```
"""
Author   : Abraham Flores
File     : p737.py
Language : Python 3.6
Created  : 3/29/2018
Edited   : 5/8/2018

San Digeo State University
MTH 693b : Computational Partial Differential Equations

Strikwerda 7.3.7 : Alternating Direction Implicit Methods

Guidance From ...
https://pdfs.semanticscholar.org/31ac/59e11ef214220bb56919549425347b8e6b88.pdf

Peaceman-Rachford Alogrithim:

    Heat equation:
        u_t = b_1*u_x + b_2*u_y

    x in [0,1]
    y in [0,1]
    t in [0,1]

    Intial Value:
        -Exact Solution

    Boundaries:
        -Exact Solution

    Exact Solution:
        b1 = 2
        b2 = 1
        u(t,x,y) = exp(1.68*t)*sin[1.2*(x-y)]*cosh[x+2y]


    dx = dy = dt = 1/10,1/20,1/40

    *Demonstrate second order Accuracy

    *Things to Note
    -- numpy mesh grid broke everything
    -- Accuracy seems worse than previous methods
    -- Fairly fast
"""

import os,glob
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.sparse import diags

#Generates intial value function
def exact(x,y,t):
    return np.exp(1.68*t)*np.sin(1.2*(x-y))*np.cosh(x+2*y)

#Contour Plot
def surf_plot(x,y,U,time,title,fileLoc):
    sns.set(font_scale = 2.0)
    sns.set_style("darkgrid", {"axes.facecolor": ".9"})

    fig,ax = plt.subplots()
    fig.set_size_inches(14.4,9)
    xlocs = []
    xlabel = []
    for i in range(0,len(x),int(len(x)/5)):
        xlocs.append(i+1)
        xlabel.append(x[i])

    ylocs = []
    ylabel = []
    for i in range(0,len(y),int(len(y)/5)):
        ylocs.append(i+1)
```

```
        ylabel.append(y[i])


    plt.xticks(xlocs,xlabel,rotation=45)
    plt.yticks(ylocs,ylabel,rotation=45)
    # Plot the contour
    plt.pcolor(U,vmin=-20.0,vmax=10.0)#vmin=np.min(np.min((U))),vmax=np.max(np.max(U
)))
    ax.annotate("t = "+ str(round(time,3)),xy=(0,0) ,xytext=(1,1),color="w")
    #legend
    clb = plt.colorbar()
    clb.set_label(r'$U(t,X,Y)$', labelpad=40, rotation=270)
    plt.xlabel('X (spatial)')
    plt.ylabel('Y (spatial)')
    plt.title(title)

    plt.savefig(fileLoc+'.png')
    plt.close()

"""
Makes a gif given a name and delay for each image in ms

--Assumes the images are in the figures directory
"""
def makeGif(gifName,delay):
    os.chdir('Figures')
    #Create txt file for gif command
    fileList = glob.glob('*.png') #star grabs everything,
    fileList.sort()
    #writes txt file
    file = open('FileList.txt', 'w')
    for item in fileList:
        file.write("%s\n" % item)
    file.close()

    os.system('convert -delay ' + str(delay) + ' @FileList.txt ' + gifName + '.gif')
    os.system('del FileList.txt')
    os.system('del *.png')
    os.chdir('..')

def ExactGIF(h,mu):
    #generate array of intial values at t = 0
    x = np.arange(0,1+h,h)
    y = np.arange(0,1+h,h)

    N = len(x)
    title = "7.3.7: EXACT: h: " +str(round(h,4)) + ", mu: " +str(mu)
    u = np.zeros((N,N))
    dt = mu*h**2
    iters = 0
    time = 0.0
    while time < 1.0:
        time = iters*dt
        for i in range(N):
            for j in range(N):
                u[i][j] = exact(x[i], y[j],time)
        #plot
        str_time = '0'*(5-len(str(iters)))+str(iters)
        outFile = "Figures\exact" + str_time
        surf_plot(x,y,u,time,title,outFile)
        iters+= 1

    #makeGif
    makeGif("Exact_Solution_h_"+str(h)+"_Mu_"+str(mu),10)

def Peaceman_Rachford(h,mu):
    bx = 2.0
    by = 1.0

    x = np.arange(0, 1.0+h, h) # x grid
    y = np.arange(0, 1.0+h, h) # y grid
    N = len(x)
    u = np.zeros((N,N)) # solution array
```

```
    HALF = u.copy() # half step
    FULL = u.copy() # full step

    #Matrix Coefficents
    Cx = bx*mu
    Cy = by*mu

    X_SWEEP = np.array([-Cx*np.ones(N-1),(2+2*Cx)*np.ones(N),-Cx*np.ones(N-1)])
    Y_SWEEP = np.array([-Cy*np.ones(N-1),(2+2*Cy)*np.ones(N),-Cy*np.ones(N-1)])
    offset = [-1,0,1]#Location of each diagonal

    X_SWEEP = diags(X_SWEEP,offset).toarray()#Generate Matrix
    Y_SWEEP = diags(Y_SWEEP,offset).toarray()#Generate Matrix

    #EMBED BOUNDARY CONDTITIONS
    X_SWEEP[0] *= 0.0
    X_SWEEP[-1] *= 0.0
    Y_SWEEP[0] *= 0.0
    Y_SWEEP[-1] *= 0.0

    X_SWEEP[0][0] = 1.0
    X_SWEEP[-1][-1] = 1.0
    Y_SWEEP[0][0] = 1.0
    Y_SWEEP[-1][-1] = 1.0

    # set initial condition:
    for i in range(N):
        for j in range(N):
            u[i][j] = exact(x[i], y[j],0.0)

    time = 0.0

#    title = \
#  "7.3.7: Peaceman-Rachford: h: " +str(round(h,4)) + ", mu: " +str(mu)
#    outFile = "Figures\PR00000"
#    surf_plot(x,y,u,time,title,outFile)
    iters=1
    L2_NORM = []
    INF_NORM = []

    dt = mu*h**2
    while time < 1.0:
        time = dt*(iters - 0.5)

        # X-direction sweep:

        # Left boundary
        HALF[:,0] = exact(x[i], 0.0, time)

        # solve linear tridiagonal system for all internal columns j:
        for j in range(1,N-1):
            #Inner points for column j:
            Y_RHS = 2*u[:,j] + Cy*(u[:,j-1] - 2*u[:,j] + u[:,j+1])

            #Lower Boundary
            Y_RHS[0] = exact(0.0, y[j], time)

            #Upper Boundary:
            Y_RHS[-1] = exact(1.0, y[j], time)

            # solve linear system: Fill Cols
            HALF[:,j] = np.linalg.tensorsolve(X_SWEEP,Y_RHS)

        # Right boundary
        HALF[:,-1] = exact(x[i], 1.0, time)

        time = iters*dt
        # Y-direction sweep:

        # Lower boundary
        FULL[0] = exact(0.0, y, time)

        # solve linear tridiagonal system for all internal rows i:
```

```
        for i in range(1,N-1):
            #inner points for row i:
            X_RHS = 2*HALF[i] + Cx*(HALF[i-1] - 2*HALF[i] + HALF[i+1])

            #Left boundary:
            X_RHS[0] = exact(x[i], 0.0, time)

            #Right boundary
            X_RHS[-1] = exact(x[i], 1.0, time)

            # solve linear system: Fill Rows
            FULL[i] = np.linalg.tensorsolve(Y_SWEEP,X_RHS)

        # Upper boundary
        FULL[-1] = exact(1.0, y, time)

        #UPDATE
        u, FULL = FULL, u

        #ERROR
        err = np.zeros((N,N))
        for i in range(N):
            for j in range(N):
                err[i][j] = abs(exact(x[i], y[j],time) -u[i][j])

        L2_NORM.append(np.sqrt(sum(sum(err*err))))
        INF_NORM.append(np.max(np.max(err)))

        #PLOT
#        str_time = '0'*(5-len(str(iters)))+str(iters)
#        outFile = "Figures\PR" + str_time
#        surf_plot(x,y,u,time,title,outFile)

        #Step Forward
        iters+=1

    return L2_NORM, INF_NORM

def best_fit(X, Y):

    xbar = sum(X)/len(X)
    ybar = sum(Y)/len(Y)
    n = len(X) # or len(Y)

    numer = sum([xi*yi for xi,yi in zip(X, Y)]) - n * xbar * ybar
    denum = sum([xi**2 for xi in X]) - n * xbar**2

    b = numer / denum
    a = ybar - b * xbar

    return a, b

def plot_norm(h,inf_norm,L2_norm):
    sns.set(font_scale = 1.65)
    sns.set_style("darkgrid", {"axes.facecolor": ".9"})
    fig,ax = plt.subplots()
    fig.set_size_inches(9,6)
    plt.scatter(h,inf_norm,linewidth=3.0,color="r",label=r'$-Log_{10}$[INFINITY NORM
]')
    plt.scatter(h,L2_norm,linewidth=3.0,color="b",label=r'$-Log_{10}$[L2 NORM]')

    plt.xlabel(r'$-Log_{10}$[dx]')
    plt.ylabel(r'$-Log_{10}$|ERROR|')

    a_inf, b_inf = best_fit(h, inf_norm)
    yfit = [a_inf + b_inf * xi for xi in h]
    plt.plot(h, yfit,color="k",label="(INF) SLOPE: "+str(round(b_inf,5)))

    a_L2, b_L2 = best_fit(h, L2_norm)
    yfit = [a_L2 + b_L2 * xi for xi in h]
    plt.plot(h, yfit,color="k",label="(L2)  SLOPE: "+str(round(b_L2,5)))
    plt.legend()
```

```
    plt.savefig("Figures/Error/norm_err.png")
    plt.close()

if __name__ == "__main__":
    grid_spacing = [1/(10*x) for x in range(1,10)]
    l2 = []
    inf = []
    for h in grid_spacing:
        mu = 1.0/h
        L,I = Peaceman_Rachford(h,mu)
        l2.append(L[-1])
        inf.append(I[-1])
        #makeGif("Peaceman_Rachford_h_"+str(h)+"_Mu_"+str(mu),10)
        #ExactGIF(h,mu)
    plot_norm(-np.log10(grid_spacing),-np.log10(inf),-np.log10(l2))

"""
Report:
    In the error figure we capture second order acuracy in the INF norm.
    We see that the slope ~ 2.0 which resembles second order accuracy. Compared
    to other methods we have seen, the PR alogrithm and other ADI methods,
    do not seem to be as accurate as they should be. In other words, ADI methods
    could be order of mangitdues of accuracy away from spectral or other FEM.
"""
```