MECE 6397 HW-8 John Anderson

1.) What is exact solution to $\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \quad -\infty < x < \infty$

with $u(x,0) = \exp(-|x|)$?

General solution $\rightarrow F(x-ct)$

Check Initial Solution
at $t=0 \rightarrow F(x) = \exp(-|x|)$

Thus $F(x-ct) = \exp(-|x-ct|)$

$$\boxed{u(x,t) = \exp(-|x-ct|)}$$

2. Determine whether the scheme is consistent w/ original eq.

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} + c\,\frac{u_j^{n+1} - u_{j-1}^n}{\Delta x} = 0$$

Find corresponding effective equation

$$\rightarrow \frac{u_j^{n+1} - u_j^n}{\Delta t} \approx \frac{\partial u}{\partial t} + \frac{1}{2}\Delta t \frac{\partial^2 u}{\partial t^2} + \dots$$

$$\rightarrow u_j^{n+1} = \cancel{u_j^n} + \Delta t \frac{\partial u}{\partial t} + \frac{1}{2}\Delta t^2 \frac{\partial^2 u}{\partial t^2} + \dots$$

$$\rightarrow u_{j-1}^n = \cancel{u_j^n} - \Delta x \frac{\partial u}{\partial x^2} + \frac{1}{2}\Delta x^2 \frac{\partial^2 u}{\partial x^2} + \dots$$

$$\rightarrow \frac{\partial u}{\partial t} + \frac{1}{2}\Delta t \frac{\partial^2 u}{\partial t^2} + \frac{c}{\Delta x}\left(\Delta t \frac{\partial u}{\partial t} + \Delta x \frac{\partial u}{\partial x} + \frac{1}{2}\Delta t^2 \frac{\partial^2 u}{\partial t^2} - \frac{1}{2}\Delta x^2 \frac{\partial^2 u}{\partial x^2}\right) =$$

$$\frac{\partial u}{\partial t} + \frac{c\Delta t}{\Delta x}\frac{\partial u}{\partial t} + \frac{1}{2}\Delta t \frac{\partial^2 u}{\partial t^2} + \frac{c\Delta t^2}{2\Delta x}\frac{\partial^2 u}{\partial t^2} + \frac{c\,\partial u}{\partial x} - \frac{c\Delta x}{2}\frac{\partial^2 u}{\partial x^2} = 0$$

$$\boxed{\left(1 + \frac{c\Delta t}{\Delta x}\right)\frac{\partial u}{\partial t} + \left(\frac{\Delta t}{2} + \frac{c\Delta t^2}{2\Delta x}\right)\frac{\partial^2 u}{\partial t^2} + \frac{c\,\partial u}{\partial x} - \frac{c\Delta x}{2}\frac{\partial^2 u}{\partial x^2} = 0}$$

Effective Equation

as $\Delta t, \Delta x \to 0$ we recover $\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0$

But for finite $\Delta t, \Delta x$ we have different equation

If we neglect the Δt^2 term.

$$\left(1 + \frac{c\Delta t}{\Delta x}\right)\frac{\partial u}{\partial t} + \frac{\Delta t}{2}\frac{\partial^2 u}{\partial t^2} + \frac{c\,\partial u}{\partial x} - \frac{c\Delta x}{2}\frac{\partial^2 u}{\partial x^2} = 0 + \dots$$

$$\left(1 + \frac{c\Delta t}{\Delta x}\right)\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} + \frac{\Delta t}{2}\frac{\partial^2 u}{\partial t^2} - \frac{c\Delta x}{2}\frac{\partial^2 u}{\partial x^2} = 0$$

$$\left(1 + c\frac{\Delta t}{\Delta x}\right)\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} + \mathcal{O}(\Delta t) + \mathcal{O}(\Delta x) = 0$$

Scheme is <u>not</u> <u>consistent</u> for finite Δx, Δt.

Especially if $\Delta x \to 0$ faster than Δt.

John Anderson
MECE 6397
Homework 8

8.3

The Analytical solution was found simply by applying the general solution f(x-ct) to the initial conditions.

$u(x,t) = (x-t)*(1-(x-t))$ 0<x-t<1
and
$u(x,t) = 0$ 1<=x-t

For all images in this report the total interval length is 4*pi. Figure 1 shows the analytical solution and the backward scheme for a courant number close to one. The solution is for t=3 seconds. We have to zoom in multiple times to see the difference between the solutions. The backward scheme is very close but is slightly less due to the dampening.

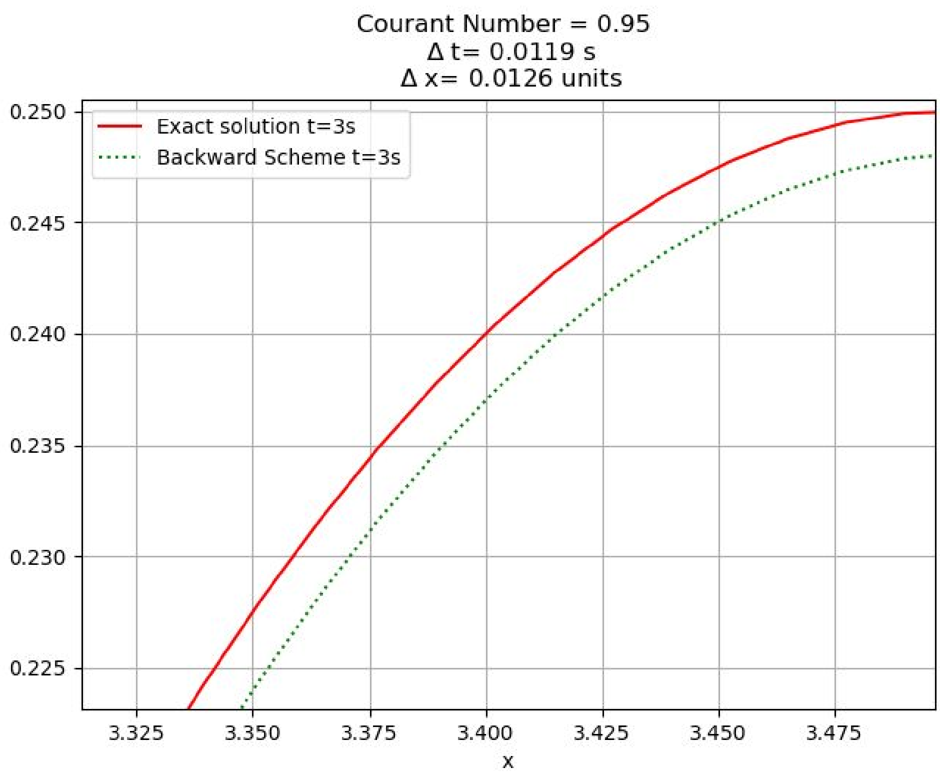

Fig. 1 Backward scheme and exact solution at t=3s. For a Courant number close to 1 very little dampening was observed.

Using different combinations of delta t and delta x led to different dampening. For small Courant numbers the dampening was significant and the wave was also wider than the analytical solution. This is different from what I initially expected when starting this assignment. I originally thought only the amplitude of the wave would decrease but this was not the case. We can see an example of this in Figure 2.

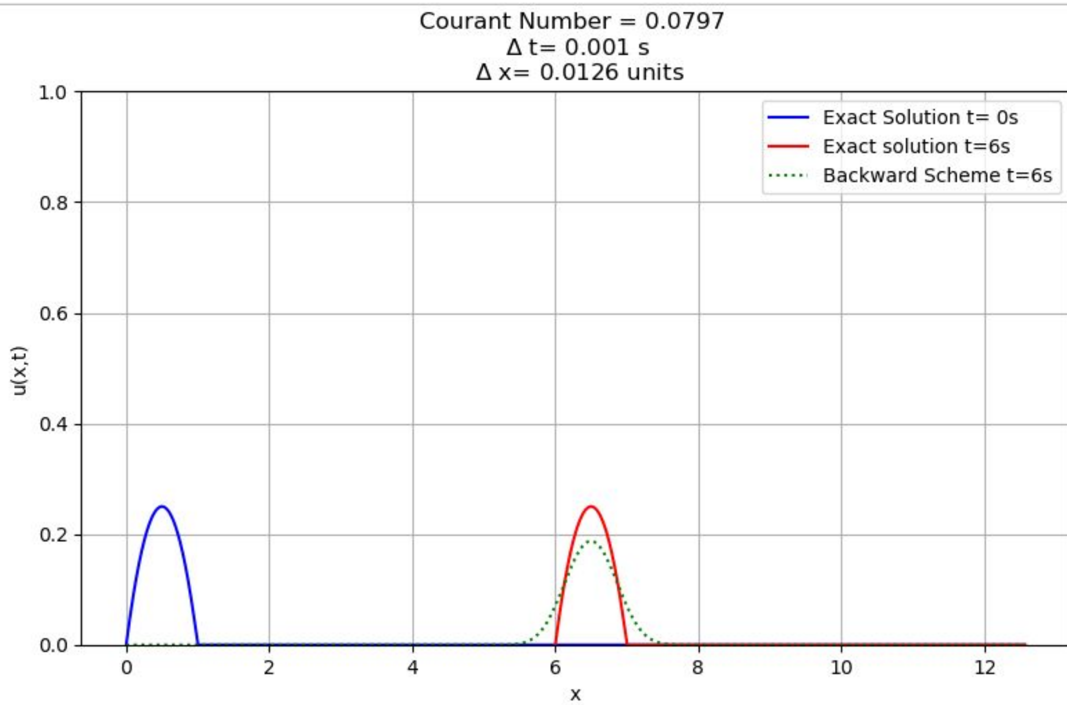

Fig. 2 Smaller courant numbers led to more dramatic dempaning. The wave is shorter and stretched wider.

;

From simplifying the backward scheme we have

$$u_j^{n+1} = (1 - C)u_j^n + Cu^n_{j-1}$$

When the Courant number is greater than 1 our solution is no longer stable and is unacceptable. We can see this in Figures 3 and 4 where the solution just falls apart and is meaningless. The solution oscillates between a very larger number in python with a very high frequency.

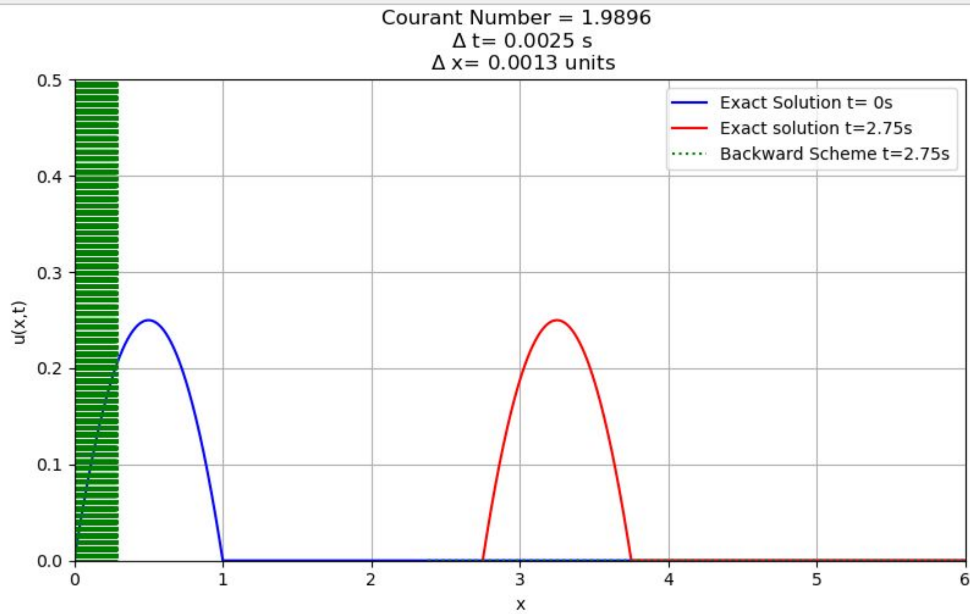

Fig.3 The Backward scheme is unstable for Courant numbers greater than 1. The solution fails to advance and the solution grows without bound.

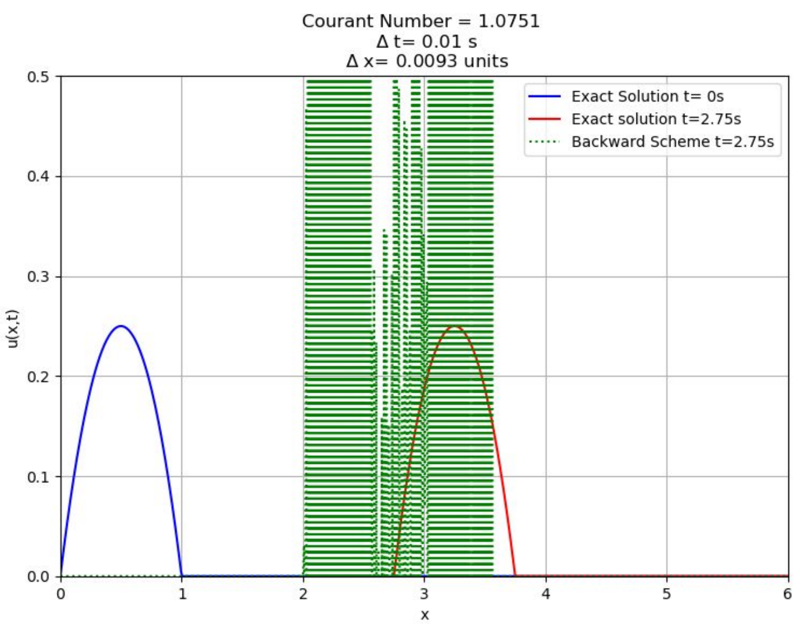

Fig 4 For a Courant number slightly over 1 the solution is still unstable but propagated with the analytical solution.

John Anderson
MECE 6397
Homework 8

From simplifying the forward scheme we have

$$u_j^{n+1} = (1 + C)u_j^n - Cu^n_{j+1}$$

which for a right traveling wave will always be unstable, even if the Courant number is less than 1. We can see this in figures 5 and 6 where the solution grows without bound for courant numbers both greater than 1 and less than 1. This forward scheme will always be unstable for a right traveling wave because it is not using the upwind node. If we had a left traveling wave and used the forward scheme we would than have conditional stability, with C courant number greater than 1 being unstable and less than 1 stable.

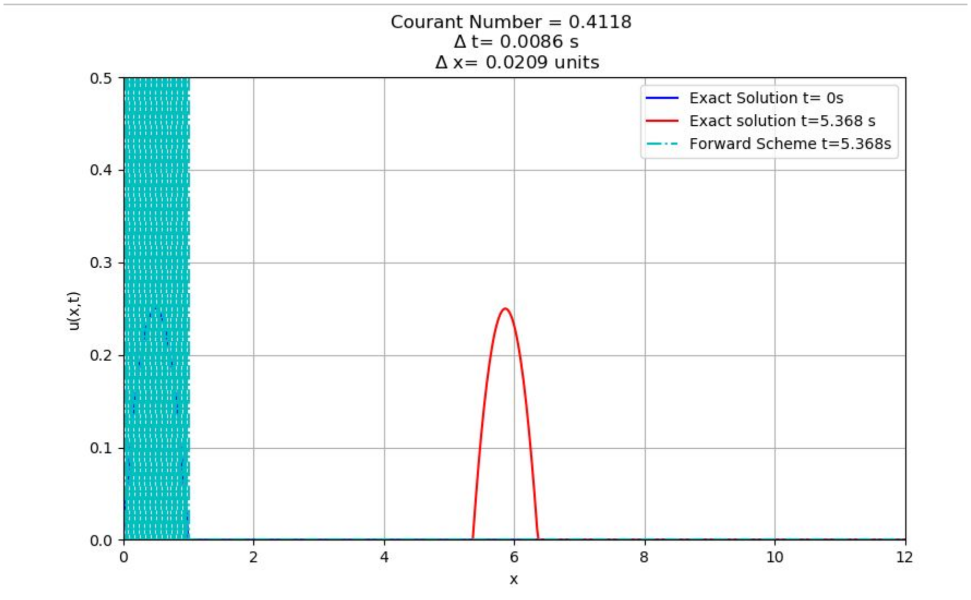

Fig.5 Unstable forward scheme even with a Courant number less than 1.

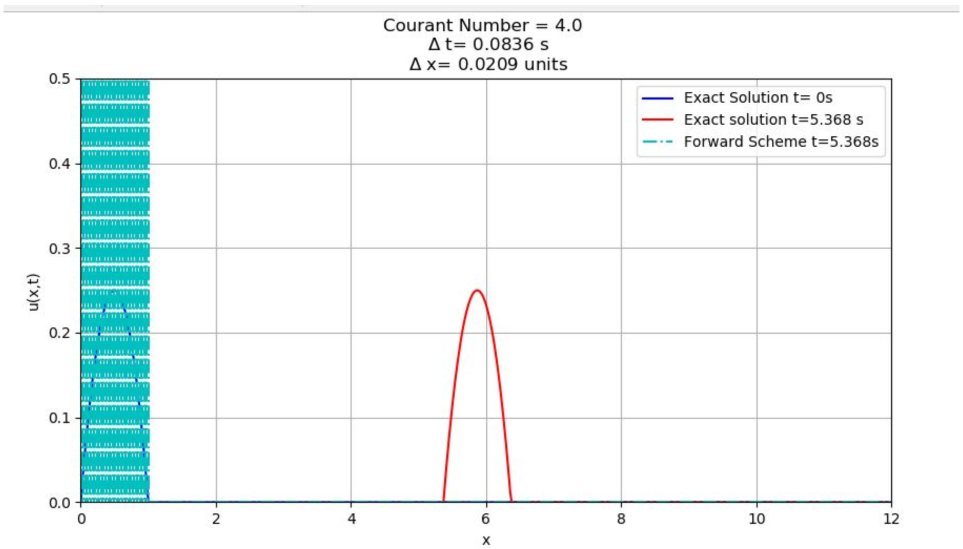

Fig 6. Unstable forward scheme for Courant number greater than

John Anderson
MECE 6397
Homework 8

Python Code

```
# -*- coding: utf-8 -*-
"""
Created on Tue Dec  1 20:13:34

@author: johna
"""
# MECE 6397, SciComp, HW8 Question 3
#Solve two different schemes for a 1-D wave equation, c=1
#Backward and Forward
#Up to me to decide how far to extend x
#\https://github.com/jeander5/-MECE_6397_HW8_COMP/

#imports
import math
import numpy as np
import matplotlib.pyplot as plt

def DIF(L ,N):
  """discretizes an interval with N interior points"""
#Inputs are interval length number of interior points
#Returns the discretize domain and the interval spacing
#Includes endpoints
  h = L/(N+1)
  x = np.linspace(0, L, N+2)
  return(x, h)

def IC(x):
  """Returns the initial condition for this Homework Problem"""
#note this also includes the boundary condition u(x=0,t)=0
  lenx=len(x)
  func_vals=np.zeros(lenx)
  k=0
  while x[k]<1:
    func_vals[k]=x[k]*(1-x[k])
    k=k+1
  return (func_vals)
#that really didnt need to be a function

def u_exact_func(x,t):
  """Returns the Analytical solution for this Assignment"""
#Inputs are the whole x domain and a single time t.
```

```
    lenx=len(x)
    func_vals=np.zeros(lenx)
    k=0
    while x[k]-t<1:
        func_vals[k] = - x[k]*x[k] + x[k] + 2*x[k]*t - t*t - t
        k=k+1
    return (func_vals)

def Backward_Scheme_func(u_n,C):
    """Returns the u values at the next time step using the backward scheme """
#Inputs are the u values at the previous, nth, time step, for a previously discretized domain
#Define constant phi outside of the loop.
    phi=1-C
    N=len(u_n)
    func_vals=np.ones(N)
#this line is for a boundary condition of u(0,t0)
    func_vals[0]=u_n[0]
    for j in range(1,N):
        func_vals[j]=u_n[j]*phi+C*u_n[j-1]
    return (func_vals)


#Advance Solution Forward_Backward Scheme
def ASF_BS_func(T,dt, u_n):
    """Moves the Solution forward T/dt number of times using the backward scheme """
#inputs for now are total Time, dt, and most up to date u vals
    N=round(T/dt)
#this still uses the dx assigned outside of the function
    C=c*dt/dx
    for j in range(N):
        u_n=Backward_Scheme_func(u_n,C)
    return(u_n)

def Forward_Scheme_func(u_n,C,BC):
    """Returns the u values at the next time step using the forward scheme """
#Inputs are the u values at the previous, nth, time step, for a previously discretized domain
#And also the right boundary condition, u(x=L,t), from the exact function
#Define constant phi outside of the loop.
    phi=1+C
    N=len(u_n)
    func_vals=np.ones(N)
#this line is for a boundary condition of u(0,t0)
    func_vals[0]=u_n[0]
    for j in range(1,N-1):
```

```
        func_vals[j]=u_n[j]*phi-C*u_n[j+1]
    func_vals[-1]=u_n[j+1]*phi-C*BC
    # Last equation different, for the right side boundary condition
    return (func_vals)

def ASF_FS_func(T,dt, u_n):
    """Moves the Solution forward T/dt number of times using the forward scheme """
#inputs for now are total Time, dt, and most up to date u vals
    N=round(T/dt)
#this still uses the dx assigned outside of the function
    C=c*dt/dx
#getting Boundary condition from the exact function
    BC=u_exact_func(x,T)[-1]
    for j in range(N):
        u_n=Forward_Scheme_func(u_n,C,BC)
    return(u_n)

#Call the functions as needed for the report

#let length of the interval be 4 pi
L=4*math.pi
#Number of internal points
N=600
#wave speed, is simply 1 here
c=1
#calling the discretizing the interval function
x, dx=DIF(L, N)
#I will define a dt here as well,
dt = dx*4
C = c*dt/dx
Uo=IC(x)

#Plotting, uncomment and modify as needed

#u_exact=u_exact_func(x,0)
#fig2, ax2 = plt.subplots()
#plt.grid(1)
#plt.plot(x,u_exact,'b')
#u_exact=u_exact_func(x,5.368)
##u_7=ASF_BS_func(3,dt,Uo)
#u_6=ASF_FS_func(5.368,dt,Uo)
#plt.plot(x,u_exact,'r')
#plt.plot(x,u_6,'-.c')
#ax2.set(ylim=(0, 0.5))
```

```
#ax2.set(xlim=(0, 12))
##u_7=ASF_BS_func(5,dt,Uo)
##plt.plot(x,u_7,'m--')
##I will just append a legend string as I go along
#legend_string=([])
#legend_string=(['Exact Solution t= 0s'])
#legend_string.append('Exact solution t=5.368 s')
#legend_string.append('Forward Scheme t=5.368s')
##legend_string.append('Backward Scheme t=5.75s')
##legend_string.append('Backward Scheme t=5s')
##this legend string reports the external courant number,
##so if I input a new delta t to the advancing function the legend string will be wrong
#ax2.title.set_text('Courant Number = %s \n $\Delta$ t= %s s \n $\Delta$ x= %s units'
#                  %(round(C,4),round(dt,4),round(dx,4)))
#ax2.legend(legend_string)
#plt.xlabel('x')
#plt.ylabel('u(x,t)')
```