HW4_Solutions

February 16, 2024

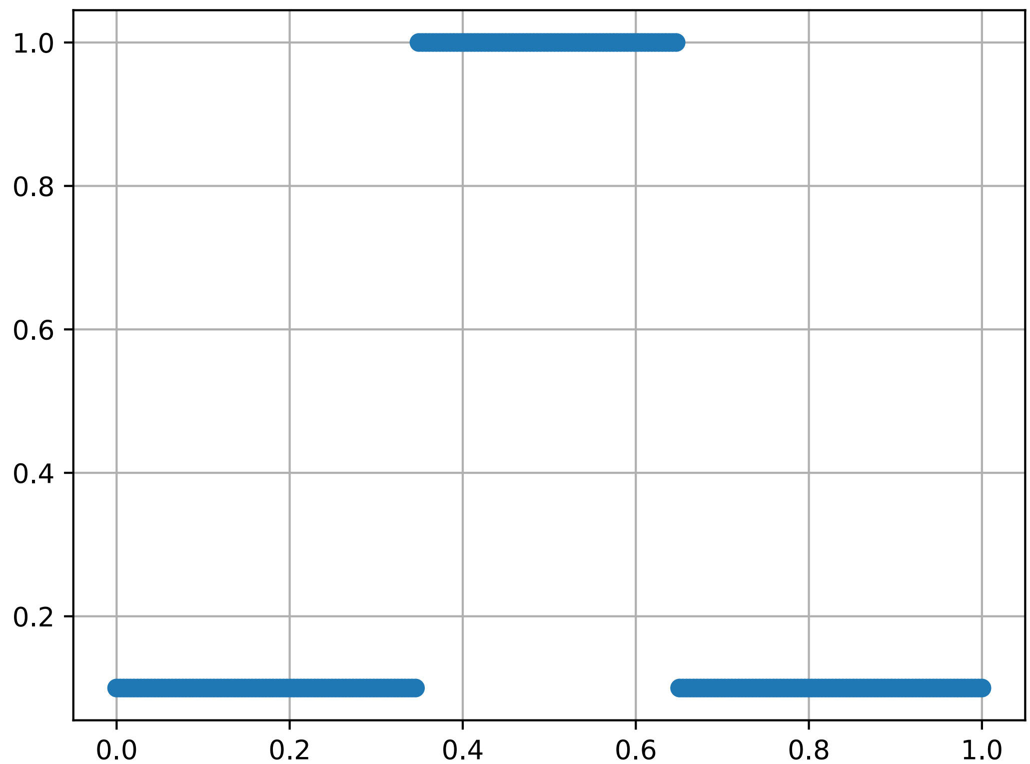
1 PHYS 6260 - Homework 4 Solutions

[1]:
```
import numpy as np
%matplotlib inline
%config InlineBackend.figure_format = 'svg'
import matplotlib.pyplot as plt
import matplotlib.animation as animation
plt.rcParams['animation.html'] = 'html5' # this is used to display animations␣
↪in jupyter notebooks
```

1.1 Problem 1

1.1.1 One-dimensional advection

[2]:
```
n = 250
L = 1.0
xedge = np.array([0.35, 0.65])
xi = (xedge * n).astype('int')
dx = L/(n-1)
x = np.linspace(0,L,n)
#u0 = np.sin(2.0*np.pi*x)
u0 = np.zeros(n) + 0.1
u0[xi[0]:xi[1]] = 1.0
plt.plot(x,u0,'o')
plt.grid()
```



1.2 Part (a): Forward in Time, Central in Space (FTCS)

Using a forward Euler method for the time derivative and a central discretization for the spatial derivative, the FTCS scheme results in

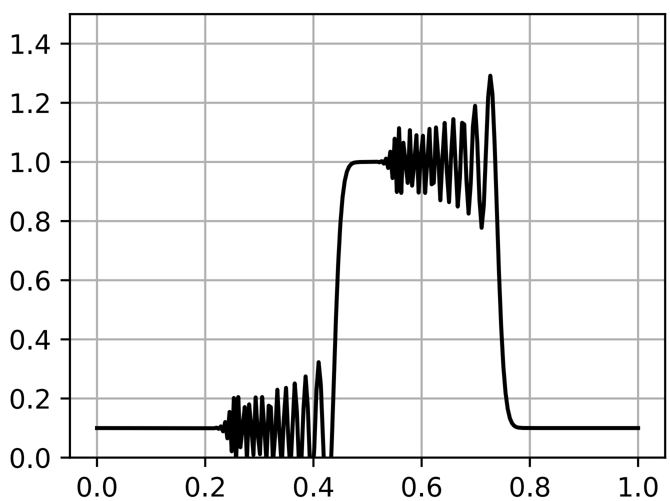
$$q_i^{n+1} = q_i^n + c\Delta t \frac{q_{i+1}^n - q_{i-1}^n}{2\Delta x} \tag{1}$$

[3]:
```
# the following implementation does not use ghost cells
dt = 1e-4   # s
tend = 0.1 # s
c = 1.0 # m/s - wave speed
cfl = c*dt/2.0/dx

sol = []
sol.append(u0)

t = 0.0
while t < tend:
    un = sol[-1]
    unew = un.copy()
    unew[1:-1] = un[1:-1] - cfl * (un[2:] - un[:-2])
```

```
    unew[-1] = un[-1] - cfl*(un[1] - un[-2]) # compute last point on the right␣
↪using periodicity
    unew[0] = unew[-1] # set periodic boundary on the left
    sol.append(unew)
    t += dt
```

[4]:
```
ims = []
fig = plt.figure(figsize=[4,3])
plt.grid()

i = 0
for solution in sol:
    if (i%10==0): # output frequency for frames
        im = plt.plot(x,solution,'k-',animated=True)
        plt.ylim(0,1.5)
        ims.append(im)
    i+=1
ani = animation.ArtistAnimation(fig, ims, interval=35, blit=True,
                                repeat_delay=1000)
# ani.save('ftcs.mp4')
ani
```

[4]: <matplotlib.animation.ArtistAnimation at 0x7fc3779cbfa0>



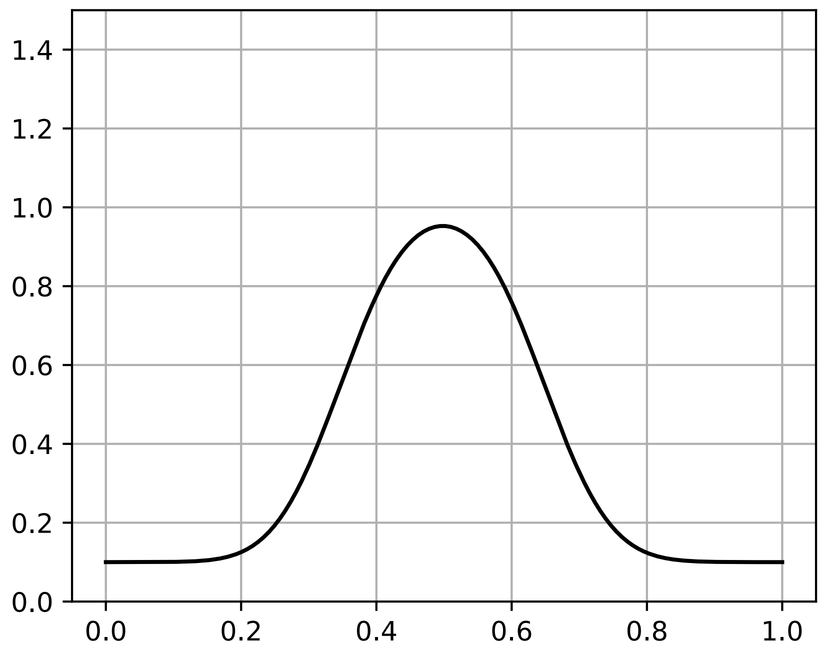
1.3 Part (b): Forward in Time, Backward (Upwind) in Space (FTUS)

We have shown in class that the previous FTCS scheme is unconditionally unstable. Here we will implement the upwind scheme in space

[5]:
```python
tend = 2.0 # s
c = 1.0 # m/s - wave speed
dt = 1e-3
cfl = c * dt / dx

sol = []
sol.append(u0)

t = 0.0
while t < tend:
    un = sol[-1]
    unew = un.copy()
    unew[1:-1] = un[1:-1] - cfl * (un[1:-1] - un[:-2])
    unew[-1] = un[-1] - cfl*(un[-1] - un[-2]) # compute last point on the right␣
 ↪using periodicity
    unew[0] = unew[-1] # set periodic boundary on the left
    sol.append(unew)
    t += dt
```

[6]:
```python
ims = []
fig = plt.figure(figsize=[5,4],dpi = 150)
plt.grid()

i = 0
for solution in sol:
    if (i%10==0):
        im = plt.plot(x,solution,'k-',animated=True)
        plt.ylim(0,1.5)
        ims.append(im)
    i+=1
ani = animation.ArtistAnimation(fig, ims, interval=35, blit=True,
                                repeat_delay=1000)
# ani.save('ftus.mp4')
ani
```

[6]:
```
<matplotlib.animation.ArtistAnimation at 0x7fc3757f59c0>
```



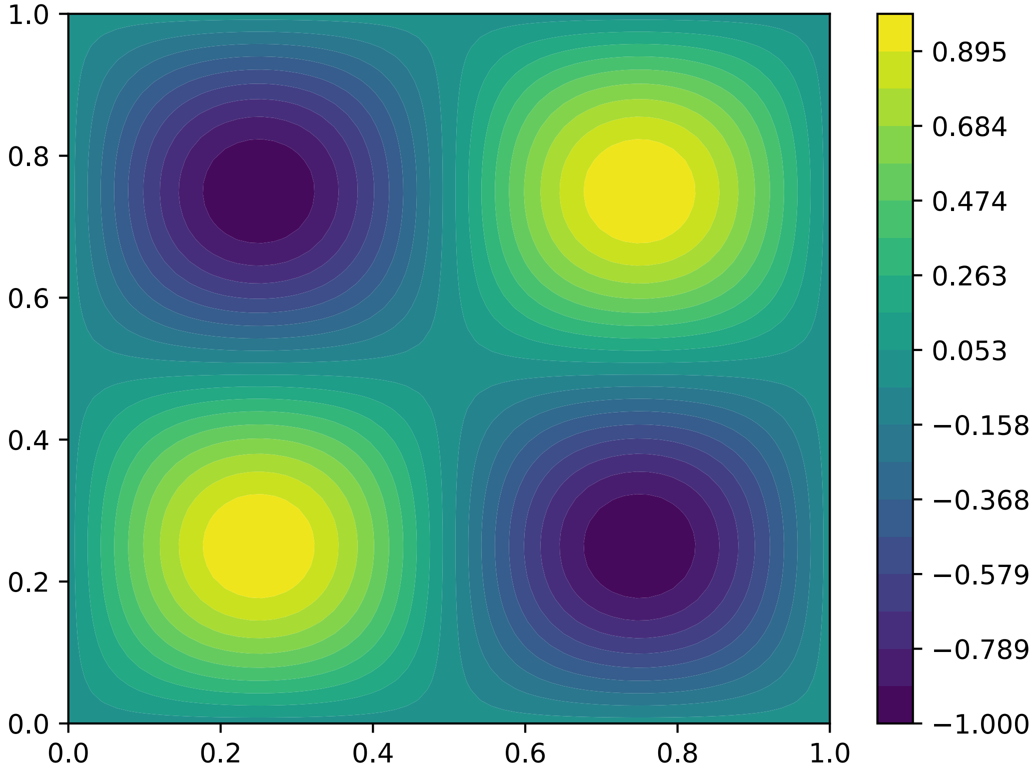
1.4 Problem 2

1.4.1 Two-dimensional advection

[7]:
```
nx = 64
ny = 64
Lx = 1.0
Ly = 1.0
x = np.linspace(0,Lx,nx)
y = np.linspace(0,Ly,ny)
dx = Lx/(nx-1)
dy = Ly/(ny-1)
# create a grid of coordinates
xx,yy = np.meshgrid(x,y)
```

[14]:
```
 0 = 2.0*np.pi
u0 = lambda x,y : np.sin( 0*x) * np.sin( 0*y)
# plot the initial condition
levs = np.linspace(-1,1,20)
plt.contourf(xx,yy,u0(xx,yy),levels=levs,vmax=1.0,vmin=-1.0)
plt.colorbar()
```

[14]: <matplotlib.colorbar.Colorbar at 0x7fc36e5b3010>

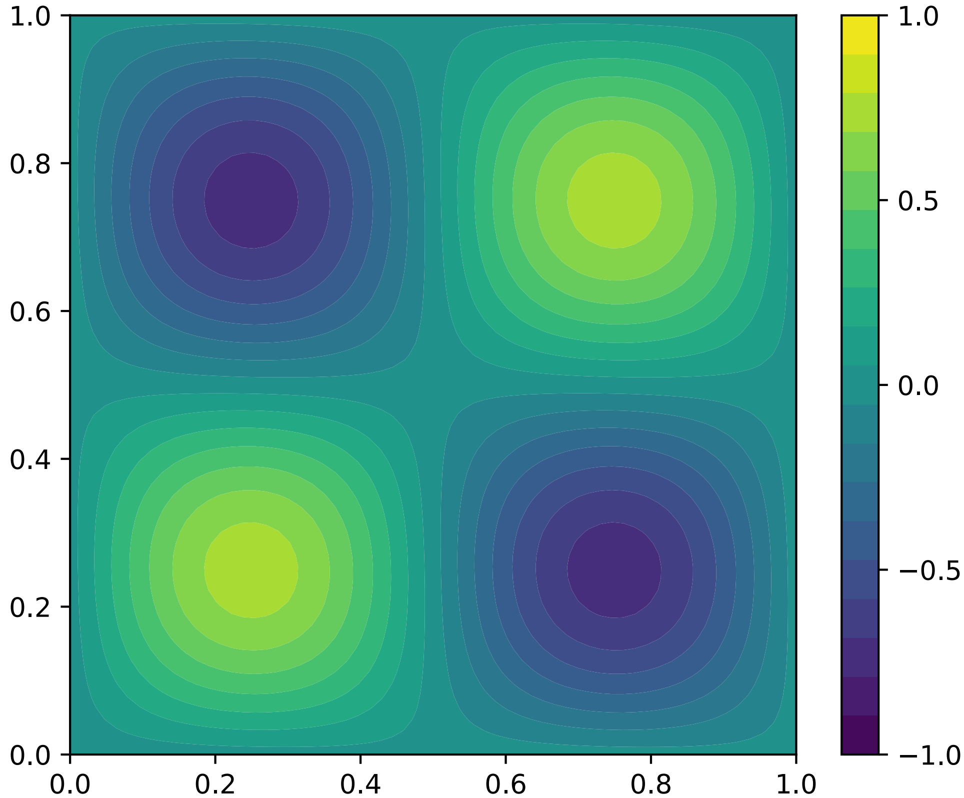


[9]:
```
cx = 1.0
cy = 1.0

dt = 0.001
tend = 0.5 #s
t = 0

cflx = cx * dt/dx
cfly = cy * dt/dy

# setup the initial condition
sol = []
u = np.zeros([ny+2,nx+2]) # we will ghost cells to simplify the implementation
    ↪of periodic BCs
u[1:-1, 1:-1] = u0(xx,yy)
# set periodic boundaries
u[:,0] = u[:,-3] #x-minus face
u[:,-1] = u[:,2] #x-plus face
u[0,:] = u[-3,:] #y-minus face
u[-1,:] = u[2,:] #y-plus face
sol.append(u)
```

[10]:
```
while t < tend:
    un = sol[-1]
    unew = un.copy()
    unew[1:-1,1:-1] = un[1:-1,1:-1] - cflx * (un[1:-1,1:-1] - un[1:-1,:-2]) -␣
↪cfly * (un[1:-1,1:-1] - un[:-2,1:-1])
    # set periodic boundaries
    unew[:,0] = unew[:,-3] #x-minus face
    unew[:,-1] = unew[:,2] #x-plus face
    unew[0,:] = unew[-3,:] #y-minus face
    unew[-1,:] = unew[2,:] #y-plus face

    sol.append(unew)
    t += dt
```

[12]:
```
fig = plt.figure(figsize=(6.1,5),facecolor='w')
ims = []
levs = np.linspace(-1,1,20)
i = 0
t = 0.0
for solution in sol:
    if (i%10==0):
        im = plt.contourf(xx,yy,solution[1:-1,1:
↪-1],cmap="viridis",levels=levs,vmax=1.0,vmin=-1.0)
        ims.append(im.collections)
    i+=1
    t += dt

cbar = plt.colorbar()
plt.clim(-1,1)
cbar.set_ticks(np.linspace(-1,1,5))

ani = animation.ArtistAnimation(fig, ims, interval=35,␣
↪blit=True,repeat_delay=1000)

ani
```

[12]:
```
<matplotlib.animation.ArtistAnimation at 0x7fc36e75a3e0>
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
1.0
0.5
0.0
−0.5
−1.0

```
[ ]:
```