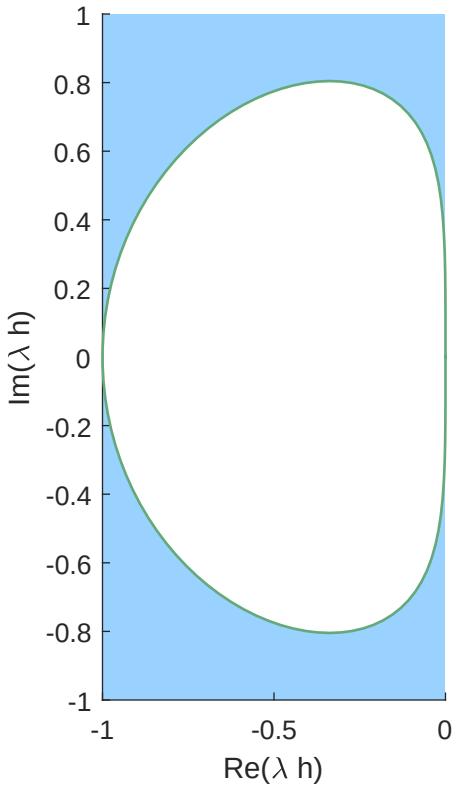
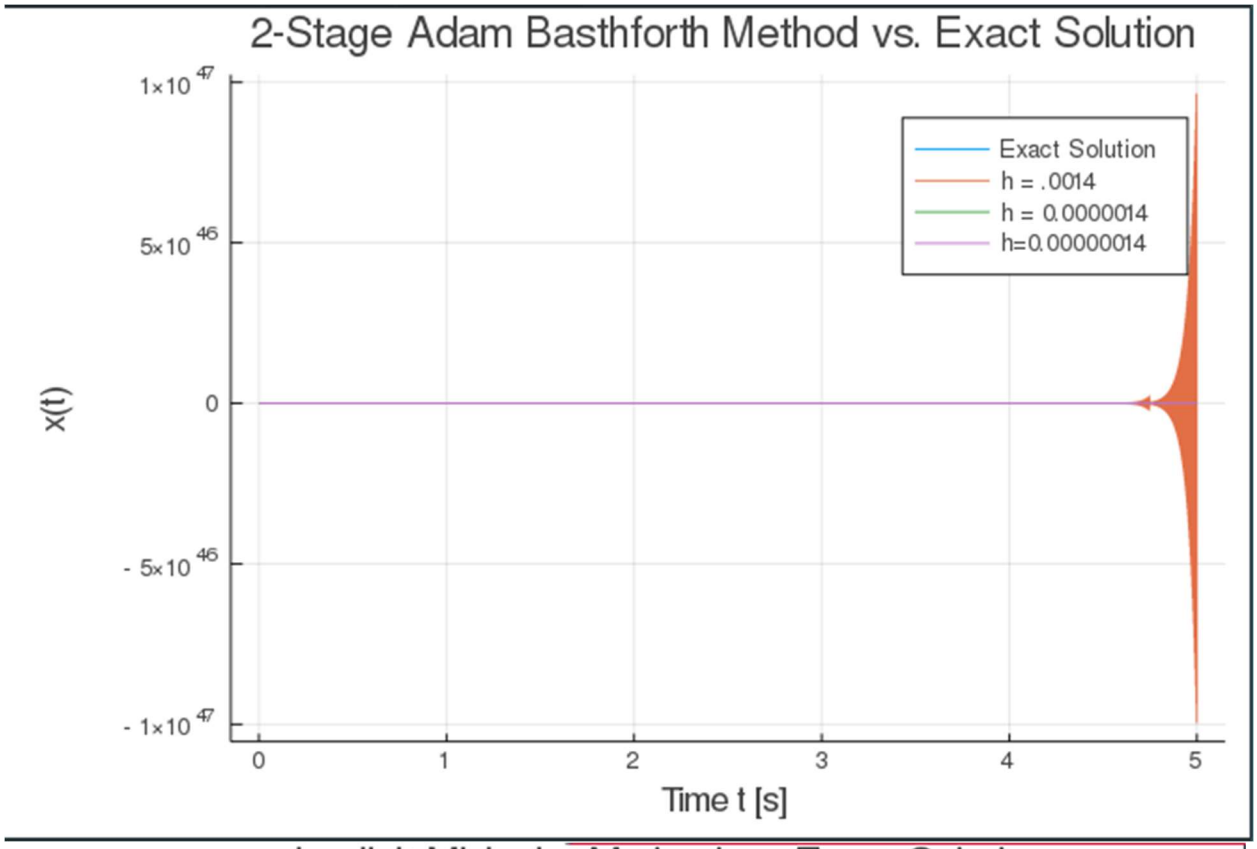
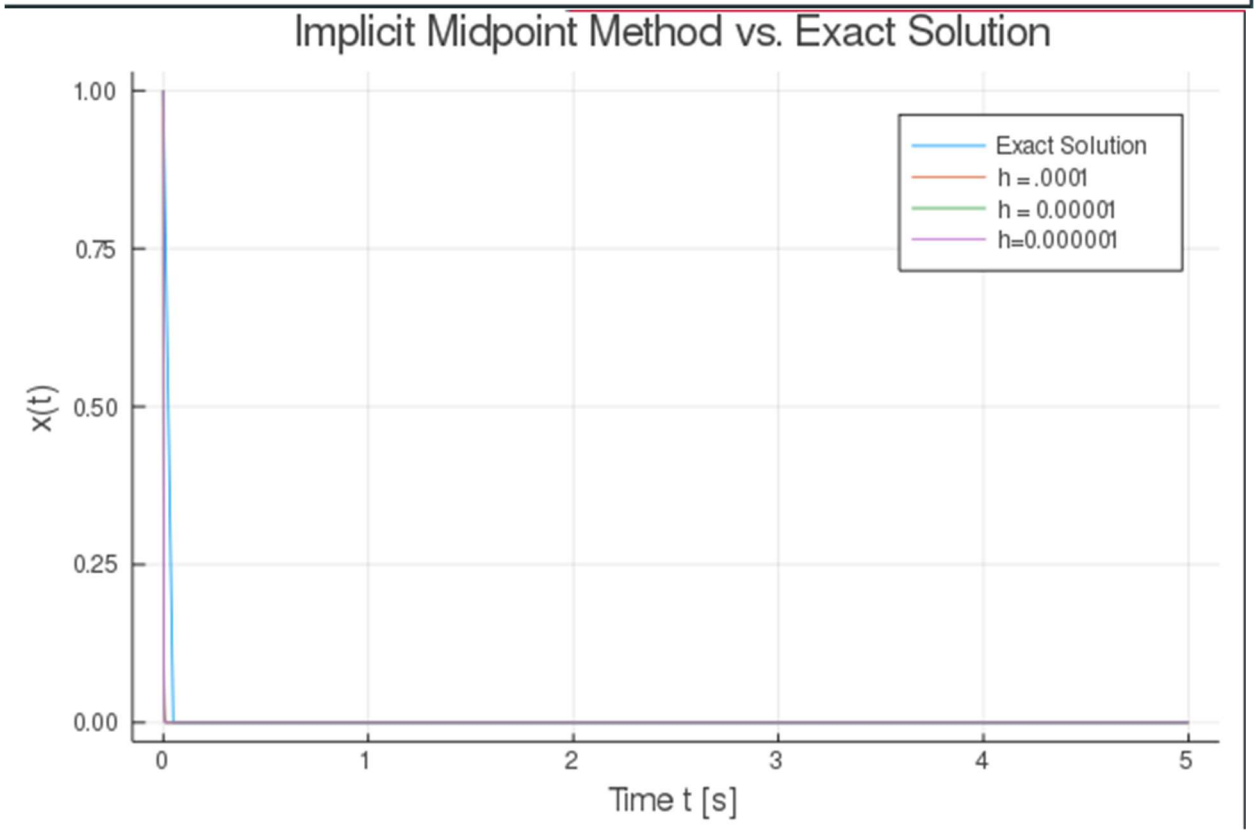

1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Im(λ h)
-1
-0.5
0
Re(λ h)

Julia Figures:





5.1 Stability

1) Trapazoidal Rule

$$x_{n+1} = x_n + \frac{h}{2}\left(f(x_n, t_n) + f(x_{n+1}, t_{n+1})\right)$$

Subs. std linear diff eqn: $\dot{x} = \lambda x$

$$x_{n+1} = x_n + \frac{h}{2}\left(\lambda x_n + \lambda x_{n+1}\right)$$

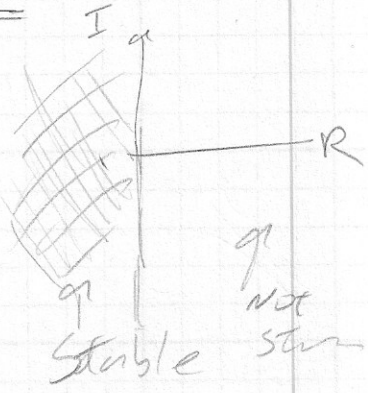
$$x_{n+1} = \left(1 + \frac{h}{2}\lambda\right) x_n + \frac{h}{2}\lambda x_{n+1}$$

$$x_{n+1} - \frac{h}{2}\lambda x_{n+1} = \left(1 + \frac{h\lambda}{2}\right) x_n$$

$$x_{n+1} = \frac{\left(1 + \frac{h}{2}\lambda\right)}{\left(1 - \frac{h}{2}\lambda\right)} x_n$$

So, $\sigma = \frac{1 + \frac{h}{2}\lambda}{1 - \frac{h}{2}\lambda}$. Recall, the method is stable if $|\sigma| < 1$.

$\therefore$ $\underline{\underline{\mathrm{Re}(h\lambda) < 0}}$



5.1 Stability

2) Runge-Kutta 2, $\alpha \in \{\frac{1}{4}, \frac{1}{2}, \frac{2}{3}\}$ (mid-point Meth., Ralston Meth.)

0	
α	α
$1-\frac{1}{2\alpha}$	$\frac{1}{2\alpha}$

$$x_{n+1} = x_n + h\left(\left(1-\frac{1}{2\alpha}\right) f(x_n, t_n) + \frac{1}{2\alpha} f\left(x_n + h\alpha f(x_n, t_n),\ t_n + \alpha h\right)\right)$$

$$x_{n+1} = x_n + h\left(\left(1-\frac{1}{2\alpha}\right)\lambda x_n + \frac{1}{2\alpha}\lambda\left(x_n + h\alpha\lambda x_n\right)\right)$$

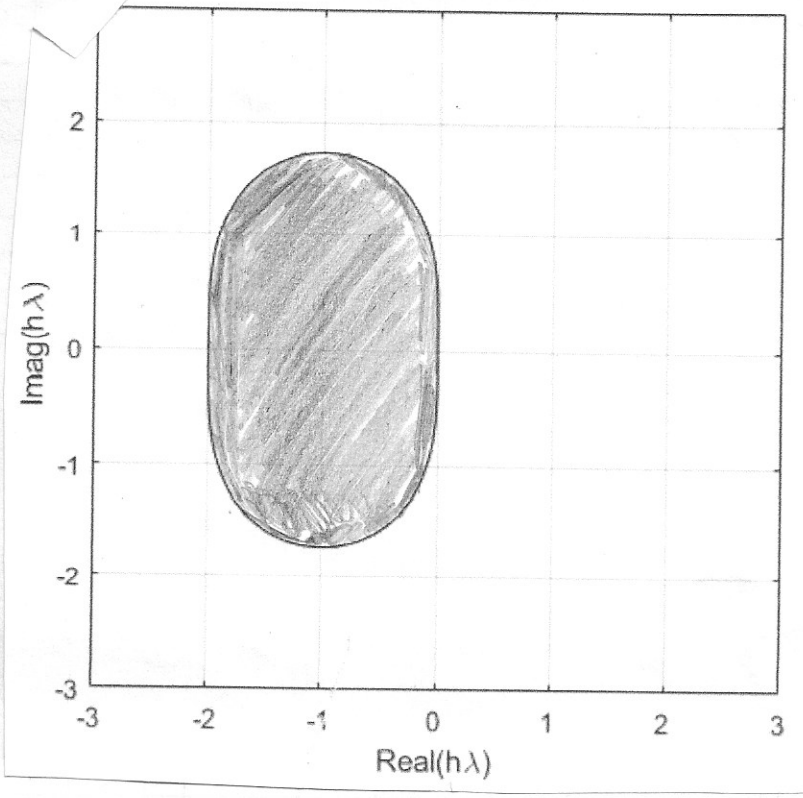
$$= 1x_n + \left(1-\frac{1}{2\alpha}\right)h\lambda x_n + \frac{1}{2\alpha}h\lambda x_n + \frac{1}{2\alpha}h\lambda(h\alpha\lambda)x_n$$

$$x_{n+1} = \left(1 + \left(1-\frac{1}{2\alpha}\right)h\lambda + \frac{1}{2\alpha}h\lambda + \frac{1}{2\alpha}h^2\lambda^2\alpha\right)x_n$$

$$|\sigma| = \left|\left(1 + h\lambda - \frac{h\lambda}{2\alpha} + \frac{h\lambda}{2\alpha} + \frac{h^2\lambda^2}{2}\right)\right| < 1 \quad < 1$$

$$\boxed{\left|1 + h\lambda + \frac{(h\lambda)^2}{2}\right| < 1}$$

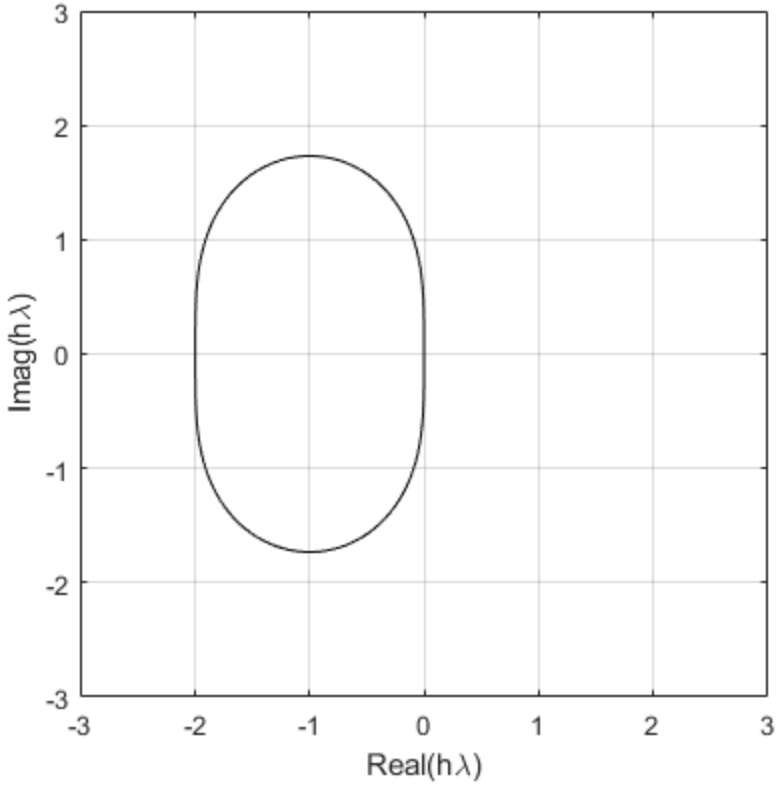
Stable in the shaded region below:



```
clear, clc, close all
```

Plot based on MIT example code Specify x range and number of points

```
x0 = -3;
x1 = 3;
Nx = 301;
% Specify y range and number of points
y0 = -3;
y1 = 3;
Ny = 301;
% Construct mesh
xv = linspace(x0,x1,Nx);
yv = linspace(y0,y1,Ny);
[x,y] = meshgrid(xv,yv);
%Calculate z
z = x + i*y;
% 2nd order Runge-Kutta growth factor
g = 1 + z + 0.5*z.^2; % Found on paper
% Calculate magnitude of g
gmag = abs(g);
% Plot contours of gmag
contour(x,y,gmag,[1 1],'k-');
axis([x0,x1,y0,y1]);
axis('square');
xlabel('Real(h\lambda)');
ylabel('Imag(h\lambda)');
grid on;
```



Published with MATLAB® R2020a

```
%% 5.1 Stability
% 5.1.1 Trapazoidal Rule Graph
clear, clc, close all
NumPoints =100;
x = linspace(-5,5,NumPoints);
y = linspace(-5,5,NumPoints);
[X,Y] = meshgrid(x,y);
Za = zeros(NumPoints,NumPoints./2); %Builds half of Z
Z = [ones(NumPoints,NumPoints./2),Za];
fig = figure();
contourf(X,Y,Z,[1 1]) %Shades the left half
legend('Stablility Region','Color',[0,200,200]/255)
```

```
%%
clear, clc, close all
%%
% Specify x range and number of points
x0 = -3;
x1 = 3;
Nx = 301;
% Specify y range and number of points
y0 = -3;
y1 = 3;
Ny = 301;
% Construct mesh
xv = linspace(x0,x1,Nx);
yv = linspace(y0,y1,Ny);
[x,y] = meshgrid(xv,yv);
%Calculate z
z = x + y*j;
% 2nd order Runge-Kutta growth factor
g = 1 + z + 0.5*z.^2; % Found on paper
% Calculate magnitude of g
gmag = abs(g);
% Plot contours of gmag
% %}
contour(x,y,gmag,[1 1],'k-'); % The [1 1] is super important here. It just draws one line↙
at the level 1
%colormap([ShadeColor;0,0,0; 1,1,1])
axis([x0,x1,y0,y1]);
axis('square');
xlabel('Real(h\lambda)');
ylabel('Imag(h\lambda)');
grid on;
%}
```

HW 5: ODEs

Paul Freihofer
ENSC-481

5.1.1

Given: A couple of methods to approximate solutions to ODEs.

Determine: The region of stability for each method if the ODE is the following:

$$\dot{x} = \lambda x$$

a) Trapezoidal Rule

$$x_{n+1} = x_n + \frac{h}{2}\left(f(t_n, x_n) + f(t_{n+1}, x_{n+1})\right)$$

b) Runge-Kutta 2 $\quad \alpha = \{1/4, 1/2, 2/3\}$

0	
α	α
	$1-\frac{1}{2\alpha}$ $\quad \frac{1}{2\alpha}$

Solution:

We start by solving for stability analytically by using a discrete method with some initial condition x_0.

$$x_i = \sigma^i x_0 \qquad (1)$$

a) $x_{n+1} = x_n + \frac{h}{2}\left(f(t_n, x_n) + f(t_{n+1}, x_{n+1})\right)$

Substituting a) with (1)

(x_n)

$$\sigma^{n+1} x_0 = \sigma^n x_0 + \frac{h}{2}\left(\lambda \cdot \sigma^n x_0 + \lambda \sigma^{n+1} x_0\right)$$

Divide x_0 out

$$\sigma^{n+1} - \frac{h}{2}\lambda\sigma^{n+1} = \sigma^n + \frac{h}{2}\lambda\sigma^n$$

Divide by σ^n

$$* \quad \frac{\sigma^{n+1}}{\sigma^n} = \sigma^{n+1-n} = \sigma \; *$$

$$\sigma\left(1 - \frac{h}{2}\lambda\right) = 1 + \frac{h}{2}\lambda$$

$$\sigma = \frac{1 + \frac{h}{2}\lambda}{1 - \frac{h}{2}\lambda} \qquad (2)$$

Now to determine stability we can say the system will be stable if:

$$|\sigma| < 1$$

From (2), $hy = z$

$$\left| \frac{1 + \frac{z}{2}}{1 - \frac{z}{2}} \right| < 1$$

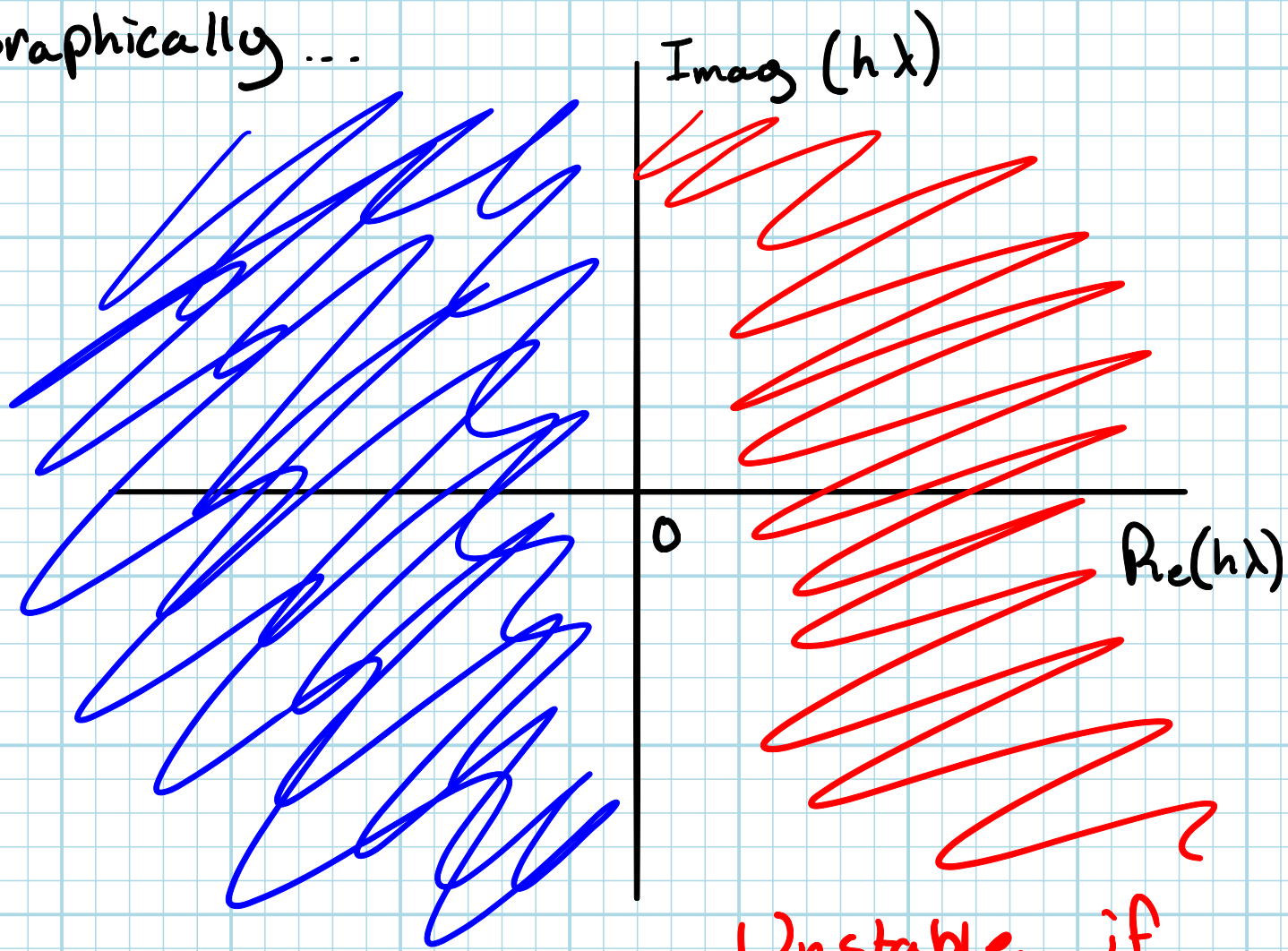
$$\left|1+\frac{z}{2}\right| < \left|1-\frac{z}{2}\right|$$

$$\left|\frac{2z}{2}\right| < |0|$$

$$|z| < 0$$

* Our system is stable if $h\lambda$ is negative. So the real part of λ must be negative.

Graphically...



Stable as long as $Re(h\lambda) < 0$

Unstable if $Re(h\lambda) > 0$

b)

0	
α	α
$1-\frac{1}{2\alpha}$	$\frac{1}{2\alpha}$

"Butcher Table for Runge Kutta 2"

We know the following from class (4/2):

0	
c_2	a_{21}
b_1	b_2

$$x_{n+1} = x_n + h(b_1 k_1 + b_2 k_2) \quad (1)$$

We know everything except k_1, k_2:

$$k_1 = f(t_n, x_n)$$

$$k_2 = f(t_n + c_2 \cdot h,\ x_n + h \cdot a_{21} \cdot k_1)$$

Let's write these out with our givens and assuming $x_i = \sigma^i x_0$

$$k_1 = \lambda \sigma^n x_0$$

$$k_2 = \lambda(\sigma^n x_0 + h \cdot \alpha \cdot \lambda \sigma^n x_0)$$

$$(1)\quad \sigma^{n+1} x_0 = \sigma^n x_0 + h\left[\left(1-\frac{1}{2\alpha}\right)(\lambda \sigma^n x_0) + \frac{\lambda}{2\alpha}\left(\sigma^n x_0 + h\alpha\lambda\sigma^n x_0\right)\right]$$

Divide by σ^n & x_0

$$\sigma = 1 + h\left[\left(1-\frac{1}{2\alpha}\right)(\lambda) + \frac{\lambda}{2\alpha}\left(1 + h\alpha\lambda\right)\right]$$

$$\sigma = 1 + h\left[\lambda - \cancel{\frac{\lambda}{2\alpha}} + \cancel{\frac{\lambda}{2\alpha}} + \frac{h\alpha\lambda^2}{2\alpha}\right]$$

$$\sigma = 1 + h\lambda + \frac{h^2\lambda^2}{2} \qquad * \; h\lambda = z$$

$$\sigma = \frac{1}{2}z^2 + z + 1$$

Stable when $|\sigma| < 1$

$$\boxed{\left|\frac{1}{2}z^2 + z + 1\right| < 1}$$

Let's Plot this:

we need to parameterize our σ equation.

$$\sigma - \frac{1}{2}z^2 - z - 1 = 0$$

Sub: $\sigma \rightarrow \cos\theta + i\sin\theta$

$z \rightarrow z_x + iz_y$

$$\cos\theta + i\sin\theta - \frac{1}{2}(z_x + iz_y)^2 - z_x - iz_y - 1 = 0$$

$$\cos\theta - \frac{1}{2}z_x^2 - z_x - 1 + \frac{1}{2}z_y^2 \ldots$$

$$+ i\sin\theta - i\frac{1}{2}z_y^2 - i\frac{1}{2}\cdot 2\, z_y z_x = 0$$

Setting real & imag. parts to Zero

Real: $\cos\theta - \frac{1}{2}z_x^2 - z_x - 1 + \frac{1}{2}z_y^2 = 0$

$$-z_x^2 - 2z_x - 2 + z_y^2 + 2\cos\theta = 0$$

$$z_y^2 = z_x^2 + 2z_x + 2 - 2\cos\theta$$

Imag: $\sin\theta - \frac{1}{2}z_y^2 - z_y z_x = 0$

$$z_y^2 = 2\sin\theta - 2z_y z_x$$

$$z_x^2 + 2z_x + 2 - 2\cos\theta = 2\sin\theta - 2z_y z_x$$

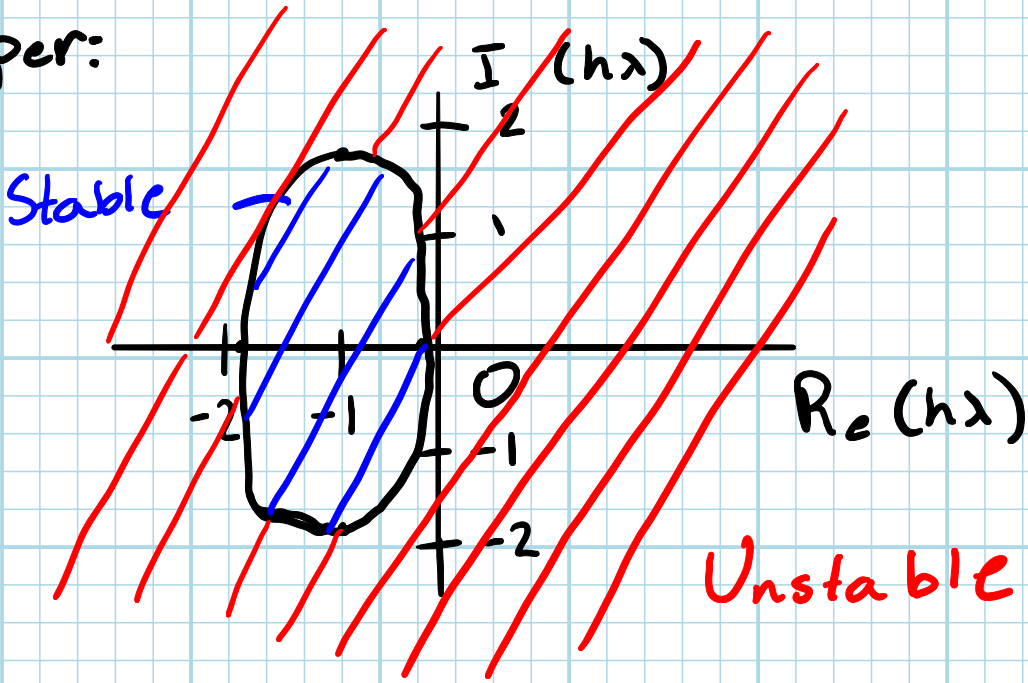
$$z_x^2 + (2 + 2z_y)z_x + (2 - 2\cos\theta - 2\sin\theta) = 0$$

Hmmm... This doesn't get clean or easy, so I'm going to jump to Joey's MIT Paper and the big idea.

<u>α doesn't matter for stability</u> with this equation $\dot{x} = \lambda x$.

$$\sigma = \frac{1}{2}z^2 + z + 1 \quad \text{*No } \alpha$$

MIT Paper:



5.2

Paul Freihoefer
ENSC 481

Given: An ODE and two cases for A.

$$\dot{x} = Ax$$

Determine: The maximum time step for two different methods.

a) Two-Stage Adams-Basthforth

b) Implicit Mid point Method

Solution:

Case 1: $A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$ Case 2: $\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -23 & -304 & -732 \end{bmatrix}$

a) 2-Stage AB

$$x_{n+2} = x_{n+1} + \frac{3}{2}h f(t_{n+1}, x_{n+1}) - \frac{1}{2}h f(t_n, x_n) \quad (1)$$

Using the I.C. Stability condition $x_i = \sigma^i x_0$

(1) $\sigma^{n+2} x_0 = \sigma^{n+1} x_0 + \frac{3}{2}h \cdot A \cdot \sigma^{n+1} x_0 - \frac{1}{2}hA\sigma^n x_0$

Divide by σ^n & x_0

$$\sigma^2 = \sigma + \frac{3}{2}hA\sigma - \frac{1}{2}hA$$

A is some nxn matrix. We can evaluate this by using the eigenvalues of A like our λ in the previous problems.

Case 1: $eig(A) = \begin{bmatrix} 0+i \\ 0-i \end{bmatrix}$

Case 2: $eig(A) = \begin{bmatrix} -731.58 \\ -0.0995 \\ -0.3160 \end{bmatrix}$

Let's Say $hA = z$

$$\sigma^2 = \sigma + \frac{3}{2} z\sigma - \frac{1}{2} z$$

$$\sigma^2 - \left(1 + \frac{3}{2} z\right)\sigma + \frac{1}{2} z = 0$$

Parameterize:

$$\sigma \to \cos\theta + i\sin\theta$$

$$z \to z_x + i z_y$$

$$(\cos\theta + i\sin\theta)^2 - \left(1 + \frac{3}{2} z_x + i\frac{3}{2} z_y\right)(\cos\theta + i\sin\theta) + \frac{1}{2}(z_x + i z_y)$$

Wolfram...

$$-\frac{3}{2} z_x \cos(\theta) + \frac{3}{2} z_y \sin(\theta) \overbrace{-\sin^2(\theta) + \cos^2(\theta)}^{\cos(2\theta)} - \cos(\theta) + \frac{z_x}{2} \ldots$$

$$+ i\left(-\frac{3}{2} z_x \sin(\theta) - \frac{3}{2} z_y \cos(\theta) - \sin(\theta) + \underbrace{2\sin(\theta)\cos(\theta)}_{\sin(2\theta)} + \frac{z_y}{2}\right) = 0$$

Real & Imag. parts set equal to zero

Real: $\cos(2\theta) - \frac{3}{2} z_x \cos(\theta) + \frac{3}{2} z_y \sin(\theta) - \cos(\theta) + \frac{z_x}{2} = 0$ (2)

Imag: $\sin(2\theta) - \frac{3}{2} z_x \sin(\theta) - \frac{3}{2} z_y \cos(\theta) - \sin(\theta) + \frac{z_y}{2} = 0$ (3)

(2) $z_x\left(\frac{3}{2}\cos(\theta)-\frac{1}{2}\right) = \cos(2\theta) + \frac{3}{2} z_y \sin(\theta) - \cos(\theta)$

$$z_x = \frac{\cos(2\theta) + \frac{3}{2} z_y \sin(\theta) - \cos(\theta)}{\frac{3}{2}\cos(\theta) - \frac{1}{2}}$$

(3) $z_y\left(\frac{3}{2}\cos(\theta)-\frac{1}{2}\right) = \sin(2\theta) - \frac{3}{2} z_x \sin(\theta) - \sin(\theta)$

$$z_y = \frac{\sin(2\theta) - \frac{3}{2} z_x \sin(\theta) - \sin(\theta)}{\frac{3}{2}\cos(\theta) - \frac{1}{2}}$$

$$z_x = \frac{\cos(2\theta) + \frac{3}{2}\left(\frac{\sin(2\theta) - \frac{3}{2} z_x \sin(\theta) - \sin(\theta)}{\frac{3}{2}\cos(\theta) - \frac{1}{2}}\right)\sin(\theta) - \cos(\theta)}{\frac{3}{2}\cos(\theta) - \frac{1}{2}}$$

$$z_x\left(\frac{3}{2}\cos(\theta)-\frac{1}{2}\right)^2 = \left(\frac{3}{2}\cos(\theta)-\frac{1}{2}\right)\left(\cos(2\theta) - \cos(\theta)\right) \ldots$$
$$+ \frac{3}{2}\sin(2\theta)\sin(\theta) - \frac{9}{4} z_x \sin^2(\theta) - \frac{3}{2}\sin^2(\theta)$$

$$z_x\left[\left(\frac{3}{2}\cos(\theta)-\frac{1}{2}\right)^2 + \frac{9}{4}\sin^2(\theta)\right] = \left(\frac{3}{2}\cos(\theta)-\frac{1}{2}\right)\left(\cos(2\theta)-\cos(\theta)\right) \ldots$$
$$+ \frac{3}{2}\sin(2\theta)\sin(\theta) - \frac{3}{2}\sin^2(\theta)$$

$$z_x = \frac{\left(\frac{3}{2}\cos(\theta)-\frac{1}{2}\right)\left(\cos(2\theta)-\cos(\theta)\right) + \frac{3}{2}\sin(2\theta)\sin(\theta) - \frac{3}{2}\sin^2(\theta)}{\left(\frac{3}{2}\cos(\theta)-\frac{1}{2}\right)^2 + \frac{9}{4}\sin^2(\theta)}$$

Now to MATLAB...

b) Implicit Midpoint Method

Wikipedia:

$$x_{n+1} = x_n + h\, f\left(t_n + \frac{h}{2}, \frac{1}{2}(x_n + x_{n+1})\right)$$

I.C, Stability $\quad x_i = \sigma^i x_0$

$$\sigma^{n+1} x_0 = \sigma^n x_0 + hA\left(\frac{1}{2}\sigma^n x_0 + \frac{1}{2}\sigma^{n+1} x_0\right)$$

Divide by x_0 & σ^n, $\quad hA = z$

$$\sigma = 1 + z\left(\frac{1}{2} + \frac{1}{2}\sigma\right)$$

$$\sigma - \frac{1}{2}z\sigma = 1 + \frac{1}{2}z$$

$$\sigma\left(1 - \frac{1}{2}z\right) = 1 + \frac{1}{2}z$$

$$\sigma = \frac{1 + \frac{1}{2}z}{1 - \frac{1}{2}z}$$

$|\sigma| < 1$ $\quad$ * Stability Condition *

$$\left|\frac{1 + \frac{1}{2}z}{1 - \frac{1}{2}z}\right| < 1$$

$$1 + \frac{1}{2}z < 1 - \frac{1}{2}z$$

$$\boxed{z < 0}$$

Hmmm.... Looks familiar

Let's go to MATLAB

PDFreihoefer / Computational-Dynamics

Branch: master ▾ Computational-Dynamics / Problem_5_2.jl Find file Copy path

PDFreihoefer Updating HW6 Problems 646f74b 2 minutes ago

1 contributor

66 lines (52 sloc) 2.34 KB Raw Blame History

```
using Pkg
Pkg.activate(".")

using Plots
using LinearAlgebra
##Loading our Trapezoidal Rule:
include("C:/Users/paulf/Documents/GitHub/Computational-Dynamics/TrapRule.jl")
using .TrapRule
##Loading our Runge Kutta Methods:
include("C:/Users/paulf/Documents/GitHub/Computational-Dynamics/AB2.jl")
using .AB2
##Setting up the Problem
f(x,t) = λ*x #The function we are working with
λ = -731 #Let's define some value to start
x0 = 0.5 #Initial guess
#h = 1 Failed to converge
#h = 0.01 Failed to converge
h1 = 0.0014
h2 = 0.0000014
h3 = 0.00000014
tf = 5 #Final time

```

```
##Plotting the exact equation
x_exact(t) = exp.(λ*t)*x0

plot( x_exact, 0, tf, label="Exact Solution")
##Plotting the Implicit Midpoint Method Results
#tRuleX, tRuleT = tRule(f, tf, h1, x0)

#plot!( tRuleT, tRuleX, label="h = .0001")

#tRuleX, tRuleT = tRule(f, tf, h2, x0)

#plot!( tRuleT, tRuleX, label="h = 0.00001")

#tRuleX, tRuleT = tRule(f, tf, h3, x0)

#plot!( tRuleT, tRuleX, label="h=0.000001",
        #xlabel="Time t [s]", ylabel="x(t)", title="Implicit Midpoint Method vs. Exact Solution")
##Plotting AB2 Results
AB2X, AB2T = ab2(f, tf, h1, x0)

plot!( AB2T, AB2X, label="h = .0014")

AB2X, AB2T = ab2(f, tf, h2, x0)

plot!( AB2T, AB2X, label="h = 0.0000014")

AB2X, AB2T = ab2(f, tf, h3, x0)

plot!( AB2T, AB2X, label="h=0.00000014",
        xlabel="Time t [s]", ylabel="x(t)", title="2-Stage Adam Basthforth Method vs. Exact Solution")
##Conclusion
#AB2:
#In this case the max step size shows that the system converges but eventually becomes unstable and explodes as
#time goes on. This is a good example showing that the smaller the step size the more accurate the method is and
#that using something close to the max step size is not optimal for accuracy but potentially optimal for cost in
#this case.
```

```
59
60 #IM:
61 #The system will never converge at some eigenvalues if you actually use the max time step and max out the
62 #iterations at some value. We could increase the max iterations but it will become more and more expensive
63 #to run the method and we will need to increase the range of time we are looking at as well. It makes more sense
64 #to decrease the final time and decrease the step size to lower the overall cost of running the equation. It
65 #should also be noted that with using a very large step size the equation the method isn't as good at approximating
66 #the exact solution. We can see in this plot that the
```

PDFreihoefer / Computational-Dynamics

Branch: master ▾ Computational-Dynamics / TrapRule.jl Find file Copy

PDFreihoefer Updating HW5 Problems e187752 23 minutes

1 contributor

55 lines (41 sloc) 1.54 KB Raw Blame History

```
1  module TrapRule
2
3  using LinearAlgebra
4  export tRule
5
6  #This function is based off of the Trapezoidal Rule
7  function tRule(f,tf,h,x0; tol = 0.001, iterMax = 200)
8      time = 0:h:tf #In this case we will denote the time step with "h"
9      n = length(time)
10     p = length(x0) #Allows us to consider multiple initial conditions
11     x = zeros(n,p) #In case we are working with matrices we want to establish x before our for loop
12
13     #Initial Condition
14     x[1,:] .= x0
15
16     #First off we need to turn our equation from implicit to explicit (x(i+1) appears twice in the equation)
17     #x[i+1,:] = x[i,:] + h/2*(f(x[i,:],time[i,:])+f(x[i+1,:],time[i+1,:]))
18
19     #Euler Step
20     fn = f(x[1,:],time[1])
21     y = x[1,:]+h*fn
22
```

```
    #B-Euler for 1 Step to get x[i+1,:]
    flag = 0
    iter = 0
    while flag == 0
           iter += 1
           y = x[1,:] + h*f(y, time[2])

           residual = norm( y - x[1,:] - h*f(y, time[2]) )

           if residual <= tol
               flag = 1
           elseif iter >= iterMax
               flag = -1
               error("Error: failed to converge")
           end
    end

    #We now have y = x[2,:] so we will plug it into the trapezoidal rule equation to make the equation explicit

    for i = 1:n-1
        #for j = 1:p, We would want this in case the initial condition has multiple values and we would change the equation below
        x[i+1,:] = x[i,:] + h/2*(f(x[i,:],time[i,:])+f(y,time[i+1,:]))
        #end
        #Reassigning y with the new value we calculated in the last step
        y = x[i+1,:]

    end

    return x, time

end

end
```

PDFreihoefer / Computational-Dynamics

Branch: master ▾ Computational-Dynamics / AB2.jl Find file Copy

PDFreihoefer Updating HW6 Problems 646f74b 4 minutes

1 contributor

33 lines (24 sloc) 634 Bytes Raw Blame History

```
1  module AB2
2
3  using LinearAlgebra
4
5  export ab2
6
7  #Credit to: Dr. Fitzgerald during class on 4/2/20
8  function ab2(f, tf, h, x0)
9      time = 0:h:tf
10     n = length(time)
11     p = length(x0)
12
13     x = zeros(n,p)
14
15     # Initial Condition
16     x[1,:] .= x0
17
18     # 1 Euler Step (this is cheap and possibly bad)
19     fn = f(x[1,:], time[1])
20     x[2,:] = x[1,:] + h*fn
21
22     # AB2 the rest
```

```
23      for i = 2:n-1
24          fn_m1 = fn # f(x[i-1,:], time[i-1])
25          fn = f(x[i,:], time[i])
26          #x[i+1,:] = x[i,:] + h/2*( 3*f(x[i,:], time[i]) - f(x[i-1,:], time[i-1])  )
27          x[i+1,:] = x[i,:] + h/2*( 3*fn - fn_m1 )
28      end
29
30      return x, time
31  end
32
33  end
```

```
%%Clearing the Workspace
clear all; close all; clc

%%Establishing our cases
A1 = [0 1; -1 0];
A2 = [0 1 0; 0 0 1; -23 -304 -732];

%%Finding the eigenvalues
eig1 = eig(A1);
eig2= eig(A2);

%%Plotting our region of stability for the 2-Stage Adam Basthforth Method
theta = linspace(0,2*pi(),100);
zx = ((3/2*cos(theta)-1/2).*(cos(2.*theta)-cos(theta))+3/2*sin(2*theta).*sin(theta)-3↙
/2*sin(theta).^2)./((3/2*cos(theta)-1/2).^2+9/4*sin(theta).^2);
zy = (sin(2*theta)-3/2*zx.*sin(theta)-sin(theta))./(3/2*cos(theta)-1/2);

fig = figure();
ax = axes('Parent', fig);
hold(ax,'on');
ax.DataAspectRatio = [1,1,1];
ax.Color = [153, 210, 255]/255;
patch(ax, zx, zy, 'w', ...
    'EdgeColor', [101, 169, 123]/255, 'LineWidth',1 );
xlabel(ax, 'Re(\lambda h)');
ylabel(ax, 'Im(\lambda h)');

%%Minimums and Maximums for AB2
zxEx = [min(zx),max(zx)];
zyEx = [min(zy)*i,max(zy)*i];

%%Case 1: AB2
%Our Eigenvalues are -i & i but the min and max on the imaginary axis are
%the same (-0.8045i, 0.8045i). This means that either eigenvalue can be
%used to calculate the max step size. We will choose the positive
%eigenvalue and solve for the max h possible if z = h*lambda(the
%eigenvalue).

hAB1 = zyEx(2)/eig1(1);

%%Case 2: AB2
%The eigenvalues here are all different but rather than testing each
%individual one we can tell which will limit our h value the most. If the
%eigenvalue is small than h can be large but if the eigenvalue is large
%than h must be small to stay within the region of stability. This means
%that we only need to consider the largest eigenvalue of -731.58 to
%calculate the max h possible for the system to stay stable.
hAB2 = zxEx(1)/eig2(1)


%%Plotting the implicit midpoint method stablity region
```

```
NumPoints =100;
x = linspace(-5,5,NumPoints);
y = linspace(-5,5,NumPoints);
[X,Y] = meshgrid(x,y);
Za = zeros(NumPoints,NumPoints./2); %Builds half of Z
Z = [ones(NumPoints,NumPoints./2),Za];
fig = figure();
contourf(X,Y,Z,[1 1]) %Shades the left half

%%Case 1: IM
%Turns out the implicit midpoint method is the same as the trapezoidal rule
%in terms of the region of stability. For case 1 this is a case where the
%eigenvalues are only complex and the real part is equal to zero. The
%method is only stable for real values of h*lambda less than zero so there
%is no time step where this case is stable using this method.
hIM1 = 0;

%%Case 2: IM
%Each eigen value is negative which means for any time step greater than
%zero the method will be stable. This means the max time step is
%essentially infinite. You could choose any time step and the method will
%be stable.
hIM2 = inf;

%%Displaying the results for both methods
fprintf('2-Stage Adam Basthforth Method\n')
fprintf('For Case 1 the max step size h is: %0.4fs\n', hAB1)
fprintf('For Case 2 the max step size h is : %0.4fs\n', hAB2)

fprintf('\nImplicit Midpoint Method\n')
fprintf('For Case 1 there is no max step size because any step size will not be stable.↙
\n')
fprintf('For Case 2 any step size can be chosen and the system will remain stable. In↙
other words the max step size could be infinite (although this is impractical).\n')
```

MATLAB Output 5_2:

2-Stage Adam Basthforth Method

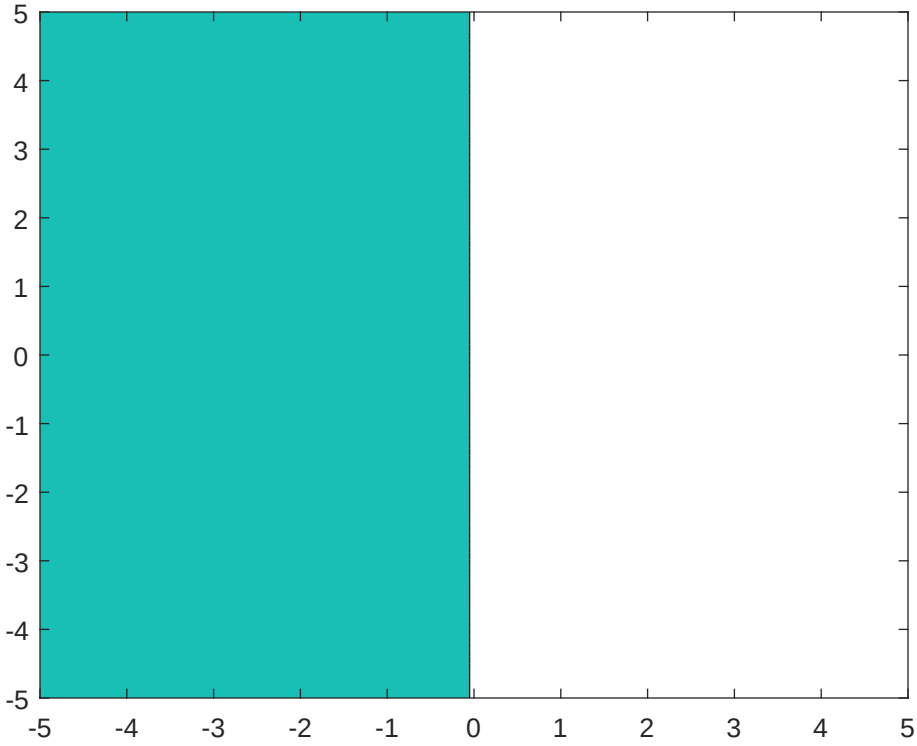
For Case 1 the max step size h is: 0.8045s

For Case 2 the max step size h is : 0.0014s

Implicit Midpoint Method

For Case 1 there is no max step size because any step size will not be stable.

For Case 2 any step size can be chosen and the system will remain stable. In other words the max step size could be infinite (although this is impractical).


5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5