AE 370: HW5

Linyi Hou

March 6, 2020

Problem 1

a. Integrating the ODE:

$$\int_{t_k}^{t_{k+1}} \dot{u}\, dt = \int_{t_k}^{t_{k+1}} f(u(t), t) dt \tag{1}$$

Substituting in the Lagrange basis for trapezoidal method:

$$u(t_{k+1}) - u(t_k) \approx \int_{t_k}^{t_{k+1}} f(u(t_k), t_k) \left(\frac{t - t_{k+1}}{-\Delta t} \right) + f(u(t_{k+1}), t_{k+1}) \left(\frac{t - t_k}{\Delta t} \right) dt \tag{2}$$

Extracting constants:

$$\begin{aligned} u(t_{k+1}) - u(t_k) &\approx \frac{f(u(t_k), t_k)}{-\Delta t} \int_{t_k}^{t_{k+1}} (t - t_{k+1}) dt + \frac{f(u(t_{k+1}), t_{k+1})}{\Delta t} \int_{t_k}^{t_{k+1}} (t - t_k) dt && (3) \\ &= \frac{\Delta t}{2} \left(f(u(t_k), t_k) + f(u(t_{k+1}), t_{k+1}) \right) && (4) \end{aligned}$$

Since $u(t_k)$ is generally not known, we substitute all $u(t_k)$ with u_k to represent the approximate solution to the IVP at time t_k. Thus Equation (4) becomes:

$$u_{k+1} - u_k = \frac{\Delta t}{2} \left(f(u_k, t_k) + f(u_{k+1}, t_{k+1}) \right) \tag{5}$$

To find the truncation error, we subtract the left side of Equation (3) from the right side of Equation (4) and divide by Δt:

$$\tau_k = \frac{u(t_{k+1}) - u(t_k)}{\Delta t} - \frac{1}{2} \left(f(u(t_k), t_k) + f(u(t_{k+1}), t_{k+1}) \right) \tag{6}$$

Retrieving the ODE from Equation (1):

$$\boxed{\tau_k = \frac{u(t_{k+1}) - u(t_k)}{\Delta t} - \frac{1}{2} \left(\dot{u}(t_k) + \dot{u}(t_{k+1}) \right)} \tag{7}$$

We now Taylor expand u and $\dot{u}$ at t_{k+1}:

$$\begin{aligned} u(t_{k+1}) &= u(t_k) + \Delta t \dot{u}(t_k) + \frac{\Delta t^2}{2} \ddot{u}(t_k) + \frac{\Delta t^3}{6} \dddot{u}(t_k) + \text{H.O.T.} && (8) \\ \dot{u}(t_{k+1}) &= \dot{u}(t_k) + \Delta t \ddot{u}(t_k) + \frac{\Delta t^2}{2} \dddot{u}(t_k) + \text{H.O.T.} && (9) \end{aligned}$$

Substituting Equations (8) and (9) into Equation (7), we find:

$$\tau_k = -\frac{\Delta t^2}{6}\ddot{u}(t_k) + \text{H.O.T.} \tag{10}$$

$$\boxed{\tau_k = \mathcal{O}(\Delta t^2)} \tag{11}$$

b. The two-step Adams-Bashforth method has form

$$u_{k+1} = u_k + \frac{\Delta t}{2}\left[-f(u_{k-1}, t_{k-1}) + 3f(u_k, t_k)\right] \tag{12}$$

Rewriting the equation by substituting approx. values u_k with true values $u(t_k)$:

$$u(t_{k+1}) = u(t_k) + \frac{\Delta t}{2}\left[-f(u(t_{k-1}), t_{k-1}) + 3f(u(t_k), t_k)\right] \tag{13}$$

Rearranging and dividing by Δt to find the truncation error:

$$\tau_k = \frac{u(t_{k+1}) - u(t_k)}{\Delta t} - \frac{1}{2}\left[-f(u(t_{k-1}), t_{k-1}) + 3f(u(t_k), t_k)\right] \tag{14}$$

Retrieving the ODE from Equation (1):

$$\boxed{\tau_k = \frac{u(t_{k+1}) - u(t_k)}{\Delta t} - \frac{1}{2}\left[-\dot{u}(t_{k-1}) + 3\dot{u}(t_k)\right]} \tag{15}$$

We now Taylor expand u and $\dot{u}$ at t_{k+1} and t_{k-1} respectively:

$$u(t_{k+1}) = u(t_k) + \Delta t\dot{u}(t_k) + \frac{\Delta t^2}{2}\ddot{u}(t_k) + \frac{\Delta t^3}{6}\dddot{u}(t_k) + \text{H.O.T.} \tag{16}$$

$$\dot{u}(t_{k-1}) = \dot{u}(t_k) - \Delta t\ddot{u}(t_k) + \frac{\Delta t^2}{2}\dddot{u}(t_k) + \text{H.O.T.} \tag{17}$$

Substituting Equations (16) and (17) into Equation (15):

$$\begin{aligned}\tau_k = {} & \frac{u(t_k) + \Delta t\dot{u}(t_k) + \frac{\Delta t^2}{2}\ddot{u}(t_k) + \frac{\Delta t^3}{6}\dddot{u}(t_k) + \text{H.O.T.} - u(t_k)}{\Delta t} \\ & - \frac{1}{2}\left[-\left(\dot{u}(t_k) - \Delta t\ddot{u}(t_k) + \frac{\Delta t^2}{2}\dddot{u}(t_k) + \text{H.O.T.}\right) + 3\dot{u}(t_k)\right]\end{aligned} \tag{18}$$

Collecting terms:

$$\tau_k = \frac{5}{12}\Delta t^2\dddot{u}(t_k) + \text{H.O.T.} \tag{19}$$

$$\boxed{\tau_k = \mathcal{O}(\Delta t^2)} \tag{20}$$

Problem 2

a. See figures and code attached in Appendix A and Appendix B, respectively.

b. See code attached in Appendix B.
First, for each method, we manually adjust the number of time steps taken between $t = 0$ and $t = 50$ until the minimum number of time steps (conversely, the maximum dt) was found that satisfies the convergence requirement of 6×10^{-5}.
Then, using the tic and toc functions, we propagate the approximations to determine the average run time. The results are as follows:

Adams-Bashforth 2-Step:	0.029 seconds	43233 steps
Heun's Method:	0.015 seconds	17362 steps
Four-Stage Runge-Kutta:	0.0053 seconds	1711 steps

Both Heun's method (two-stage Runge-Kutta) and the four-stage Runge-Kutta method are variants of the trapezoid method. Both methods show quicker convergence times and fewer steps required as compared to the two-step Adams-Bashforth method, which suggests that the trapezoid method is better suited for solving the predator-prey problem.

c. For a two-step Adams-Moulton method:

$$u_{k+1} = u_k + \frac{\Delta t}{12}\left[-f(u_{k-1}, t_{k-1}) + 8f(u_k, t_k) + 5f(u_{k+1}, t_{k+1})\right] \tag{21}$$

The only unknown in this equation is u_{k+1}. Since for the predator-prey problem $f(u_k, t_k) = f(u_k)$, Equation (21) becomes:

$$u_{k+1} = u_k + \frac{\Delta t}{12}\left[-f(u_{k-1}) + 8f(u_k) + 5f(u_{k+1})\right] \tag{22}$$

This equation would be solved in MATLAB using syms, where we set up two variables x, y such that $u_{k+1} = [x, y]^T$. Thus Equation (22) becomes a system of two equations with two variables, which can be easily solved.

Appendix A: Figures

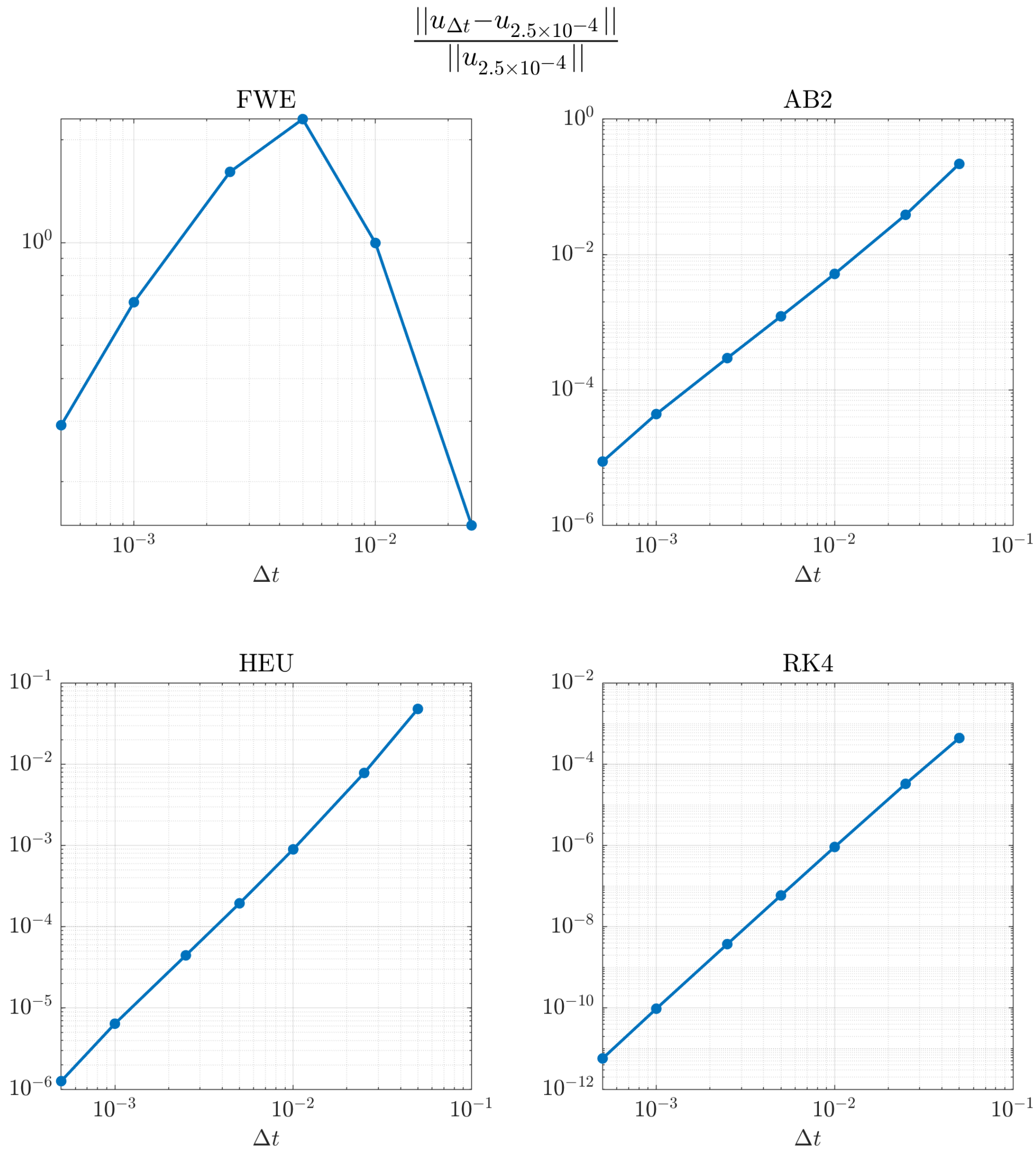

Figure 1: Error Plot for Problem 2a

Appendix B: Code

```
%% Problem 2 Part a
close all
clear;clc

%--problem parameters

    b = 2; %birth rate of prey
    p = 1; %effect of predation on prey
    d = 3; %death rate of predators
    g = 1; %growth of predators due to eating prey

    t0 = 0; %starting time
    T = 50; %final time

    u0 = [1;1]; %IC

    f = @(u) [(b-p*u(2))*u(1); (g*u(1)-d)*u(2)]; %RHS

%--

%part a)
    %--simulation params
        dtvect = [5e-2, 2.5e-2, 1e-2, 5e-3, 2.5e-3, 1e-3, 5e-4, 2.5e-4];
    %--

    %initialize vector that stores approximate soln at T for various dt
    u_FWE_keep = zeros( 2,length( dtvect ) ); % forward Euler
    u_AB2_keep = zeros( 2,length( dtvect ) ); % Adams-Bashforth 2-step
    u_HEU_keep = zeros( 2,length( dtvect ) ); % Heun's
    u_RK4_keep = zeros( 2,length( dtvect ) ); % Runge-Kutta 4


    %advance in time
    for j = 1 : length(dtvect)

        %current dt
        dt = dtvect(j);

```

```
    tk = t0; %initialize time iterate

    %initialize iterates for various methods
    % FWE
    u_FWE_k = u0;
    % AB2
    u_AB2_k = u0;
    u_AB2_km1 = u0;
    % HEU
    u_HEU_k = u0;
    % RK4
    u_RK4_k = u0;


    %run to final time T
    for jj = 1 : T/dt

        % ============= FWE =============
        % propagate and update
        u_FWE_k = u_FWE_k + f(u_FWE_k) * dt;

        % ============= AB2 =============
        % propagate
        if jj < 2
            %advance with Heun's for 1st time step
            u_AB2_kp1 = u_AB2_k + ...
                        0.5 * dt * ...
                        ( f(u_AB2_k) + f(u_AB2_k + dt * f(u_AB2_k)) );
        else
            u_AB2_kp1 = u_AB2_k + dt / 2 * ...
                        ( -f(u_AB2_km1) + 3 * f(u_AB2_k) );
        end
        % update iterates
        u_AB2_km1 = u_AB2_k;
        u_AB2_k = u_AB2_kp1;

        % ============= HEU =============
        % propagate and update
        u_HEU_k = u_HEU_k + ...
```

```
                    0.5 * dt * ...
                    ( f(u_HEU_k) + f(u_HEU_k + dt * f(u_HEU_k)) );

        % ============= RK4 =============
        % propagate and update
        y1 = f(u_RK4_k);
        y2 = f(u_RK4_k + dt/2 * y1);
        y3 = f(u_RK4_k + dt/2 * y2);
        y4 = f(u_RK4_k + dt * y3);
        u_RK4_k = u_RK4_k + 1/6 * dt * (y1 + 2*y2 + 2*y3 + y4);


        % ============= update time =============

        tk = tk + dt;

    end

    % store solution at T
    % FWE
    u_FWE_keep(:,j) = u_FWE_k;
    % AB2
    u_AB2_keep(:,j) = u_AB2_kp1;
    % HEU
    u_HEU_keep(:,j) = u_HEU_k;
    % RK4
    u_RK4_keep(:,j) = u_RK4_k;

end

%compute difference between solution at smallest dt and the other dts

%initialize vector
u_FWE_diff = zeros( length( dtvect )-1,1 );
u_AB2_diff = zeros( length( dtvect )-1,1 );
u_HEU_diff = zeros( length( dtvect )-1,1 );
u_RK4_diff = zeros( length( dtvect )-1,1 );

for j = 1 : length( dtvect )-1

```

```
        u_FWE_diff(j) = norm(u_FWE_keep(:,j) - u_FWE_keep(:,end)) ...
                            / norm(u_FWE_keep(:,end));
        u_AB2_diff(j) = norm(u_AB2_keep(:,j) - u_AB2_keep(:,end)) ...
                            / norm(u_AB2_keep(:,end));
        u_HEU_diff(j) = norm(u_HEU_keep(:,j) - u_HEU_keep(:,end)) ...
                            / norm(u_HEU_keep(:,end));
        u_RK4_diff(j) = norm(u_RK4_keep(:,j) - u_RK4_keep(:,end)) ...
                            / norm(u_RK4_keep(:,end));

    end

    f = figure(1);

    subplot(2,2,1)
    loglog( dtvect(1:end-1), u_FWE_diff, '.-', 'markersize', 25, 'linewidth', 2 )
    title('FWE', 'fontsize', 16, 'interpreter' , 'latex')
    set( gca, 'Color', [1 1 1] )
    set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )
    xlabel('$\Delta t$', 'fontsize', 16, 'interpreter' , 'latex')
    grid(gca,'minor')
    grid on

    subplot(2,2,2)
    loglog( dtvect(1:end-1), u_AB2_diff, '.-', 'markersize', 25, 'linewidth', 2 )
    title('AB2', 'fontsize', 16, 'interpreter' , 'latex')
    set( gca, 'Color', [1 1 1] )
    set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )
    xlabel('$\Delta t$', 'fontsize', 16, 'interpreter' , 'latex')
    grid(gca,'minor')
    grid on

    subplot(2,2,3)
    loglog( dtvect(1:end-1), u_HEU_diff, '.-', 'markersize', 25, 'linewidth', 2 )
    title('HEU', 'fontsize', 16, 'interpreter' , 'latex')
    set( gca, 'Color', [1 1 1] )
    set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )
    xlabel('$\Delta t$', 'fontsize', 16, 'interpreter' , 'latex')
```

```
    grid(gca,'minor')
    grid on

    subplot(2,2,4)
    loglog( dtvect(1:end-1), u_RK4_diff, '.-', 'markersize', 25, 'linewidth', 2 )
    title('RK4', 'fontsize', 16, 'interpreter' , 'latex')
    set( gca, 'Color', [1 1 1] )
    set( gca, 'fontsize', 16, 'ticklabelinterpreter', 'latex' )
    xlabel('$\Delta t$', 'fontsize', 16, 'interpreter' , 'latex')
    grid(gca,'minor')
    grid on


    sgtitle('$\frac{||u_{\Delta t} - u_{2.5\times10^{-4}}||}{|| u_{2.5\times10^{-4}} ||}$', 'fontsize', 28,  'interpreter' , 'latex' )
    set(f, 'PaperPositionMode', 'manual')
    set(f, 'Color', [1 1 1])
    set(f, 'PaperUnits', 'centimeters')
    set(f, 'PaperSize', [30 30])
    set(f, 'Units', 'centimeters' )
    set(f, 'Position', [0 0 30 30])
    set(f, 'PaperPosition', [0 0 30 30])

    svnm = 'error_q2';
    print( '-dpng', svnm, '-r200' )

    % store the reference values for all four methods in a .mat file
    u_FWE_ref = u_FWE_keep(:,end);
    u_AB2_ref = u_AB2_keep(:,end);
    u_HEU_ref = u_HEU_keep(:,end);
    u_RK4_ref = u_RK4_keep(:,end);

    save('ref_data.mat','u_FWE_ref','u_AB2_ref','u_HEU_ref','u_RK4_ref');

%% Problem 2 Part b, AB2
close all
clear;clc

% maunally edit this variable for convergence
```

```
num_steps = 43233;

% simulation params
b = 2; %birth rate of prey
p = 1; %effect of predation on prey
d = 3; %death rate of predators
g = 1; %growth of predators due to eating prey
u0 = [1;1]; %IC
f = @(u) [(b-p*u(2))*u(1); (g*u(1)-d)*u(2)]; %RHS

tk = 0; % starting time
T = 50; % final time
dt = (T-tk)/num_steps; % time step

% load reference value
load('ref_data.mat','u_AB2_ref');

% ====================== BEGIN TIMER ========================
tic

%initialize iterates for AB2
u_AB2_k = u0;
u_AB2_km1 = u0;

%advance to final time T
for jj = 1 : T/dt

    % ============= AB2 =============
    % propagate
    if jj < 2
        %advance with Heun's for 1st time step
        u_AB2_kp1 = u_AB2_k + ...
                    0.5 * dt * ...
                    ( f(u_AB2_k) + f(u_AB2_k + dt * f(u_AB2_k)) );
    else
        u_AB2_kp1 = u_AB2_k + dt / 2 * ...
                    ( -f(u_AB2_km1) + 3 * f(u_AB2_k) );
    end
    % update iterates
    u_AB2_km1 = u_AB2_k;
```

```
    u_AB2_k = u_AB2_kp1;
    tk = tk + dt;

end

% ====================== END TIMER ========================
t_AB2 = toc;

% manually check for convergence
u_AB2_keep = u_AB2_kp1;
u_AB2_diff = norm(u_AB2_keep - u_AB2_ref) / norm(u_AB2_ref);
disp(['Error: ' num2str(u_AB2_diff)])
if u_AB2_diff <= 6e-5
    disp('Successfully converged!');
else
    disp('Failed to converge.');
end
disp(['Time to converge: ' num2str(t_AB2) ' seconds'])


%% Problem 2 Part b, HEU
close all
clear;clc

% maunally edit this variable for convergence
num_steps = 17362;

% simulation params
b = 2; %birth rate of prey
p = 1; %effect of predation on prey
d = 3; %death rate of predators
g = 1; %growth of predators due to eating prey
u0 = [1;1]; %IC
f = @(u) [(b-p*u(2))*u(1); (g*u(1)-d)*u(2)]; %RHS

tk = 0; % starting time
T = 50; % final time
dt = (T-tk)/num_steps; % time step

% load reference value
```

```
load('ref_data.mat','u_HEU_ref');

% ====================== BEGIN TIMER ========================
tic

%initialize iterates for HEU
u_HEU_k = u0;

%advance to final time T
for jj = 1 : T/dt

    % ============= HEU =============
    % propagate and update
    u_HEU_k = u_HEU_k + ...
              0.5 * dt * ...
              ( f(u_HEU_k) + f(u_HEU_k + dt * f(u_HEU_k)) );
    tk = tk + dt;

end

% ====================== END TIMER ========================
t_HEU = toc;

% manually check for convergence
u_HEU_keep = u_HEU_k;
u_HEU_diff = norm(u_HEU_keep - u_HEU_ref) / norm(u_HEU_ref);
disp(['Error: ' num2str(u_HEU_diff)])
if u_HEU_diff <= 6e-5
    disp('Successfully converged!');
else
    disp('Failed to converge.');
end
disp(['Time to converge: ' num2str(t_HEU) ' seconds'])


%% Problem 2 Part b, RK4
close all
clear;clc

% maunally edit this variable for convergence
```

```
num_steps = 1711;

% simulation params
b = 2; %birth rate of prey
p = 1; %effect of predation on prey
d = 3; %death rate of predators
g = 1; %growth of predators due to eating prey
u0 = [1;1]; %IC
f = @(u) [(b-p*u(2))*u(1); (g*u(1)-d)*u(2)]; %RHS

tk = 0; % starting time
T = 50; % final time
dt = (T-tk)/num_steps; % time step

% load reference value
load('ref_data.mat','u_RK4_ref');

% ====================== BEGIN TIMER =======================
tic

%initialize iterates for RK4
u_RK4_k = u0;

%advance to final time T
for jj = 1 : T/dt

    % ============= RK4 =============
    % propagate and update
    y1 = f(u_RK4_k);
    y2 = f(u_RK4_k + dt/2 * y1);
    y3 = f(u_RK4_k + dt/2 * y2);
    y4 = f(u_RK4_k + dt * y3);
    u_RK4_k = u_RK4_k + 1/6 * dt * (y1 + 2*y2 + 2*y3 + y4);
    tk = tk + dt;

end

% ====================== END TIMER =======================
t_RK4 = toc;

```

```
% manually check for convergence
u_RK4_keep = u_RK4_k;
u_RK4_diff = norm(u_RK4_keep - u_RK4_ref) / norm(u_RK4_ref);
disp(['Error: ' num2str(u_RK4_diff)])
if u_RK4_diff <= 6e-5
    disp('Successfully converged!');
else
    disp('Failed to converge.');
end
disp(['Time to converge: ' num2str(t_RK4) ' seconds'])
```