# Homework 3.

Amath 383
Introduction to Continuous Mathematical Modeling



Due: 10/25/22 at 11:59pm to Gradescope

**Directions**:

Complete all exercises as neatly as possible. Up to 3 points may be deducted for homework that is illegible and/or poorly organized. You are encouraged to type homework solutions, and a half bonus point will be awarded to students who use LaTeX. (Check out my LaTeX beginner document and overleaf.com if you are new to LaTeX.) If you prefer not to type homeworks, I ask that **homeworks be scanned.** (I will not accept physical copies.) In addition, **homeworks must be in .pdf format.**

**Pro-Tips**:

- You have access to the textbook, which inspired some of these exercises. You may find that the textbook offers alternative explanations that can help you solve these exercises.
- If a result was already derived in class, no need to rederive it here. Just make sure you cite where in the lecture notes the result your using can be found.
- Teamwork makes the dream work, but please indicate at the top of your assignment who your collaborators are.
- Don't wait until the last minute. ☺

## **Exercise 1.** (Unit 1.5)

Consider the model equation

$$\frac{dP}{dt} = 1 - rP + P^2. \tag{1}$$

a. For what value(s) of $r$ does (1) have two fixed points? How about one fixed point? No fixed points?

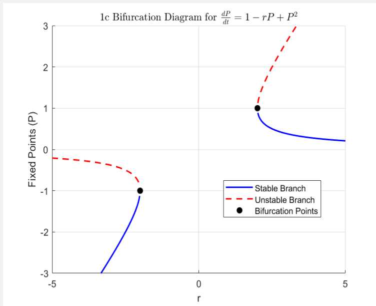
> Based on discriminate of quadratic equation is positive, negative or zero based on $r$.
> $\frac{-(-r)\pm\sqrt{(r)^2-4(1)(1)}}{2(1)}$
> $r^2 - 4 > 0$?
> Two fixed points when $r < -2, r > 2$,
> One fixed points when $r = -2, r = 2$,
> No fixed points when $-2 < r < 2$

b. Show, using the appropriate conditions, that this model equation has two saddle-node bifurcations. What are the bifurcation points?

> $r = 2$ the model equation has two saddle node points because the bifurcation points $(1, 2), (-1, -2)$ satisfies the following saddle point bifurcation equations:
> $F(P^*, r^*) = F(1, 2) = 1 - (2)(1) + (1)^2 = 1 - 2 + 1 = 0$
> $F(P^*, r^*) = F(-1, -2) = 1 - (-2)(-1) + (-1)^2 = 1 - 2 + 1 = 0$
>
> $\frac{dF}{dP}(P^*, r^*) = -r + 2P = -2 + 2 = 0$
> $\frac{dF}{dP}(P^*, r^*) = -r + 2P = 2 - 2 = 0$
>
> $\frac{dF}{dr}(P^*, r^*) = -P = -1 \neq 0$
> $\frac{dF}{dr}(P^*, r^*) = -P = 1 \neq 0$
>
> $\frac{d^2F}{dP^2}(P^*, r^*) = 2 \neq 0$
>
> At $r = 2$, the quadratic equation $P^2 - rP + 1 = 0$ has a double root, meaning the two fixed points merge. This is characteristic of saddle-node bifurcations.

c. Using software of your choice, plot a bifurcation diagram for $-5 \leq r \leq 5$. Be sure to identify the stable and unstable branches in your plot.

Matlab is the software of my choice.




Now consider the model equation

$$\frac{dP}{dt} = P(r - e^P). \tag{2}$$

d. For what value(s) of $r$ does (2) have two fixed points? How about one fixed point? No fixed points?

$\frac{dP}{dt} = P(r - e^P) = 0$

Two fixed points when $r > 0, 0 < r < 1$, at $P = 0, P = \ln(r)$ because natural log is only defined for $r > 0$
One fixed points when $r \leq 0$ and $r = 1$ because there is always at least one fixed point at $P = 0$ and when $r = 1$ then $P = \ln(1) = 0$ so then $P = 0$ is the only fixed point.

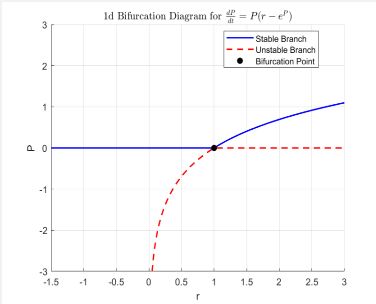
e. Show, using the appropriate conditions, that this model equation has a transcritical bifurcation. What is the bifurcation point?

Transcritical bifurcation
The bifurcation point is $(0, 1)$

$F(P^*, r^*) = 0(1 - e^0) = 0$
$\frac{dF}{dP}(P^*, r^*) = (1 - e^{P^*}) = (1 - e^0) = 1 - 1 = 0$
$\frac{dF}{dr}(P^*, r^*) = P = 0$
$\frac{d^2F}{dPdr}(P^*, r^*) = \frac{dF}{dr}(r - e^P) = 1 \neq 0$
$\frac{d^2F}{dP^2}(P^*, r^*) = -e^P = -e^0 = 1 \neq 0$

f. Using software of your choice, plot a bifurcation diagram for $-1.5 \leq r \leq 3$. Be sure to identify the stable and unstable branches in your plot.

Matlab is the software of my choice




## **Exercise 2.** (Units 1.5-1.6)

The FitzHugh-Nagumo equations were proposed independently by Richard FitzHugh in 1961 and Jin-ichi Nagumo in 1962 as a model for the firing of individual nerve cells, such as neurons in the brain. The (dimensionless) model equations are as follows:

$$\frac{dV}{dt} = V - \frac{1}{3}V^3 - W + I, \tag{3a}$$
$$\frac{dW}{dt} = \frac{1}{T_R}\left(V - \frac{W}{T_F}\right), \tag{3b}$$

where $V$ represents the voltage potential across the membrane of the cell, $I$ represents the net current due to ions flowing into and out of the cell, $W$ is an artificial relaxation variable that controls how $V$ relaxes after the cell fires, $T_R > 0$ is a constant that controls the time scale of the relaxation period between cell firings, and $T_F > 0$ is a constant that controls the time scale for the decay of the relaxation variable, allowing the cell to unrelax and refire.

a. Suppose $T_R \gg 1$, meaning $\frac{dW}{dt}$ is effectively zero. In this case, what is $W$ as a function of $V$ and $T_F$?

$\frac{dW}{dt} = \frac{1}{T_R}\left(V - \frac{W}{T_F}\right) = 0$
$\frac{1}{T_R}\left(V - \frac{W}{T_F}\right) = 0$
$\left(V - \frac{W}{T_F}\right) = 0$
$V = \frac{W}{T_F}$
$\boxed{W = VT_F}$

b. Show that, when $T_R \gg 1$ and there is no net current flowing across the cell membrane, we have

$$\frac{dV}{dt} = V\left(1 - T_F - \frac{1}{3}V^2\right). \tag{4}$$

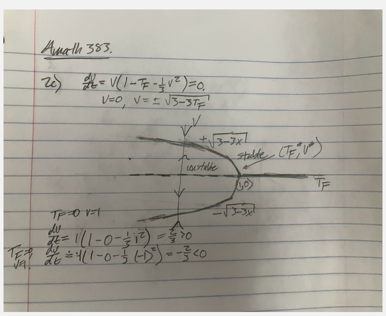
Net current = 0 so $I = 0$
$\frac{dV}{dt} = V - \frac{1}{3}V^3 - W + I$
$\frac{dV}{dt} = V - \frac{1}{3}V^3 - W + (0)$
$\frac{dV}{dt} = V(1 - \frac{1}{3}V^2 - \frac{W}{V})$
$* \frac{W}{V} = T_F$
$\frac{dV}{dt} = V\left(1 - \frac{1}{3}V^2 - T_F\right)$

c. Show, using the appropriate conditions, that (4) has a Pitchfork bifurcation with respect to the parameter $T_F$. Sketch the bifurcation diagram. Is this Pitchfork bifurcation sub- or super-critical?

$\frac{dV}{dt} = V\left(1 - T_F - \frac{1}{3}V^2\right) = 0$

$V = 0$ obvious
$1 - T_F - \frac{1}{3}V^2 = 0$
$-\frac{1}{3}V^2 = -1 + T_F$
$V^2 = 3 - 3T_F$
$V = \pm\sqrt{3 - 3T_F}$

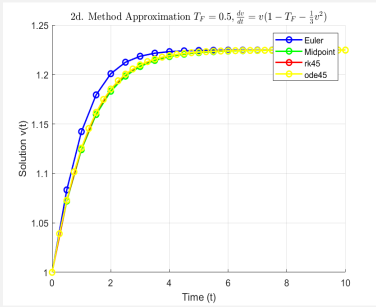
Fixed Points at $V = 0$, $V = +\sqrt{3 - 3T_F}$, $V = -\sqrt{3 - 3T_F}$
Supercritical bifurcation.



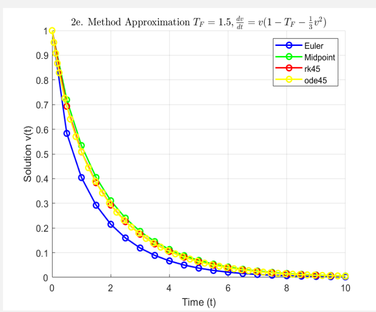

d. Suppose initially $V(0) = 1$. Solve (4) with $T_F = 0.5$ by the following numerical methods:

- Euler's method,
- The midpoint method,
- The Runge-Kutta method,
- MATLAB's ode45 method.

Use $N = 20$ time steps over the interval $0 \le t \le 10$. The code for these methods can be found on the Homework 3 Canvas page. Superimpose the graphs of each numerical solution onto one graph and attach this graph to your homework assignment. *Include a legend on your graph to distinguish each of the numerical solutions.*




e. Repeat the analysis in (d) but with $T_F = 1.5$. How are your plots qualitatively different than those in (d)? How can your analysis in (c) account for this difference?
**Remark**: As you have seen, $T_F$ must be sufficiently small in order for the cell to unrelax and build up voltage potential. (Otherwise, the cell continues to relax.) This makes sense, since a small $T_F$ means that $W$ decays quickly, allowing $V$ to increase more rapidly.




How are your plots qualitatively different than those in (d)?
In part (c) the plots increase from the initial condition 1 to a stable equilibrium of about 1.23
In part (d) the plots decrease from the initial condition 1 to a stable equilibrium of 0

How can your analysis in (c) account for this difference?
In part (c) the bifurcation diagram shows that when $T_F < 1$ there are three equilibrium two stable with values $V = \pm\sqrt{3 - 3T_F}$ and one unstable with value $V = 0$. An initial condition of 1 would go towards the $+\sqrt{3 - 3T_F}$ in the long term. While when $T_F > 1$ there is only one stable equilibrium at $V = 0$. Meaning an initial condition of 1 would go to 0 in the long term.

## **Exercise 3.** (Unit 1.6)

Consider the logistic equation with initial growth rate $r = 1$ and carrying capacity $K = 1$:

$$\frac{dP}{dt} = P(1 - P).$$

a. Find the exact solution of this equation satisfying the initial condition $P(0) = \frac{1}{2}$.

$\frac{dP}{dt} = P(1 - P).$
$\frac{dP}{P(1-P)} = dt$
$\ln(P) - \ln(P - 1) = t + C_1$
$\ln\left(\frac{P}{P-1}\right) = t + C_1$
$\frac{P}{P-1} = e^{t+C_1}$
$P = \frac{1}{1+C_1 e^{-t}}$

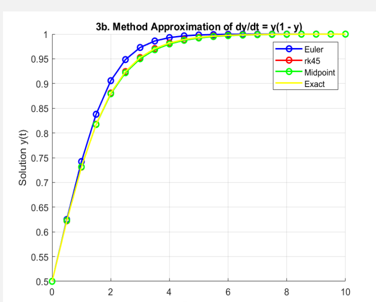
$P(0) = \frac{1}{2} = \frac{1}{1+C_2 e^{-(0)}}$
$= \frac{1}{2} = \frac{1}{1+C_2}$
$C_2 = 1$

$$\boxed{P(t) = \frac{1}{1 + e^{-t}}}$$

b. Find the solution of this equation satisfying the initial condition $P(0) = 1/2$ by the following numerical methods:

- Euler's method,
- The midpoint method,
- The Runge-Kutta method.

Use $N = 20$ time steps over the interval $0 \le t \le 10$. Superimpose the graph of the exact solution and the graphs of each of the numerical solutions onto one graph and attach your graph to your homework assignment. *Include a legend on your graph to distinguish each of the solutions.*




Define the error of each method as the maximum difference in absolute value between the exact solution and the numerical solution over $0 \le t \le 10$:

$$\mathcal{E} = \max_{0 \le t \le 10} \left| P_{\text{exact}}(t) - P_{\text{numerical}}(t) \right|.$$

What is the error of each numerical method?

**Hint**: You may find MATLAB's built-in max function useful.

$error_{eu} = 0.0250$
$error_{mi} = 0.0024$
$error_{rk} = 3.6566\text{e} - 05$

c. Obtain the errors for each numerical method using $N = 200$ time steps. Have the errors improved?

$error_{eu} = 0.0023$
$error_{mi} = 1.8708\text{e} - 05$
$error_{rk} = 2.9289\text{e} - 09$
Error values have decreased for all of them.

d. According to theory, the error $\mathcal{E}$ of a finite-difference method is approximately given by

$$\mathcal{E} = Ch^p,$$

where $h$ is the step size used for the method, $p$ is the order of the method, and $C$ is a constant of proportionality. Given two errors $\mathcal{E}_1$ and $\mathcal{E}_2$ for two different step sizes $h_1$ and $h_2$, show that

$$p = \frac{\ln\left(\frac{\mathcal{E}_2}{\mathcal{E}_1}\right)}{\ln\left(\frac{h_2}{h_1}\right)}.$$

$\mathcal{E}_1 = Ch_1^p, \mathcal{E}_2 = Ch_2^p$
$\ln(\mathcal{E}_1) = p\ln(Ch_1), \ln(\mathcal{E}_2) = p\ln(Ch_2)$
$\ln(\mathcal{E}_2) - \ln(\mathcal{E}_1) = p\ln(Ch_2) - p\ln(Ch_1)$
$\ln(\mathcal{E}_2) - \ln(\mathcal{E}_1) = p(\ln(Ch_2) - \ln(Ch_1))$
$\ln\left(\frac{\mathcal{E}_2}{\mathcal{E}_1}\right) = p\ln\left(\frac{Ch_2}{Ch_1}\right)$
$\ln\left(\frac{\mathcal{E}_2}{\mathcal{E}_1}\right) = p\ln\left(\frac{Ch_2}{Ch_1}\right)$
$\ln\left(\frac{\mathcal{E}_2}{\mathcal{E}_1}\right) = p\ln\left(\frac{h_2}{h_1}\right)$
$p = \frac{\ln\left(\frac{\mathcal{E}_2}{\mathcal{E}_1}\right)}{\ln\left(\frac{h_2}{h_1}\right)}$

e. Using the errors computed in (b) and (c) and the formula derived in (d), estimate the order of Euler's method, the midpoint method, and the Runge-Kutta method. Do these orders match what you expect?

$p = \frac{\ln\left(\frac{\varepsilon}{\varepsilon}\right)}{\ln\left(\frac{h_2}{h_1}\right)}$

$p_{eu} = \frac{\ln\left(\frac{0.0023}{0.0250}\right)}{\ln\left(\frac{200}{20}\right)} = 1.0362 \approx 1$

$p_{mi} = \frac{\ln\left(\frac{1.8708e-05}{0.0024}\right)}{\ln\left(\frac{200}{20}\right)} = 2.1082 \approx 2$

$p_{rk} = \frac{\ln\left(\frac{2.9289e-09}{3.6566e-05}\right)}{\ln\left(\frac{200}{20}\right)} = 4.0964 \approx 4$

Yes the orders match what I expected with higher the order the lower the error.
euler method is first order
midpoint method is second order
fourth order runge-kutta method is indeed fourth order

```matlab
%% AMATH 383 HW 3

%% Excercise 1c
%dp/dt = 1 - rP + P^2

% Define the range for r and the function for the fixed points
r_values = linspace(-5, 5, 500);
fixed_points_positive = NaN(length(r_values), 1);
fixed_points_negative = NaN(length(r_values), 1);

% Compute fixed points based on the discriminant condition
for i = 1:length(r_values)
    r = r_values(i);
    discriminant = r^2 - 4;
    if discriminant >= 0 % Real roots exist
        % quadformula for P.
        root1 = (r + sqrt(discriminant)) / 2;
        root2 = (r - sqrt(discriminant)) / 2;
        fixed_points_positive(i) = root2;
        fixed_points_negative(i) = root1;
    end
end

% Plot the bifurcation diagram
figure;
hold on;
plot(r_values, fixed_points_positive, 'b-', 'LineWidth', 1.5); % Stable Branch
plot(r_values, fixed_points_negative, 'r--', 'LineWidth', 1.5); % Unstable Branch

% Highlight bifurcation points
plot([-2, 2], [-1, 1], 'ko', 'MarkerFaceColor', 'k', 'MarkerSize', 6); % Bifurcation Points

% Formatting the plot
ax = gca; % gridlines and axis features.
ax.XLim = [-5, 5];
ax.YLim = [-3, 3];
ax.XGrid = 'on';
ax.YGrid = 'on';
title('1c Bifurcation Diagram for $\frac{dP}{dt} = 1 - rP + P^2$','Interpreter','latex');
xlabel('r');
ylabel('Fixed Points (P)');
legend('Stable Branch', 'Unstable Branch', 'Bifurcation Points',
```

```
    'Location', 'best');
hold off;

%% Excercise 1d
%dP/dt = P(r-e^P)

% Define the range for r and the function for the fixed points
r_values = linspace(-1.5, 3, 500);
fixed_points_positive = NaN(length(r_values), 1);
fixed_points_negative = NaN(length(r_values), 1);

% Compute fixed points based on the discriminant condition
for i = 1:length(r_values)
    r = r_values(i);
    if r < 1
        fixed_points_positive(i) = 0;
    end
    if r >0 && r < 1
        fixed_points_negative(i) = log(r);
    end
    if r > 1
        fixed_points_positive(i) = log(r);
        fixed_points_negative(i) = 0;
    end
end

% Plot the bifurcation diagram
figure;
hold on;
plot(r_values, fixed_points_positive, 'b-', 'LineWidth', 1.5); % Stable Branch
plot(r_values, fixed_points_negative, 'r--', 'LineWidth', 1.5); % Unstable Branch

% Highlight bifurcation points
plot([1], [0], 'ko', 'MarkerFaceColor', 'k', 'MarkerSize', 6); % Bifurcation Points

% Formatting the plot
ax = gca; % gridlines and axis features.
ax.XLim = [-1.5, 3];
ax.YLim = [-3, 3];
ax.XGrid = 'on';
ax.YGrid = 'on';
title('1d Bifurcation Diagram for $\frac{dP}{dt} = P(r-e^P)$','Interpreter','latex');
xlabel('r');
```

```
ylabel('P');
legend('Stable Branch', 'Unstable Branch', 'Bifurcation Point', 'Location', 'best');
hold off;



%% Excercise 2d
V_0 = 1;
T_F = 0.5
ode_RHS = @(t,v) v*(1 - T_F - (1/3)*v^2);

[t2d_eu, soln2d_eu] = UW_euler_method_383(ode_RHS, [0, 10], 20, V_0)
[t2d_mi, soln2d_mi] = UW_midpoint_method_383(ode_RHS, [0, 10] , 20, V_0)
[t2d_rk, soln2d_rk] = UW_rk4_method_383(ode_RHS, [0, 10], 20, V_0)
[t2d_45, soln2d_45] = ode45(ode_RHS,[0,10],V_0)

figure;
hold on
ax = gca; % gridlines and axis features.
%ax.XLim = [-1.5, 3];
ax.YLim = [0, 2.0];
ax.XGrid = 'on';
ax.YGrid = 'on';

plot(t2d_eu, soln2d_eu, '-o', 'LineWidth', 1.5,'Color','blue');
plot(t2d_mi, soln2d_mi, '-o', 'LineWidth', 1.5,'Color','green');
plot(t2d_rk, soln2d_rk, '-o', 'LineWidth', 1.5,'Color','red');
plot(t2d_45, soln2d_45, '-o', 'LineWidth', 1.5,'Color','yellow');
xlabel('Time (t)');
ylabel('Solution v(t)');

title('2d. Method Approximation $T_F = 0.5, \frac{dv}{dt} = v(1 - T_F - \frac13v^2)$','Interpreter','latex');
legend('Euler','Midpoint','rk45','ode45')
grid on;
hold off

%% Excercise 2e
T_F = 1.5
ode_RHS = @(t,v) v*(1 - T_F - (1/3)*v^2);

[t2e_eu, soln2e_eu] = UW_euler_method_383(ode_RHS, [0, 10], 20,
```

```
    1)
[t2e_mi, soln2e_mi] = UW_midpoint_method_383(ode_RHS, [0, 10] ,
    20, 1)
[t2e_rk, soln2e_rk] = UW_rk4_method_383(ode_RHS, [0, 10], 20, 1)
[t2e_45, soln2e_45] = ode45(ode_RHS,[0,10],1)

figure;
hold on
ax = gca; % gridlines and axis features.
%ax.XLim = [-1.5, 3];
ax.YLim = [0, 2.0];
ax.XGrid = 'on';
ax.YGrid = 'on';

plot(t2e_eu, soln2e_eu, '-o', 'LineWidth', 1.5,'Color','blue');
plot(t2e_mi, soln2e_mi, '-o', 'LineWidth', 1.5,'Color','green');
plot(t2e_rk, soln2e_rk, '-o', 'LineWidth', 1.5,'Color','red');
plot(t2e_45, soln2e_45, '-o', 'LineWidth', 1.5,'Color','yellow')
    ;
xlabel('Time (t)');
ylabel('Solution v(t)');
title('2e. Method Approximation $T_F = 1.5, \frac{dv}{dt} = v(1
    - T_F - \frac13v^2)$','Interpreter','latex');
legend('Euler','Midpoint','rk45','ode45')
grid on;
hold off

%% Excercise 3.
%num_steps = 20;
%init_cond = 0.5;
%t_start = 0;
%t_end = 10;

num_steps = 20;
t_range = [0,10];
h = (t_range(2)-t_range(1))/num_steps;
t = t_range(1):h:t_range(2);
y_exact = exp(t) ./ (1 + exp(t));

ode_RHS = @(t,y) y*(1-y);
[t3b_eu, soln3b_eu] = UW_euler_method_383(ode_RHS, [0, 10], 20,
    0.5)
[t3b_rk, soln3b_rk] = UW_rk4_method_383(ode_RHS, [0, 10], 20,
    0.5)
[t3b_mi, soln3b_mi] = UW_midpoint_method_383(ode_RHS, [0, 10] ,
    20, 0.5)

```

```
% Plot the solution
figure;
hold on
plot(t3b_eu, soln3b_eu, '-o', 'LineWidth', 1.5,'Color','blue');
plot(t3b_rk, soln3b_rk, '-o', 'LineWidth', 1.5,'Color','red');
plot(t3b_mi, soln3b_mi, '-o', 'LineWidth', 1.5,'Color','green');
plot(t, y_exact, '-', 'LineWidth', 1.5,'Color','yellow' )
xlabel('Time (t)');
ylabel('Solution y(t)');
title('3b. Method Approximation of dy/dt = y(1 - y)');
legend('Euler','rk45','Midpoint','Exact')
grid on;
hold off

%% Part b error N = 20
num_steps = 20;
t_range = [0,10];
h = (t_range(2)-t_range(1))/num_steps;
t = t_range(1):h:t_range(2);
y_exact = exp(t) ./ (1 + exp(t));

% Compute error
error_eu = max(abs(y_exact - soln3b_eu))
error_rk = max(abs(y_exact - soln3b_rk))
error_mi = max(abs(y_exact - soln3b_mi))


%% Part c error N = 200
num_steps = 200;
t_range = [0,10];
h = (t_range(2)-t_range(1))/num_steps;
t = t_range(1):h:t_range(2);
y_exact = exp(t) ./ (1 + exp(t));

[t3c_eu, soln3c_eu] = UW_euler_method_383(ode_RHS, [0, 10], 200,
    0.5);
[t3c_rk, soln3c_rk] = UW_rk4_method_383(ode_RHS, [0, 10], 200,
   0.5);
[t3c_mi, soln3c_mi] = UW_midpoint_method_383(ode_RHS, [0, 10] ,
   200, 0.5);

% Compute error
error_eu = max(abs(y_exact - soln3c_eu))
error_rk = max(abs(y_exact - soln3c_rk))
error_mi = max(abs(y_exact - soln3c_mi))
```

```matlab

%%

function [t,soln] = UW_euler_method_383(ode_RHS, t_range,
   num_steps,init_cond)
    h = (t_range(2)-t_range(1))/num_steps;
    t = t_range(1):h:t_range(2);
    soln = [init_cond,nan(1,length(t)-1)];
% ode_RHS is the function handle of the RHS of the ODE, i.e.,
% num_steps is the number of time steps from t_start
% init_cond is the initial condition
%t_range = [t_start t_end];

    for j = 1:length(t)-1
        soln(j+1) = soln(j) + h*ode_RHS(t(j),soln(j));
    end
end


function [t,soln] = UW_rk4_method_383(ode_RHS,t_range,num_steps,
   init_cond)

% ode_RHS is the function handle of the RHS of the ODE, i.e.,
%ode_RHS = @(t,y) y
% t_range = [t_start t_end];
% num_steps is the number of time steps from t_start
% init_cond is the initial condition

h = (t_range(2)-t_range(1))/num_steps;
t = t_range(1):h:t_range(2);
soln = [init_cond,nan(1,length(t)-1)];

for j = 1:length(t)-1
K1 = ode_RHS(t(j),soln(j));
K2 = ode_RHS(t(j)+h/2,soln(j)+h*K1/2);
K3 = ode_RHS(t(j)+h/2,soln(j)+h*K2/2);
K4 = ode_RHS(t(j)+h,soln(j)+h*K3);
soln(j+1) = soln(j) + h/6*(K1+2*K2+2*K3+K4);
end
end


function [t,soln] = UW_midpoint_method_383(ode_RHS,t_range,
   num_steps,init_cond)

% ode_RHS is the function handle of the RHS of the ODE, i.e.,
% ode_RHS = @(t,y) [insert RHS of ODE here]
```

```
% t_range = [t_start t_end]
% num_steps is the number of time steps from t_start
% init_cond is the initial condition

h = (t_range(2)-t_range(1))/num_steps;
t = t_range(1):h:t_range(2);
soln = [init_cond,nan(1,length(t)-1)];

for j = 1:length(t)-1
soln(j+1) = soln(j) + h*ode_RHS(t(j)+h/2,soln(j)+h/2*...
    ode_RHS(t(j),soln(j)));
end
end
```