Problem 3

The attached code solves the problem

$$a(x,y)u(x,y) - \mathrm{div}(d(x,y)\mathrm{grad}((u(x,y)))) = f(x,y)$$

for $0 \le x \le L_x$, $0 \le y \le L_y$ where $u = g(x,y)$ on the boundary.

I chose a fairly difficult problem to stress-test the code, choosing

$$\begin{aligned}
a(x,y) &= \cos(\pi x)\cos(\pi y) \\
d(x,y) &= (x+1)^2(y+1)^2 \\
f(x,y) &= \cos(\pi x)\cos(\pi y)\sin(2\pi x^3)\sin(3\pi y^2) \\
&\quad - [2(x+1)(y+1)^2(6\pi x^2\cos(2\pi x^3)\sin(3\pi y^2)) \\
&\quad + 2(x+1)^2(y+1)(6\pi y\sin(2\pi x^3)\cos(3\pi y^2)) \\
&\quad + (x+1)^2(y+1)^2\{-36\pi^2 x^4\sin(2*\pi x^3)\sin(3\pi y^2) \\
&\quad + 12\pi x\cos(2\pi x^3)\sin(3\pi y^2) \\
&\quad - 36\pi^2 y^2\sin(2\pi x^3)\sin(3\pi y^2) \\
&\quad + 6\pi\sin(2\pi x^3)\cos(3\pi y^2)\}]
\end{aligned}$$

with $g(x,y) = 0$, which has exact solution

$$u(x,y) = \sin(2\pi x^3)\sin(3\pi y^2)\,.$$

This problem was chosen because it has several Fourier modes in addition to polynomial growth. If the finite difference solver converges for this problem, we will know that it was not a coincidence, and that all aspects of the code are working correctly together.

In Figure 1 we plot the finite difference solution and in Figure 2 we plot the exact solution. Finally, in Figure 3 we plot the convergence for this problem. We do, in fact, get the second order convergence that we were expecting. Thus it appears that everything is implemented correctly.

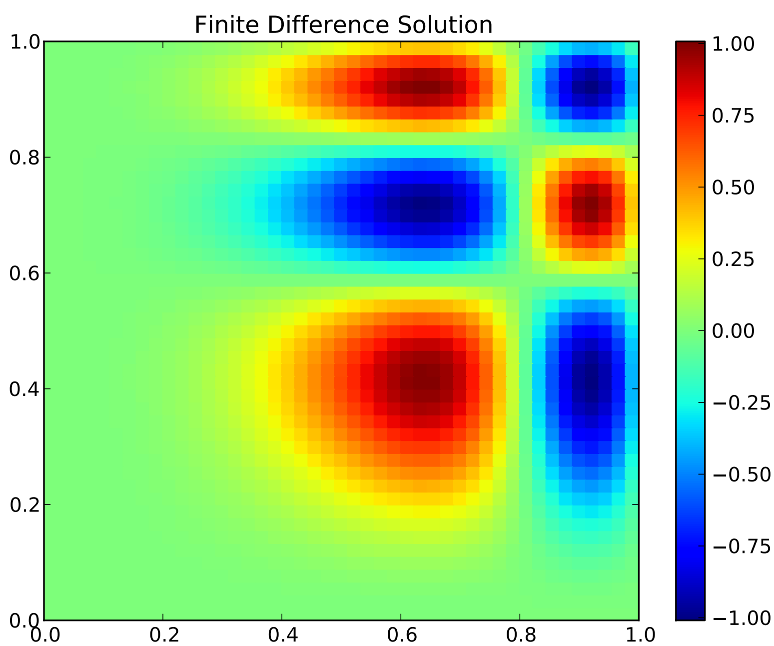

Figure 1: Finite difference solution

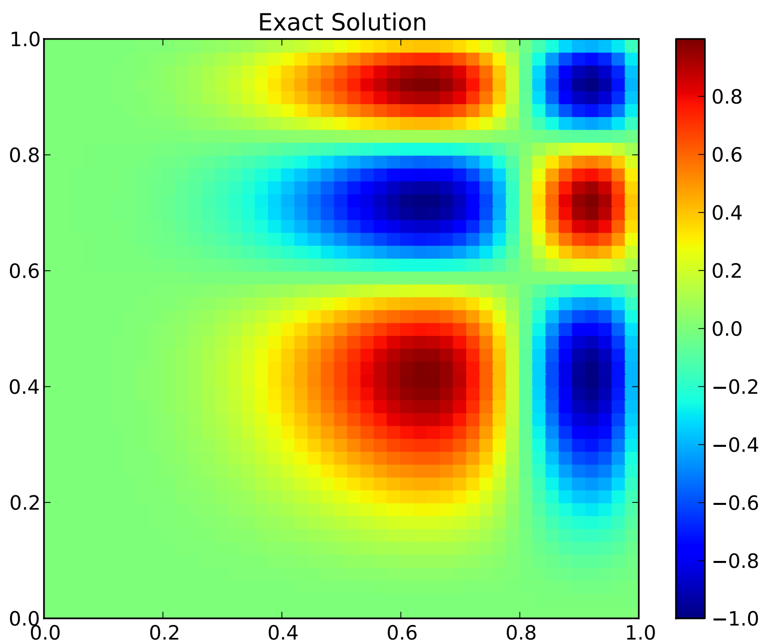

Figure 2: Exact solution

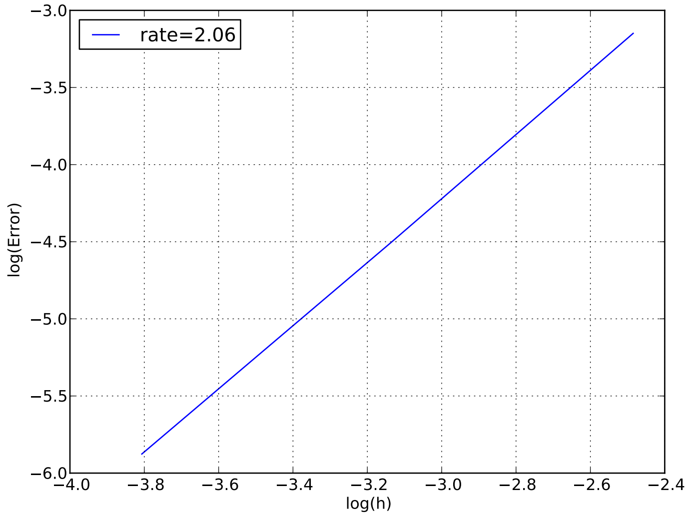

Figure 3: Second order finite difference convergence

FDSolver.py

```
# Second Order Finite Difference Solver
# Solves a(x,y)u(x,y) - div(d(x,y) grad(u(x,y))) = f(x,y)
# Written by: Truman Ellis
# Numerical Treatment of Differential Equations
# Spring 2011

from pylab import *
import code

close('all')

# Define Initial Refinement
nx = 11
ny = 11

# Define problem domain
xmin = 0.
xmax = 1.
ymin = 0.
ymax = 1.

# Define boundary conditions
def g(x,y):
    return 0

# Define forcing function to allow for convergence tests
def f(x,y):
    return a(x,y)*u_ex(x,y) - (dx(x,y)*ux(x,y)
        + dy(x,y)*uy(x,y) + d(x,y)*(uxx(x,y)+uyy(x,y)))

def a(x,y):
    return cos(pi*x)*cos(pi*y)

def d(x,y):
    return (x+1)**2*(y+1)**2

# Set functions for exact solution
def dx(x,y):
    return 2*(x+1)*(y+1)**2

def dy(x,y):
    return 2*(x+1)**2*(y+1)

def u_ex(x,y):
    return sin(2*pi*x**3)*sin(3*pi*y**2)

def ux(x,y):
    return 2*pi*cos(2*pi*x**3)*sin(3*pi*y**2)*3*x**2
```

```
def uy(x,y):
    return 3*pi*sin(2*pi*x**3)*cos(3*pi*y**2)*2*y

def uxx(x,y):
    return -4*pi**2*sin(2*pi*x**3)*sin(3*pi*y**2)*9*x**4 \
        + 2*pi*cos(2*pi*x**3)*sin(3*pi*y**2)*6*x

def uyy(x,y):
    return -9*pi**2*sin(2*pi*x**3)*sin(3*pi*y**2)*4*y**2 \
        + 3*pi*sin(2*pi*x**3)*cos(3*pi*y**2)*2

# Define element mapping
def IndexMap(i,j, ny):
    return j + ny*i

def ReverseMap(k, ny):
    return (k/ny, k%ny)

# Set number of refinement steps for convergence test
nref = 3
error = zeros(nref)
ndofs = zeros(nref)
hs = zeros(nref)
for r in range(0,nref):
    if r > 0:
        nx *= 2
        ny *= 2
    ndofs[r] = nx*ny
    hx = (xmax - xmin)/(nx + 1)
    hy = (ymax - ymin)/(ny + 1)
    hs[r] = hx

    X = linspace(xmin, xmax, nx+2)
    xh = 0.5*(X[0:-1]+X[1:])
    Y = linspace(ymin, ymax, ny+2)
    yh = 0.5*(Y[0:-1]+Y[1:])
    x = X[1:-1]
    y = Y[1:-1]
    A = zeros((nx*ny,nx*ny))
    u = zeros(nx*ny)
    exact = zeros(nx*ny)
    b = zeros(nx*ny)


    for i in range(0,nx):
        for j in range(0,ny):
            A[IndexMap(i,j,ny),IndexMap(i,j,ny)] = a(x[i],y[j]) \
                + (d(xh[i+1], y[j]) \
                + d(xh[i], y[j]))/(hx**2) \
                + (d(x[i], yh[j+1]) \
                + d(x[i], yh[j]))/(hy**2)
            b[IndexMap(i,j,ny)] += f(x[i],y[j])
            if (i < nx-1):
                A[IndexMap(i,j,ny),IndexMap(i+1,j,ny)] = -d(xh[i+1], y[j])/hx**2
            else:
                b[IndexMap(i,j,ny)] += g(xmax,y[j])/hx**2
            if (i > 0):
                A[IndexMap(i,j,ny),IndexMap(i-1,j,ny)] = -d(xh[i], y[j])/hx**2
            else:
                b[IndexMap(i,j,ny)] += g(xmin,y[j])/hx**2
            if (j < ny -1):
                A[IndexMap(i,j,ny),IndexMap(i,j+1,ny)] = -d(x[i], yh[j+1])/hy**2
            else:
                b[IndexMap(i,j,ny)] += g(x[i],ymax)/hy**2
            if (j > 0):
                A[IndexMap(i,j,ny),IndexMap(i,j-1,ny)] = -d(x[i], yh[j])/hy**2
            else:
                b[IndexMap(i,j,ny)] += g(x[i],ymin)/hy**2

            # Calculate exact solution
            exact[IndexMap(i,j,ny)] = u_ex(x[i],y[j])

    # Solve for finite difference solution
    u = linalg.solve(A,b)
    error[r] = sqrt(((u-exact)**2).sum()/(nx*ny))

    # Plotting
    if r == 2:
        U = zeros((nx+2, ny+2))
        E = zeros((nx+2, ny+2))
        for i in range(0, nx):
            for j in range(0, ny):
                U[i+1,j+1] = u[IndexMap(i,j,ny)]
                E[i+1,j+1] = exact[IndexMap(i,j,ny)]
```

```
        for i in range(0,nx+2):
            U[i,0] = g(X[i],Y[0])
            U[i,-1] = g(X[i],Y[-1])
            E[i,0] = g(X[i],Y[0])
            E[i,-1] = g(X[i],Y[-1])
        for j in range(0, ny+2):
            U[0,j] = g(X[0],Y[j])
            U[-1,j] = g(X[-1],Y[j])
            E[0,j] = g(X[0],Y[j])
            E[-1,j] = g(X[-1],Y[j])

        figure(1)
        pcolor(X, Y, U.T)
        colorbar()
        title('Finite Difference Solution')
        figure(2)
        pcolor(X, Y, E.T)
        colorbar()
        title('Exact Solution')
        show()

figure(3)
plot(log(hs),log(error))
(m,b) = polyfit(log(hs),log(error),1)
xlabel('log(h)')
ylabel('log(Error)')
grid()
legend(('rate=%.2f' % (m,),), loc='best')
```