Math 4540/MSSC 5540 - Homework #2

Henri Medeiros Dos Reis

October 9, 2023

Directions. All work is to be done in complete sentences.. Assignments must be stapled. Each problem (1.,2., ... not i),ii))...) must begin on a separate sheet of paper. Please use the back. Please begin each problem with the full statement of the problem. While you are encouraged to work through confusion with your classmates, your work must be written in your own words. The assignment is due by 5:00 on Wednesday, October 11, 2024 in Dr. Clough's mailbox outside CU 340 in the narrow dark hallway.

1. i) 1.5.1b). Apply two steps of the Secant Method to the following equations with initial guesses $x_0 = 1$, and $x_1 = 2$. (b) $e^x + x = 7$

$$f(x) = e^x + x - 7 = 0$$
$$i = 1, x_0 = 1, x_1 = 2$$

$$\begin{aligned} x_2 &= x_0 - \frac{f(x_1)(x_1 - x_0)}{f(x_1) - f(x_0)} \\ &= 2 - \frac{(e^2 + 2 - 7)(2 - 1)}{(e^2 + 2 - 7) - (e^1 + 1 - 7)} \\ &\approx 1.5787 \end{aligned}$$

$$i = 2, x_1 = 2, x_2 = 1.5787$$

$$\begin{aligned} x_3 &= 1.5787 - \frac{f(x_2)(x_2 - x_1)}{f(x_2) - f(x_1)} \\ &= 1.5787 - \frac{(e^{1.5787} + 1.5787 - 7)(1.5787 - 2)}{(e^{1.5787} + 1.5787 - 7) - (e^2 + 2 - 7)} \\ &\approx 1.6602 \end{aligned}$$

ii) 1.5.2b) Apply two steps of the Method of False Position with initial bracket $[1, 2]$ to the equation of Exercise 1.

$$f(x) = e^x + x - 7 = 0$$
$$i = 1, a = 1, b = 2$$

$$\begin{aligned} c &= \frac{bf(a) - af(b)}{f(a) - f(b)} \\ &= \frac{2(e^1 + 1 - 7) - 1(e^2 + 2 - 7)}{(e^1 + 1 - 7) - (e^2 + 2 - 7)} \\ &\approx 1.5787 \\ \Rightarrow f(a)f(c) &> 0 \\ \Rightarrow a &= 1.5787 \end{aligned}$$

$$i = 1, a = 1.5787, b = 2$$

$$\begin{aligned} c &= \frac{bf(a) - af(b)}{f(a) - f(b)} \\ &= \frac{2(e^{1.5787} + 1.5787 - 7) - 1.5787(e^2 + 2 - 7)}{(e^{1.5787} + 1.5787 - 7) - (e^2 + 2 - 7)} \\ &\approx 1.6602 \\ \Rightarrow f(a)f(c) &> 0 \\ \Rightarrow a &= 1.6602 \end{aligned}$$

iii) Repeat the above except with Newton's method.

$$f(x) = e^x + x - 7 = 0, f'(x) = e^x + 1$$
$$i = 0, x_0 = 1$$

$$\begin{aligned} x_{i+1} &= x_i - \frac{f(x_i)}{f'(x_i)} \\ &= 1 - (e^1 + 1 - 7)e^1 + 1 \\ &= 1.8826 \end{aligned}$$

$$i = 1, x_1 = 1.8826$$

$$
\begin{aligned}
x_{i+1} &= x_i - \frac{f(x_i)}{f'(x_i)} \\
&= 1.8826 - \frac{(e^{1.8826} + 1.8826 - 7)}{e^{1.8826} + 1} \\
&= 1.6907
\end{aligned}
$$

iv) Computer Exercise 1.5.1b). Use the Secant Method to find the (single) solution to the equation in Exercise 1.

```
clear all;
close all;

f = @(x) exp(x)+x-7;
x(1) = 1;
x(2) = 2;

tol = 1e-8;
deltax = inf;
i = 2;
while deltax > tol
    x(i+1) = x(i)-(f(x(i))*(x(i)-x(i-1)))/(f(x(i))-f(x(i
       -1)));
    deltax = abs(x(i+1)-x(i));
    i = i+1;
end

fprintf(['Algorithm converged after %d ' ...
    'iterations, and the root is %.4f. ' ...
    '\n f(root) = %.8f\n'] ,i,x(i),f(x(i)))

% Output:
% Algorithm converged after 8 iterations, and the root
   is 1.6728.
% f(root) = 0.00000000
```

v) Computer Exercise 1.5.2b) Use Method of False Position to find the solution of each equation in Exercise 1.

```
clear all;
close all;

f = @(x) exp(x)+x-7;
```

```
a = 1;
b = 2;
x(1) = b;
tol = 1e-8;
deltax = inf;
i = 1;
while deltax > tol
    c = (b*f(a)-a*f(b))/(f(a)-f(b));
    if c==0
        x(i+1) = c;
        break;
    end
    if f(a)*f(c) < 0
        b = c;
        x(i+1) = b;
    else
        a = c;
        x(i+1) = a;
    end
    deltax = abs(x(i+1)-x(i));
    i = i+1;
end

fprintf(['Algorithm converged after %d ' ...
    'iterations, and the root is %.4f. ' ...
    '\n f(root) = %.8f\n'] ,i,x(i),f(x(i)))

% Output:
% Algorithm converged after 11 iterations, and the root
   is 1.6728.
% f(root) = -0.00000001
```

vi) Repeat above with Newton's method.

```
clear all;
close all;

f = @(x) exp(x)+x-7;
df = @(x) exp(x)+1;

x(1) = 1;
```

```
tol = 1e-8;
deltax = inf;
i = 1;
while deltax > tol
    x(i+1) = x(i)-f(x(i))/df(x(i));
    deltax = abs(x(i+1)-x(i));
    i = i+1;
end

fprintf(['Algorithm converged after %d ' ...
    'iterations, and the root is %.4f. ' ...
    '\n f(root) = %.8f\n'] ,i,x(i),f(x(i)))

% Output:
% Algorithm converged after 6 iterations, and the root
   is 1.6728.
% f(root) = -0.00000000
```

2. Consider the following four methods for calculating $2^{1/4}$, the fourth root of 2.

   (a) Rank them for speed of convergence, from fastest to slowest. Be sure to give reasons for your ranking.

   (A) Bisection Method applied to $f(x) = x^4 - 2$

   (B) Secant Method applied to $f(x) = x^4 - 2$

   (C) Fixed-Point Iteration applied to $g(x) = \frac{x}{2} + \frac{1}{x^3}$

   (D) Fixed-Point Iteration applied to $g(x) = \frac{2x}{3} + \frac{2}{3x^3}$

   The rank will be (B), (D), (A), and (C). All methods converge to *root* ≈ 1.1892. (B) is going to be the fastest, since it converges Super linearly. The other three methods all converge linearly, and are dependent on parameters. Considering the convergence to be 10^{-7} difference from iteration i and $i+1$, Bisection is going to be a little slower than (D), since it is going to have some trouble inside the small intervals, but still faster than (C), since $S = 0.5$. The difference between (C) and (D) is quite big, since the derivative of (C) at the root is $S = -0.3333$, while the derivative of (D) at the root is $S = -0.9975$.

   (b) Are there any methods that will converge faster than all above suggestions?

   Yes, Newton's Method is going to converge faster than all the previous suggestions. Since it is quadraticaly convergent.

3. i) Computer problem 1.4.11. The ideal gas law for a gas at low temperature and pressure is $PV = nRT$, where P is pressure (in atm), V is volume (in L), T is temperature (in K), n is the number of moles of gas, ad $R = 0.0820578$ is the molar gas constant. The van der Waals equation

$$\left(P + \frac{n^2 a}{V^2}\right)(V - nb) = nRT$$

covers the non ideal case where these assumptions do not hold. Use the ideal gas law to compute an initial guess, followed by Newton;s Method applied to the van der Waals equation to find the volume of one mole of oxygen at 320 K and a pressure of 15 atm. For oxygen, $a = 1.36L^2 - atm/mole^2$ and $b = 0.003183L/mole$. State your initial guess and solution with three significant digits.

In MATLAB

```
clear all;
close all;

f = @(P,V,n,a,b,R,T) P*V+n^2*a*n*b/V^2+n^2*a*V/V^2-P*n*b
   -n*R*T;
df = @(P,a,n,b,V) P-2*b*n^3*a/V^3-a*n^2/V^2;

R = 0.0820575;
T = 320;
P = 15;
n = 1;
a = 1.36;
b = 0.003183;
V(1) = n*R*T/P;
tol = 1e-7;
deltax = inf;
i = 1;
while deltax > tol
    V(i+1) = V(i)-f(P,V(i),n,a,b,R,T)/df(P,a,n,b,V(i));
    deltax = abs(V(i+1)-V(i));
    i = i+1;
end
fprintf(['Algorithm converged after %d ' ...
    'iterations. \n V(final) = %.8f ' ...
    '\n f(root) = %.8f' ...
    '\n Initial guess: %.8f\n'] ...
    ,i,V(i),f(P,V(i),n,a,b,R,T),V(1))
% Output:
%Algorithm converged after 4 iterations.
```

```
% V(final) = 1.70031988
% f(root) = 0.00000000
% Initial guess: 1.75056000
```

ii) Computer problem 1.4.12. Use the data from Computer Problem 11 to find the volume of 1 mole of benzene vapor at 700 K under a pressure of 20 atm. For benzene, $a = 18.0L^2 - atm/mole^2$ and $b = 0.1154L/mole$

```
clear all;
close all;

f = @(P,V,n,a,b,R,T) P*V+n^2*a*n*b/V^2+n^2*a*V/V^2-P*n*b
   -n*R*T;
df = @(P,a,n,b,V) P-2*b*n^3*a/V^3-a*n^2/V^2;

R = 0.0820575;
T = 700;
P = 20;
n = 1;
a = 18;
b = 0.1154;
V(1) = n*R*T/P;
tol = 1e-7;
deltax = inf;
i = 1;
while deltax > tol
    V(i+1) = V(i)-f(P,V(i),n,a,b,R,T)/df(P,a,n,b,V(i));
    deltax = abs(V(i+1)-V(i));
    i = i+1;
end
fprintf(['Algorithm converged after %d ' ...
    'iterations. \n V(final) = %.8f ' ...
    '\n f(root) = %.8f' ...
    '\n Initial guess: %.8f\n'] ...
    ,i,V(i),f(P,V(i),n,a,b,R,T),V(1))
%  Output:
% Algorithm converged after 5 iterations.
% V(final) = 2.63022340
% f(root) = 0.00000000
% Initial guess: 2.87201250
```

4. 5.1.7. Develop a formula for a two-point backward-difference formula for approximating $f'(x)$,including error term.

   Let's use the Taylor series for $f(xh)$ to obtain a derivative formula, and combine all terms afters f', to be the error term.

$$f(x-h) = f(x) + f'(x)(x-h-h) + \frac{f''(c)(-h)^2}{2}, c \in [x-h, x]$$

   Then we can solve for $f'(x)$

$$\begin{aligned} f'(x) &= \frac{f(x-h) - f(x)}{-h} - \frac{f''(c)(-h)^2}{2}, c \in [x-h, x] \\ &= \frac{f(x) - f(x-h)}{-h} - \frac{f''(c)(-h)^2}{2}, c \in [x-h, x] \end{aligned}$$

5. 5.1.11. Find a second-order formula for approximating $f'(x)$ by applying extrapolation to the two-point forward-difference formula.

   $f'(x) \approx \frac{f(x+h)-f(x)}{h}$, then we can extrapolate

$$
\begin{aligned}
F_{n+1} &= \frac{2^n F_n(h/2) - F_n(n)}{2^n - 1} \\
F_2(x) &= \frac{2\left(\frac{f(x+h/2)-f(x)}{h/2} - \frac{f(x+h)-f(x)}{h}\right)}{2-1} \\
&= \frac{2f(x+h/2) - 2f(x)}{h/2} - \frac{f(x+h) - f(x)}{h} \\
&= \frac{4f(x+h/2) - 3f(x) - f(x+h)}{h}
\end{aligned}
$$

6. 5.1.12.

(a) Compute the two-point forward-difference formula approximation to $f'(x)$ for $f(x) = 1/x$, where x and h are arbitrary

$$\begin{aligned} f'_{approx}(x) &\approx \frac{1/(x+h) - 1/x}{h} \\ &= \frac{1}{h(x+h)} - \frac{1}{hx} \\ &= \frac{hx - (xh + h^2)}{hx(xh + h^2)} \\ &= \frac{-h^2}{h^2(x^2 + xh)} \\ &= \frac{-1}{x^2 + xh} \end{aligned}$$

(b) Subtract the correct answer to get the error explicitly, and shoe that it is approximately proportional to h.

$$f'(x) = \frac{-1}{x^2}$$

$$\begin{aligned} error(x) = f'(x) - f'_{approx}(x) &= \frac{-1}{x^2} - (\frac{-1}{x^2 + xh}) \\ &= \frac{-1}{x^2} + \frac{-1}{x^2 + xh} \\ &= \frac{-(x^2 + xh) + x^2}{x^2(x^2 + xh)} \\ &= \frac{-h}{x^3 + x^2 h} \end{aligned}$$

As it is possible to see the error is proportional to h.

(c) Repeat parts (a) and (b), using the three point centered difference formula instead. Now the error should be proportional to h^2

First, let's compute the three point centered difference.

$$
\begin{aligned}
f'_{approx}(x) &\approx \frac{1/(x+h) - 1/(x-h)}{2h} \\
&= \frac{1}{2hx + 2h^2} - \frac{1}{2hx - 2h^2} \\
&= \frac{-4h^2}{(2hx + 2h^2)(2hx - 2h^2)} \\
&= \frac{-4h^2}{4h^2x^2 - 4h^4} \\
&= \frac{-1}{x^2 - h^2}
\end{aligned}
$$

Then the error

$$
\begin{aligned}
error(x) = f'(x) - f'_{approx}(x) &= \frac{-1}{x^2} + \frac{1}{x^2 - h^2} \\
&= \frac{-(x^2 - h^2) + x^2}{x^2(x^2 - h^2)} \\
&= \frac{h^2}{x^2(x^2 - h^2)}
\end{aligned}
$$

As it is possible to see the error is proportional to h^2.

7. i) 5.2.1 a) Apply the composite Trapezoid Rule with $m = 1, 2$ and 4 panels to approximate the integral. Compute the error by comparing with the exact value from calculus.

$$(a) \int_0^1 x^2 dx$$

```
clear all;
close all;

%% Composite trap
f = @(x) x.^2;
int_f = @(x) x.^3./3;
m = [1 2 4];
a = 0;
b = 1;
composite_trap(f,int_f,m,a,b)

%% Define the function so we can use it future problems
function composite_trap(f,int_f,m,a,b)
    for i = m
        h = (b-a)/i;
        x_i = a+h:h:b-h;

        intg = h/2*(f(a)+2*sum(f(x_i))+f(b));

        exact = int_f(b)-int_f(a);
        error = abs(exact-intg);
        fprintf(['Integral is approximatelly = %.6f '
           ...
            'when m = %d\n Error = %.6f\n'] ...
            ,intg,i,error);
    end
end
% Output:
%Integral is approximatelly = 0.500000 when m = 1
% Error = 0.166667
%Integral is approximatelly = 0.375000 when m = 2
% Error = 0.041667
%Integral is approximatelly = 0.343750 when m = 4
% Error = 0.010417
```

ii) 5.2.3 a) Apply the Composite Midpoint Rule with $m = 1, 2$ and 4 panels to approximate the integrals in Exercise 1, and report the errors

```
clear all;
close all;

%% Composite midpoint rule
f = @(x) x.^2;
int_f = @(x) x.^3./3;
m = [1 2 4];
a = 0;
b = 1;
composite_midpoint(f,int_f,m,a,b)

%% Define the function so we can use it future problems
function composite_midpoint(f,int_f,m,a,b)
    for i = m
        h = (b-a)/i;
        x_i = a + h/2 : h : b - h/2; % Midpoints

        intg = h * sum(f(x_i));

        exact = int_f(b) - int_f(a);
        error = abs(exact - intg);
        fprintf(['Integral is approximately = %.6f ' ...
            'when m = %d\n Error = %.6f\n'] ...
            , intg, i, error);
    end
end
% Output:
%Integral is approximately = 0.250000 when m = 1
% Error = 0.083333
%Integral is approximately = 0.312500 when m = 2
% Error = 0.020833
%Integral is approximately = 0.328125 when m = 4
% Error = 0.005208
```

8. i) Computer problem 5.2.1 b) Use the composite Trapezoid Rule with $m = 16$ and 32 panels to approximate the definite integral. Compare with the correct integral and report the two errors

$$(b) \int_0^1 \frac{x^3 dx}{x^2 + 1}$$

Using the function defined above, we can only change the values for m and the function handles, and get the desired results.

```
%% Composite trap
f = @(x) x.^3./(x.^2+1);
int_f = @(x) -(log(x.^2+1)-x.^2)./2;
m = [16 32];
a = 0;
b = 1;
composite_trap(f,int_f,m,a,b)
% Output:
%Integral is approximatelly = 0.153752 when m = 16
% Error = 0.000326
%Integral is approximatelly = 0.153508 when m = 32
% Error = 0.000081
```

ii) Computer problem 5.2.1 d)

$$(d) \int_1^3 x^2 lnx dx$$

Using the function defined in problem 7, we can only change the values for m, a, b and the function handles, and get the desired results.

```
%% Composite trap
f = @(x) x.^2.*log(x);
int_f = @(x) (x.^3.*(3.*log(x)-1))./9;
m = [16 32];
a = 1;
b = 3;
composite_trap(f,int_f,m,a,b)
% Output:
%Integral is approximatelly = 7.009809 when m = 16
% Error = 0.011188
%Integral is approximatelly = 7.001419 when m = 32
% Error = 0.002797
```

iii) Computer problem 5.2.2 b) Apply the composite Simpson's Rule to the integrals in problem 1. Use $m = 16$ and 32, and report errors.

$$(b) \int_0^1 \frac{x^3 dx}{x^2 + 1}$$

Let's also define the function for Simpson's rule so it can be used later.

```
%% Composite Simpson's
f = @(x) x.^3./(x.^2+1);
int_f = @(x) -(log(x.^2+1)-x.^2)./2;
m = [16 32];
a = 0;
b = 1;
composite_simp(f,int_f,m,a,b)

%% Define the function so we can use it future problems
function composite_simp(f,int_f,m,a,b)
    for i = m
        h = (b-a)/(2*i);
        x_i = a+h:h:b-h;

        intg = h/3*( ...
            f(a)+4*sum(f(x_i(1:2:end)))+ ...
            2*(sum(f(x_i(2:2:end-1))))+f(b));

        exact = int_f(b)-int_f(a);
        error = abs(exact-intg);
        fprintf(['Integral is approximatelly = %.6f '
           ...
            'when m = %d\n Error = %.6f\n'] ...
            ,intg,i,error);
    end
end
% Output:
%Integral is approximatelly = 0.153426 when m = 16
% Error = 0.000000
%Integral is approximatelly = 0.153426 when m = 32
% Error = 0.000000
```

iv) Computer problem 5.2.3 d) Use the composite Trapezoid Rule with $m = 16$ and 32 panels to approximate the definite integral

$$(d) \int_0^1 ln(x^2 + 1)dx$$

Using the function defined for trapezoid rule, we can only change the function handles, a, b, and get the desired results. In this case, we can ignore the error, since it was not asked in the problem.

```
%% Composite trap
f = @(x) log(x.^2+1);
int_f = @(x) x;
m = [16 32];
a = 0;
b = 1;
composite_trap(f,int_f,m,a,b)

% Output:
%Integral is approximatelly = 0.264269 when m = 16
%Integral is approximatelly = 0.264025 when m = 32
```

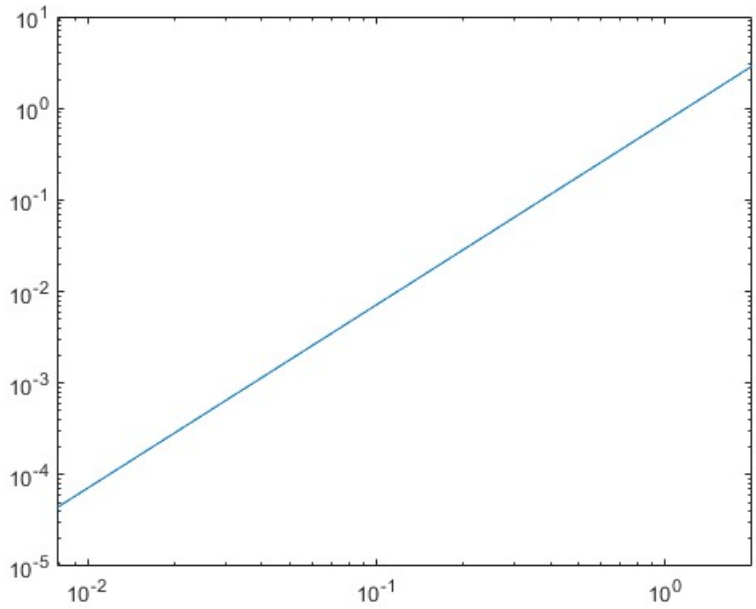
v) Computer problem 5.2.9 d) For the integrals in Computer Problem 1, calculate the approximation error of the composite Trapezoid Rule for $h = b-a, h/2, h/4, ..., h/2^8$, and plot. Make a log-log plot, using, for example, MATLAB's loglog command. What is the slope of the plot, and does it agree with theory?

$$(d) \int_1^3 x^2 lnx dx$$

For this problem, I will include the code for trapezoid rule since I made some changes to accommodate the different h values.

```
%% Composite trap
f = @(x) x.^2.*log(x);
int_f = @(x) (x.^3.*(3.*log(x)-1))./9;
m(1) = 0;
for i=1:8
    m(i+1)=2.^i;
end
a = 1;
b = 3;
m_size = size(m);
error(1) = 0;
for i = 0:m_size(2)-1
error(i+1) = composite_trap(f,int_f,m(i+1),a,b);
end
h = (b-a)./m;
h(1) = (b-a);
loglog(h,error)

slope = (log(error(8))-log(error(7)) ...
    )/(log(h(8))-log(h(7)));
```

```
fprintf('Slope = %.8f', slope)
%% Define the function so we can use it future problems
function error = composite_trap(f,int_f,m,a,b)
    for i = m
        if i==0
            h=(b-a);
        else
            h = (b-a)/i;
        end
        x_i = a+h:h:b-h;

        intg = h/2*(f(a)+2*sum(f(x_i))+f(b));

        exact = int_f(b)-int_f(a);
        error = abs(exact-intg);
    end
end
% Output:
% Slope = 2.00000273
```



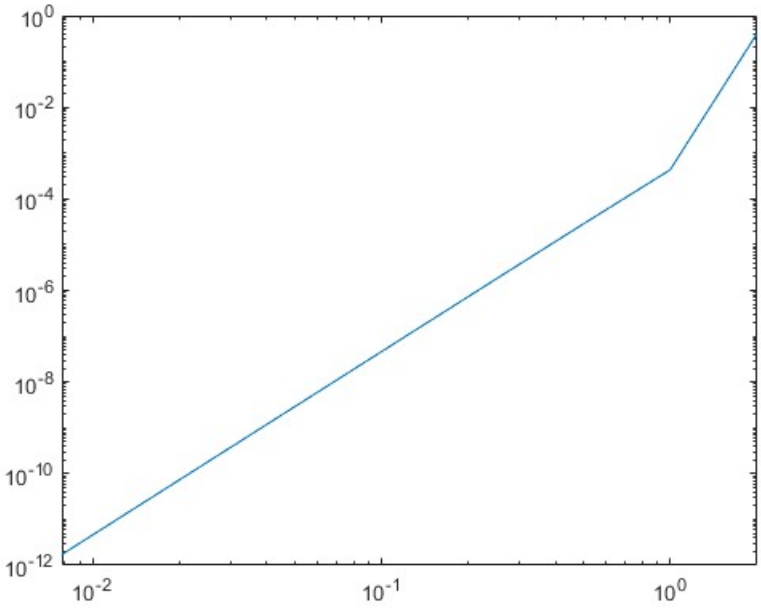
Since the theoretical error, is $\frac{-h^2}{12}(b-a)f''(c), c \in (a,b)$, the slope of the loglog plot should be 2, since it is h^2.

vi) Computer problem 5.2.10 d) Carry out Computer Problem 9, but use the composite Simpson's Rule instead of the composite Trapezoid Rule.

For this problem, I will include the code for trapezoid rule since I made some changes to accommodate the different h values.

```
%% Composite Simpsons
f = @(x) x.^2.*log(x);
int_f = @(x) (x.^3.*(3.*log(x)-1))./9;
m(1) = 0;
for i=1:8
    m(i+1)=2.^i;
end
a = 1;
b = 3;
m_size = size(m);
error(1) = 0;
for i = 0:m_size(2)-1
error(i+1) = composite_simp(f,int_f,m(i+1),a,b);
end
h = (b-a)./m;
h(1) = (b-a);
loglog(h,error)

slope = (log(error(8))-log(error(7)) ...
    )/(log(h(8))-log(h(7)));
fprintf('Slope = %.8f', slope)

%% Define the function so we can use it future problems
function error = composite_simp(f,int_f,m,a,b)
    for i = m
        if i==0
            h=(b-a);
        else
            h = (b-a)/(2*i);
        end
        x_i = a+h:h:b-h;

        intg = h/3*( ...
            f(a)+4*sum(f(x_i(1:2:end)))+ ...
            2*(sum(f(x_i(2:2:end-1))))+f(b));

        exact = int_f(b)-int_f(a);
        error = abs(exact-intg);
    end
end
```

```
% Output:
% Slope = 4.00000580
```



Since the theoretical error, is $\frac{-h^4}{90}(b-a)f^{iv}(c), c \in (a,b)$, the slope of the loglog plot should be 4, since it is h^4.

9. 5.3.3. Show that the extrapolation of the composite Trapezoid Rules in R_{11} and R_{21} yields the composite Simpson's Rule (with step size h_2) in R_{22}.

The extrapolation of R_{11}, R_{21} and R_{22} are given by:

$$\begin{aligned}
R_{11} &= \frac{h_1}{2}(f(a)+f(b)) \\
R_{21} &= \frac{h_2}{2}\left(f(a)+f(b)+2f(\frac{a+b}{2})\right) \\
&= \frac{1}{2}R_{11}+h_2 f(\frac{a+b}{2}) \\
R_{22} &= \frac{4\left(\frac{1}{2}R_{11}+h_2 f(\frac{a+b}{2})\right)-R_{11}}{3} \\
&= \frac{\left(2R_{11}+4h_2 f(\frac{a+b}{2})\right)-R_{11}}{3} \\
&= \frac{1}{3}\left(R_{11}+4h_2 f(\frac{a+b}{2})\right) \\
&= \frac{1}{3}\left(\frac{h_1}{2}(f(a)+f(b))+4h_2 f(\frac{a+b}{2})\right) \\
&= \frac{h_2}{3}\left(f(a)+f(b)+4f(\frac{a+b}{2})\right), \text{ since } h_2=\frac{h_1}{2}
\end{aligned}$$

While Simpson's Rule is given by

$$\int_a^b \approx \frac{h}{3}\left(f(a)+f(b)+4f(\frac{a+b}{2})\right)$$

setting the step size to be h_2 gives

$$\int_a^b \approx \frac{h_2}{3}\left(f(a)+f(b)+4f(\frac{a+b}{2})\right)$$

10. Computer problem 5.3.1 d) Use Romberg Integration approximationR_{55} to approximate the definite integral. Compare with the correct integral, and report the error.

$$(d) \int_1^3 x^2 lnx dx$$

Lets define a function romberg, so we can use it in future problems.

```
%% Romberg
f = @(x) x.^2.*log(x);
int_f = @(x) (x.^3.*(3.*log(x)-1))./9;
a = 1;
b = 3;
n = 5;
R = romberg(f,a,b,n);
error = R(n,n) - (int_f(b)-int_f(a));
fprintf(['Integral is approximatelly = %.6f ' ...
            'when n = %d\n Error = %d\n'] ...
            ,R(n,n),n,error);
function r = romberg(f,a,b,n)
    h = (b-a)./(2.^(0:n-1));
    r(1,1)=(b-a)*(f(a)+f(b))/2;
    for j=2:n
        subtotal = 0;
        for i=1:2^(j-2)
            subtotal = subtotal + f(a+(2*i-1)*h(j));
        end
        r(j,1) = r(j-1,1)/2+h(j)*subtotal;
        for k=2:j
            r(j,k) = ( ...
                4^(k-1)*r(j,k-1)-r(j-1,k-1))/(4^(k-1)-1);
        end
    end
end

% Output:
% Integral is approximatelly = 6.998622 when n = 5
% Error = -3.003336e-09
```

11. Computer problem 5.3.2 d) Use Romberg Integration to approximate the definite integral. As stopping criterion, continue until two successive diagonal entries differ by less than 0.5×10^{-8}

$$(d) \int_1^3 x^2 lnx dx$$

Using the function defined above, we can solve the problem using a while loop and an if condition.

```
%% Romberg
f = @(x) x.^2.*log(x);
int_f = @(x) (x.^3.*(3.*log(x)-1))./9;

tol = 0.5*1e-8;
a = 1;
b = 3;
n = 2;
R = romberg(f,a,b,n);
count = 0;
while count<2
    n=n+1;
    R= romberg(f,a,b,n);
    if R(n,n)-R(n-1,n-1)>tol
        R_prev = R;
        count = 0;
    else
        count = count+1;
    end
end
error = R(n,n) - (int_f(b)-int_f(a));
fprintf(['Integral is approximatelly = %.6f ' ...
            'when n = %d\n Error = %d\n'] ...
            ,R(n,n),n,error);
% Output:
% Integral is approximatelly = 6.998622 when n = 7
% Error = -6.217249e-15
```

12. 5.4.1 a) Apply Adaptive Quadrature by hand, using Trapezoid Rule with tolerance $TOL = 0.05$ to approximate the integrals. Find the approximation error.

$$(a) \int_0^1 x^2 dx$$

$$\begin{aligned}
f(x) &= x^2 \\
\text{Let } trap(f,a,b) &= (f(a)+f(b)) * (b-a)/2 \\
a_1 &= 0 \\
b_1 &= 1 \\
tol_1 &= 0.05 \\
\int_a^b f(x)dx = int &= 0 \\
n &= 1 \\
app_1 &= trap(f,a,b) = 0.5 \\
c &= (a_1+b_1)/2 = 0.5 \\
oldapp &= app_1 = 0.5 \\
app_1 &= trap(f,a_1,c) = 0.0625 \\
app_2 &= trap(f,c,b_1) = 0.3125 \\
|oldapp - (app_1+app_2)| = 0.125 &< 3 * tol_1 = 0.15 \rightarrow true \\
\int_a^b f(x)dx &= int + app_1 + app_2 = 0 + 0.0625 + 0.3125
\end{aligned}$$

13. i) Computer problem 5.4.1 c) Use Adaptive Trapezoid Quadrature to approximation to approximate the definite integral with 0.5×10^{-8}. Report the answer with eight correct decimal places and the number of sub intervals required.

$$(c) \int_0^1 xe^x dx$$

Using the code provided in the book and making a few changes we can get the following results:

```
clear all;
close all;
%% Adaptive Trapezoide quadrature

f = @(x) x.*exp(x);
a = 0;
b = 1;
tol = 0.5*1e-8;
[n,int] = adapquad(f,a,b,tol);

fprintf(['Integral is approximatelly = %.8f ' ...
            'when n = %d\n '],int,n);
%Program 5.2 Adaptive Quadrature
% Computes approximation to definite integral
% Inputs: Matlab function f, interval [a0,b0],
% error tolerance tol0
% Output: approximate definite integral
function [subint,int] =adapquad(f,a0,b0,tol0)
    int=0; n=1; a(1)=a0; b(1)=b0; tol(1)=tol0; app(1)=
       trap(f,a,b);
    subint = 1;
    while n>0 % n is current position at end of the list
        c=(a(n)+b(n))/2; oldapp=app(n);
        app(n)=trap(f,a(n),c);app(n+1)=trap(f,c,b(n));
        if abs(oldapp-(app(n)+app(n+1)))<3*tol(n)
            int=int+app(n)+app(n+1); % success
            n=n-1; % done with interval
        else % divide into two intervals
            b(n+1)=b(n); b(n)=c; % set up new intervals
            a(n+1)=c;
            tol(n)=tol(n)/2; tol(n+1)=tol(n);
            n=n+1; % go to end of list, repeat
        end
        subint = subint+1;
```

```
    end
end
function s=trap(f,a,b)
s=(f(a)+f(b))*(b-a)/2;
end

%Output:
% Integral is approximatelly = 1.00000000 when n = 12424
```

ii) Computer problem 5.4.2 c) Modify the MATLAB code for Adaptive Trapezoid Rule Quadrature to use Simpson's Rule instead, applying the criterion (5.42) with the 15 replaced by 10. Approximate the integral in example 5.12 within 0.005, and compare with Figure 5.5(b). How many sub-intervals were required?

```
%% Adaptive Simpson quadrature

f = @(x) (1+sin(exp(3.*x)));
a = -1;
b = 1;
tol = 0.005;
[n,int] = adapquad(f,a,b,tol);

fprintf(['Integral is approximatelly = %.8f ' ...
            'when n = %d\n '],int,n);

function [subint,int] =adapquad(f,a0,b0,tol0)
    int=0; n=1; a(1)=a0; b(1)=b0; tol(1)=tol0;
    app(1)=simpson(f,a,b);
    subint = 1;
    while n>0 % n is current position at end of the list
        c=(a(n)+b(n))/2; oldapp=app(n);
        app(n)=simpson(f,a(n),c);
        app(n+1)=simpson(f,c,b(n));
        if abs(oldapp-(app(n)+app(n+1)))<10*tol(n)
            int=int+app(n)+app(n+1); % success
            n=n-1; % done with interval
        else % divide into two intervals
            b(n+1)=b(n); b(n)=c; % set up new intervals
            a(n+1)=c;
            tol(n)=tol(n)/2; tol(n+1)=tol(n);
            n=n+1; % go to end of list, repeat
        end
        subint = subint+1;
```

```
    end
end
function s=simpson(f,a,b)
s=(f(a)+f(b)+4*f((a+b)/2))*(b-a)/6;
end
%Output:
% Integral is approximatelly = 2.50028250 when n = 20
```

As we can see, it took only 20 steps to get the result within 0.005.

iii) Computer problem 5.4.3 c) Carry out the steps of Computer problem 1 for adaptive Simpson's Rule, developed in Computer Problem 2.

Using the function defined in the problem aboce, we only need to change, a,b, and f. Which gives

```
%% Adaptive Simpson quadrature

f = @(x) x.*exp(x);
a = 0;
b = 1;
tol = 0.5*1e-8;
[n,int] = adapquad(f,a,b,tol);

fprintf(['Integral is approximatelly = %.8f ' ...
            'when n = %d\n '],int,n);
% Output
%Integral is approximatelly = 1.00000000 when n = 40
```

As we can see, it converges to the solution with respect to the error much quicker than the trapezoid one.