**ME526 Homework 4**

Andrew Alferman

Dan Magee

1. The leap frog scheme was computed using the attached Python code. The governing equations of the scheme as well as the other two schemes investigated in this problem were solved by hand on the attached paper. The output of the code is seen below. With a step size of 0.15, the leap frog scheme exhibited a higher frequency than the exact solution, and subsequently can be seen to shift towards the left side of the graph, however the scheme exhibited no amplitude error as demonstrated by extending the range of the plot to very large numbers along the x axis.



Error vs. time step proved problematic to solve because smaller time steps converged to a solution that was out of phase with the exact solution. One such plot is as follows: 

In order to accurately determine the order of accuracy, the error would have to be computed about a point at which both the exact solution and the leapfrog scheme converge, or the phase shift would have to be taken into account..

(b) The problem was solved using a central differencing scheme (trapezoidal) using a step size of 0.15, as seen in the above figure. The trapezoidal method had a slightly lower frequency than the exact solution, and subsequently can be seen to shift to the right of the graph slightly.

(c) This problem was solved by hand on the attached sheet.

(d) The mixed scheme exhibits the same weakness as the leapfrog method in that it has a higher frequency than the exact solution and therefore shifts the graph to the right. Additionally, the scheme appears to have an amplitude error which has manifested in a growing solution, as seen in the figure below. The mixed scheme is not stable with a time step size of 0.15, though the rate at which it blows up is relatively small. The scheme may have benefits in that it may be faster to compute than the central differencing formula, which requires a matrix to be transposed at ever time step in the attached Python code. Additionally, the scheme does not have as much phase error as a normal leapfrog method.



The code below was used to solve problem 1:

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Sun Nov 13 21:45:56 2016

@authors: Andrew Alferman and Dan Magee

"""

import numpy as np

import matplotlib.pyplot as plt

def leapfrog(Ym1, Y, A, dt):

return Ym1 + 2\*dt\*np.dot(A, Y)

def trapezoidal(Y, A, dt):

Im = np.linalg.inv(np.eye(2) - dt\*0.5\*A)

Ip = np.eye(2) + dt\*0.5\*A

return np.dot(np.dot(Im, Ip), Y)

def fwdeuler(Y, A, dt):

return np.dot((np.eye(2) + dt\*A), Y)

def exactsoln(omega, t):

return np.cos(omega \* t) + 0.25\*np.sin(omega \* t)

# Set up all of the initial conditions and problem parameters

omega = 4.0

dt = 0.15

t = 0.0

y0 = 1.0

y0p = 0.0

Y0 = [[y0], [y0p]]

Y0 = np.array(Y0)

# Set up the Jacobian matrix for this particular problem

A = [[0.0, 1.0], [-1.0 \* omega\*\*2, 0.0]]

A = np.array(A)

# Set up lists to log all of the values for plotting later

Yll, Ylt, Ylm, tl, ye, errorl = [1], [1], [], [0], [1], [0]

# Run a forward Euler approximation for the first timestep only

Yl, Yt, Ym, Ylm1, Yold = Y0, Y0, Y0, Y0, Y0

Yl = fwdeuler(Yl, A, dt)

Yt = trapezoidal(Yt, A, dt)

t += dt

while t <= 6:

ye.append(exactsoln(omega, t))

Yll.append(Yl[0])

Ylt.append(Yt[0])

tl.append(t)

Ylnew = leapfrog(Ylm1, Yl, A, dt)

Yt = trapezoidal(Yt, A, dt)

Ylm1 = Yl

Yl = Ylnew

t += dt

plt.figure(1)

plt.plot(tl, Yll, label='Leapfrog')

plt.plot(tl, Ylt, label='Trapezoidal')

plt.plot(tl, ye, label='Exact Solution')

plt.legend(bbox\_to\_anchor=(1, 1), loc=2)

plt.xlabel('Time')

plt.ylabel('Y Value')

plt.title('Comparison of Schemes (dt = 0.15)')

Ymold = 0

t = 0

selector = 0

while t <= 6:

if selector != 2:

Ym = trapezoidal(Ym, A, dt)

selector += 1

else:

Ym = leapfrog(Ymm1, Ym, A, dt)

selector = 0

Ylm.append(Ym[0])

Ymm1 = Ymold

Ymold = Ym

t += dt

plt.figure(2)

plt.plot(tl, Ylm, label='Mixed Scheme')

plt.plot(tl, ye, label='Exact Solution')

plt.legend(bbox\_to\_anchor=(1, 1), loc=2)

plt.xlabel('Time')

plt.ylabel('Y Value')

plt.title('Mixed Scheme vs. Exact Solution')

plt.show()