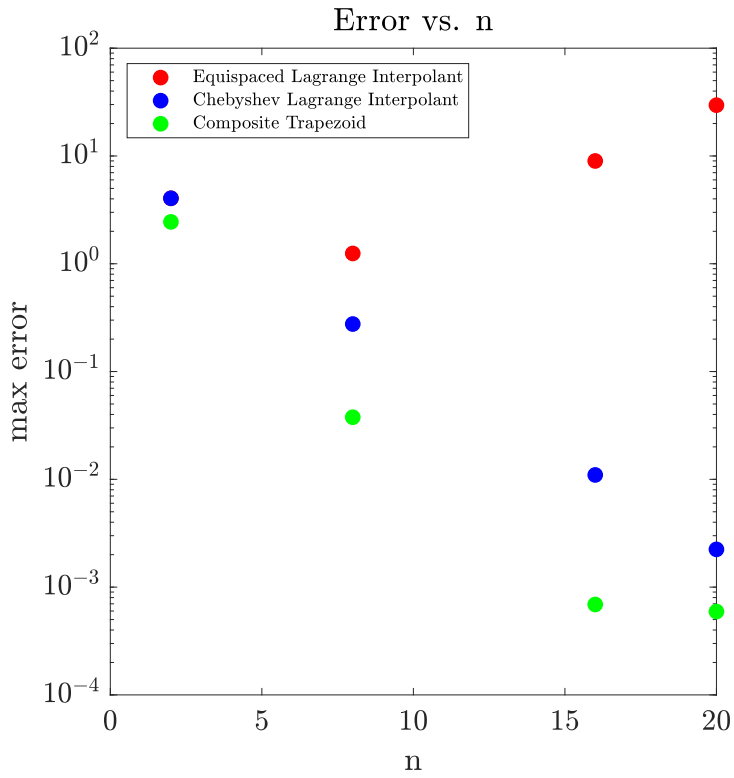


Figure 1: Error for parts A, B, and C plotted against the number of points, "n"

```
%% Problem 1
clear all, close all, clc

% SETUP

% declare the different n values to eval at
nvect = [2 8 16 20];
% define the function to appx.
fcn = @(x) 1 ./ (1 + x.^2);
% set up the endpoints of the range
endp = [-5,5];

% EXACT INTEGRAL VALUE

% declare a sym
syms x
% eval the integral
int_exact = double(int(fcn(x), x, endp(1), endp(2)))
% clear the symbolic
clear x

```

```
% ERROR STRUCT INITIATION

% equispaced lagrange
err.eqLag = [];
% chebyshev lagrange
err.chebLag = [];
% equispaced composite trapezoid
err.comTrap = [];
```

1.3 Part A Code

```
%% Part A

% SETUP

% create a symbolic x
syms x
% set up a for loop to go through nvect
for i = 1:length(nvect)
    % assign a variable the specific number of n
    ni = nvect(i);

    % INTERPOLATION

    % create the interpolation points
    intPoints = interp_Maker(endp,ni,'eq');
    % evaluate the given function at the interpolation points to get coeffs
    intVals = fcn(intPoints);
    % create the lagrangian function
    func = lag_Func(intPoints,intVals);

    % EVALUATION

    % evaluate the integral with the approximating function
    int_appx = double(int(func(x),x,endp(1),endp(2)));
    % find the error between the appx and the exact answer
    err.eqLag(i) = norm(int_appx-int_exact);
end
disp("Equispaced Lagrange = " + num2str(err.eqLag))
% clear the symbolic just in case
clear x i func int_appx intPoints intVals ni
```

1.4 Part B Code

```
%% Part B

% SETUP

% declare a symbolic x for integrating
syms x
% set up a for loop to go through nvect
for i = 1:length(nvect)
    % assign a variable the specific number of n
    ni = nvect(i);

    % INTERPOLATION

    % create the interpolation points
    intPoints = interp_Maker(endp,ni,'ch');
    % evaluate the given function at the interpolation points to get coeffs
    intVals = fcn(intPoints);
    % create the lagrangian function
    func = lag_Func(intPoints,intVals);

    % EVALUATION

    % evaluate the integral with the approximating function
    int_appx = double(int(func(x),x,endp(1),endp(2)));
    % find the error between the appx and the exact answer
    err.chebLag(i) = norm(int_appx-int_exact);
end
disp("Chebspaced Lagrange = " + num2str(err.chebLag))
% clear the symbolic just in case
clear x i func int_appx intPoints intVals ni
```

1.5 Part C Code

```
%% Part C

% SETUP

% for loop to iterate through nvect
for i = 1:length(nvect)
    % assign a variable the n
    ni = nvect(i);
    % create the uniformly spaced points
    intpoints = interp_Maker(endp,ni,'eq');
    % find delta, should be equal everywhere so use first two points
    delta = intpoints(2)-intpoints(1);
```

```
    % assign the first value to comp trap, f(x0)
    comptrap = fcn(intpoints(1));
    % iterate and add to comp trap, the sigma from j=1 to n-1 of f(xj)
    for j = 2:length(intpoints)-1
        comptrap = comptrap + 2*fcn(intpoints(j));
    end
    % add the last part f(xn) and multiply through the .5 and delta
    int_appx = .5*delta*(comptrap + fcn(intpoints(length(intpoints))));
    % assign the error
    err.comTrap(i) = norm(int_appx-int_exact);
end
% display the final errors found
disp("Uniform Composite Trapezoid = " + num2str(err.comTrap));
```

1.6 Plotting Code

```
%% Plotting A, B, C
semilogy(nvect,err.eqLag,'r.',...
    nvect,err.chebLag,'b.',...
    nvect,err.comTrap,'g.',...
    'markersize',26);
h = legend('Equispaced Lagrange Interpolant',...
    'Chebyshev Lagrange Interpolant',...
    'Composite Trapezoid');
set( h, 'location', 'NorthWest', 'interpreter', 'latex', 'fontsize', 10)
xlabel( 'n', 'interpreter', 'latex', 'fontsize', 12)
ylabel( 'max error', 'interpreter', 'latex', 'fontsize', 12)
title( 'Error vs. n', 'interpreter', 'latex', 'fontsize', 12);
set(gca, 'TickLabelInterpreter','latex', 'fontsize', 16 )
set(gcf, 'PaperPositionMode', 'manual')
set(gcf, 'Color', [1 1 1])
set(gca, 'Color', [1 1 1])
set(gcf, 'PaperUnits', 'centimeters')
set(gcf, 'PaperSize', [15 15])
set(gcf, 'Units', 'centimeters' )
set(gcf, 'Position', [0 0 15 15])
set(gcf, 'PaperPosition', [0 0 15 15])
```

2 Problem 2

See attached PDF pages.

3 Problem 3

3.1 Part B

Problem 3 asked to approximate the error for the integral

$$\int_0^2 \sin(10\pi x)dx$$

for $n = 4, 16, 24, 48$ subsections. This error was plotted in 2. As can be seen, the error was extremely small, close to machine epsilon (2.2204e-16). There seemed to be a small increase in error for $n = 24$, however this is nearly negligible and is not considered to be an issue with the method of approximation, but an artifact of how small the approximation is. This is quite remarkable indeed! The code is seen in 3.2, 3.3, 3.4.

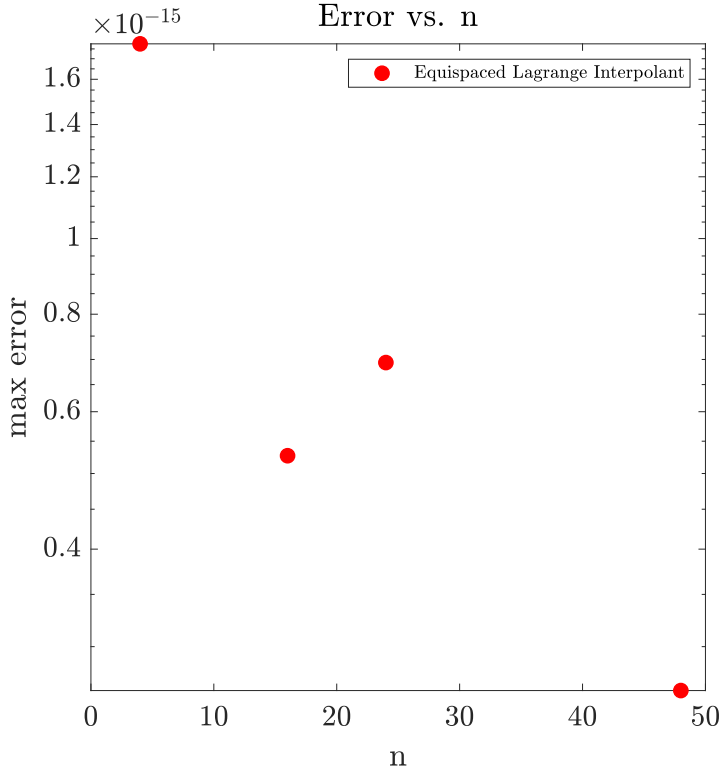


Figure 2: Caption

3.2 General Code

```
%% Problem 3
clear all, close all, clc

```

```
% SETUP

% declare the different n values to eval at
nvect = [4 16 24 48];
% define the function to appx.
fcn = @(x) sin(10*pi*x);
% set up the endpoints of the range
endp = [0,2];

% EXACT INTEGRAL VALUE

% declare a sym
syms x
% eval the integral
int_exact = double(int(fcn(x), x, endp(1), endp(2)))
% clear the symbolic
clear x

% ERROR STRUCT INITIATION

% equispaced composite trapezoid
err = [];
```

3.3 Part A Code

```
%% Part A

% SETUP
tic
% for loop to iterate through nvect
for i = 1:length(nvect)
    % assign a variable the n
    ni = nvect(i);
    % create the uniformly spaced points
    intpoints = interp_Maker(endp,ni,'eq');
    % find delta, should be equal everywhere so use first two points
    delta = intpoints(2)-intpoints(1);
    % assign the first value to comp trap, f(x0)
    comptrap = fcn(intpoints(1));
    % iterate and add to comp trap, the sigma from j=1 to n-1 of f(xj)
    for j = 2:length(intpoints)-1
        comptrap = comptrap + 2*fcn(intpoints(j));
    end
    % add the last part f(xn) and multiply through the .5 and delta
    int_appx = .5*delta*(comptrap + fcn(intpoints(length(intpoints))));
    % assign the error
```

```
    err(i) = norm(int_appx-int_exact);
end
toc
% display the final errors found
disp("Uniform Composite Trapezoid = " + num2str(err));
```

3.4 Plotting Code

```
%% Plotting A
semilogy(nvect,err,'r.',...
    'markersize',26);
h = legend('Equispaced Lagrange Interpolant',...
    'Chebyshev Lagrange Interpolant',...
    'Composite Trapezoid');
set( h, 'location', 'NorthEast', 'interpreter', 'latex', 'fontsize', 10)
xlabel( 'n', 'interpreter', 'latex', 'fontsize', 12)
ylabel( 'max error', 'interpreter', 'latex', 'fontsize', 12)
title( 'Error vs. n', 'interpreter', 'latex', 'fontsize', 12);
set(gca, 'TickLabelInterpreter','latex', 'fontsize', 16 )
set(gcf, 'PaperPositionMode', 'manual')
set(gcf, 'Color', [1 1 1])
set(gca, 'Color', [1 1 1])
set(gcf, 'PaperUnits', 'centimeters')
set(gcf, 'PaperSize', [15 15])
set(gcf, 'Units', 'centimeters' )
set(gcf, 'Position', [0 0 15 15])
set(gcf, 'PaperPosition', [0 0 15 15])
```

4 Problem 4

4.1 Part C

Problem 4 asked to approximate the integral

$$\int_0^2 x^2 \sin(10x) dx$$

using the composite trapezoidal rule for $n = 4, 16, 24, 48$. The specific Richardson-Extrapolation used is

$$E_{RE} = \frac{4}{3}E(2n) - \frac{1}{3}E(n)$$

where $E(n)$ is the error between the exact integral and the approximate integral found using the composite trapezoid method for n subsections. The idea behind this is that the

overestimate of the $E(2n)$ component is brought closer to the actual value of the integral by subtracting the $E(n)$ component. The plot of the error can be seen in 3 3 shows how

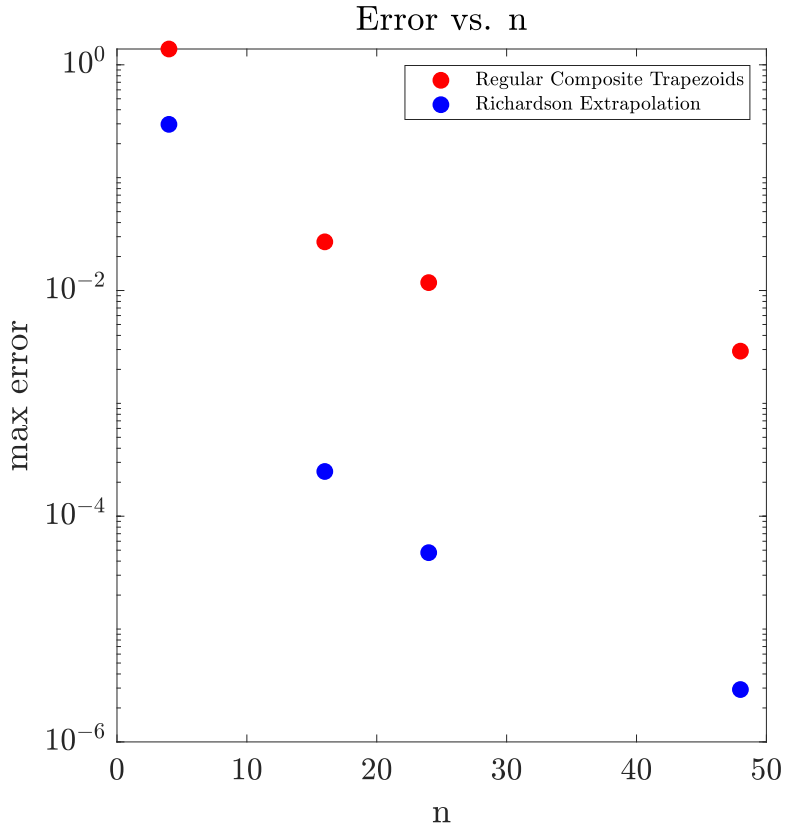


Figure 3: Regular Comp. Trap. Appx. versus Richardson Extrapolation

much better the Richardson Extrapolation is at approaching the exact integral, with error nearly 4 orders of magnitude lower than regular composite trapezoidal approximation. Code is shown in 4.2, 4.3, 4.4, and 4.5.

4.2 General Code

```
%% Problem 4
clear all, close all, clc

% SETUP

% declare the different n values to eval at
nvect = [4 16 24 48];
% define the function to appx.
fcn = @(x) x.^2.*sin(10.*x);
% set up the endpoints of the range
endp = [0,2];
```

```

% EXACT INTEGRAL VALUE

% declare a sym
syms x
% eval the integral
int_exact = double(int(fcn(x), x, endp(1), endp(2)))
% clear the symbolic
clear x

% ERROR STRUCT INITIATION

% equispaced composite trapezoid
err.compTrap1 = [];
% equispaced for delta/2
err.compTrap2 = [];
% richardson extrapolation
err.RichExtrap = [];
```

4.3 Part A Code

```
%% Part A

% SETUP

% for loop to iterate through nvect
tic
for i = 1:length(nvect)
    % assign a variable the n
    ni = nvect(i);
    % create the uniformly spaced points
    intpoints = interp_Maker(endp,ni,'eq');
    % find delta, should be equal everywhere so use first two points
    delta = intpoints(2)-intpoints(1);
    % assign the first value to comp trap, f(x0)
    comptrap = fcn(intpoints(1));
    % iterate and add to comp trap, the sigma from j=1 to n-1 of f(xj)
    for j = 2:length(intpoints)-1
        comptrap = comptrap + 2*fcn(intpoints(j));
    end
    % add the last part f(xn) and multiply through the .5 and delta
    int_appx = .5*delta*(comptrap + fcn(intpoints(length(intpoints))));
    % assign the error
    err.compTrap1(i) = norm(int_appx-int_exact);
end
toc
```

```
% display the final errors found
disp("Uniform Composite Trapezoid = " + num2str(err.compTrap1));
```

4.4 Part B Code

```
%% Part B

% SETUP
tic
% for loop to iterate through nvect
for i = 1:length(nvect)
    % assign a variable the n
    ni = 2*nvect(i);
    % create the uniformly spaced points
    intpoints = interp_Maker(endp,ni,'eq');
    % find delta, should be equal everywhere so use first two points
    delta = (intpoints(2)-intpoints(1));
    % assign the first value to comp trap, f(x0)
    comptrap = fcn(intpoints(1));
    % iterate and add to comp trap, the sigma from j=1 to n-1 of f(xj)
    for j = 2:length(intpoints)-1
        comptrap = comptrap + 2*fcn(intpoints(j));
    end
    % add the last part f(xn) and multiply through the .5 and delta
    int_appx = .5*delta*(comptrap + fcn(intpoints(length(intpoints))));
    % assign the error
    err.compTrap2(i) = abs(int_appx-int_exact);
end
toc
err.RichExtrap = abs((4/3)*err.compTrap2-(1/3)*err.compTrap1);
% display the final errors found
disp("Richardson Extrapolation = " + num2str(err.RichExtrap));
```

4.5 Plotting Code

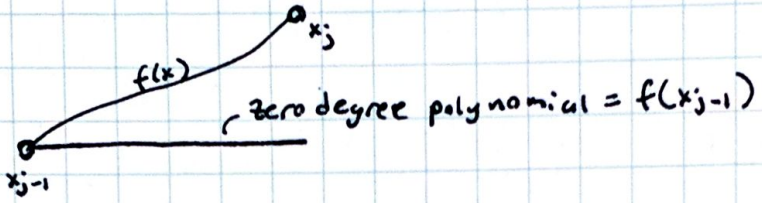
```
%% Plotting A, B
semilogy(nvect,err.compTrap1,'r.',...
    nvect,err.RichExtrap,'b.',...
    'markersize',26);
h = legend('Regular Composite Trapezoids',...
    'Richardson Extrapolation');
set( h, 'location', 'NorthEast', 'interpreter', 'latex', 'fontsize', 10)
xlabel( 'n', 'interpreter', 'latex', 'fontsize', 12)
ylabel( 'max error', 'interpreter', 'latex', 'fontsize', 12)
```

```
title( 'Error vs. n', 'interpreter', 'latex', 'fontsize', 12);
set(gca, 'TickLabelInterpreter','latex', 'fontsize', 16 )
set(gcf, 'PaperPositionMode', 'manual')
set(gcf, 'Color', [1 1 1])
set(gca, 'Color', [1 1 1])
set(gcf, 'PaperUnits', 'centimeters')
set(gcf, 'PaperSize', [15 15])
set(gcf, 'Units', 'centimeters' )
set(gcf, 'Position', [0 0 15 15])
set(gcf, 'PaperPosition', [0 0 15 15])
```

2

a) $$\int_a^b f(x)\,dx = \sum_{j=1}^{n} \int_{x_{j-1}}^{x_j} f(x)\,dx$$

b) A zero degree polynomial is a horizontal line: $x^0 = 1$. See this diagram:



Across the interval $[x_{j-1}, x_j]$, $f(x) \approx f(x_{j-1})$

c) $$\int_a^b f(x)\,dx \approx \sum_{j=1}^{n} \int_{x_{j-1}}^{x_j} f(x_{j-1})\,dx = \sum_{j=1}^{n} \overbrace{(x_j - x_{j-1})}^{\Delta x} f(x_{j-1}) = \sum_{j=1}^{n} \Delta x\, f(x_{j-1})$$

d) $$\text{error} = \left| \sum_{j=1}^{n} \left[\int_{x_{j-1}}^{x_j} f(x)\,dx - \int_{x_{j-1}}^{x_j} f(x_{j-1})\,dx \right] \right|$$

Taylor expansion: $A = T_{exp}(f(x_{j-1})) = f(x) + f'(x)(x_{j-1} - x) + HOT$

$$\text{error} = \left| \sum_{j=1}^{n} \left[\int_{x_{j-1}}^{x_j} f(x)\,dx - \int_{x_{j-1}}^{x_j} (f(x) + f'(x)(x_{j-1} - x) + HOT)\,dx \right] \right|$$

Pull HOT out to use Δ ineq.

$$\text{error} = \left| \sum_{j=1}^{n} \left[- \int_{x_{j-1}}^{x_j} (f'(x)(x_{j-1} - x) + HOT)\,dx \right] \right|$$

Triangle Inequality:

$$\left| \sum_{j=1}^{n} \left[- \int_{x_{j-1}}^{x_j} f'(x)(x_{j-1} - x)\,dx + HOT \right] \right| \le \sum_{j=1}^{n} \left| \overbrace{- \int_{x_{j-1}}^{x_j} f'(x)(x_j - x)\,dx}^{C} \right| + HOT$$

So:

$$C \le \int_{x_{j-1}}^{x_j} |f'(x)|\,|(x_{j-1} - x)|\,dx$$ new "C"

$$C \le \max_{x \in [a,b]} |f'(x)|\,|\overbrace{(x_{j-1} - x_j)}^{\Delta x}| \int_{x_{j-1}}^{x_j} dx = \max_{x \in [a,b]} |f'(x)|\,\Delta x^2, \quad n = \frac{(b-a)}{\Delta x}$$

replaces the $\sum$

$$\boxed{\text{so error} \le \left[\max_{x \in [a,b]} |f'(x)| (b-a) \Delta x \right]}$$