Numerical Treatment of Differential Equations: Homework 6

Truman Ellis

March 11, 2011

Our finite difference code solves Laplace's equation

$$-\Delta u = f$$

by approximating the Laplacian with matrix operator $\mathbf{A}$. $\mathbf{A}$ has eigenvalues $\lambda^{a,b} = \frac{4}{h^2}\left\{\sin^2\left(\frac{a\pi h}{2}\right) + \sin^2\left(\frac{b\pi h}{2}\right)\right\}$ and eigenvectors $v_{ij}^{a,b} = \sin(ai\pi h)\sin(bj\pi h)$ for $1 \leq a\,, b \leq m$.

If we set $f = \lambda^{a,b} v^{a,b}$ we will be solving the eigenvalue problem

$$\mathbf{A}u = \lambda^{a,b} v^{a,b}$$

and we should exactly recover the solution $u = v^{a,b}$. Indeed, for various values of a and b we recover the exact eigenvectors to machine precision as shown in Figures (1) - (6).

If we instead replace our forcing function with the eigenvalues and eigenvectors of the continuous operator, $\lambda_{a,b} = (a^2 + b^2)\pi^2$ and $v_{a,b} = \sin(a\pi x)\sin(b\pi y)$, respectively, we can observe the convergence of our discrete operator to the continuous. The final set of figures demonstrate this convergence for various values of a and b.

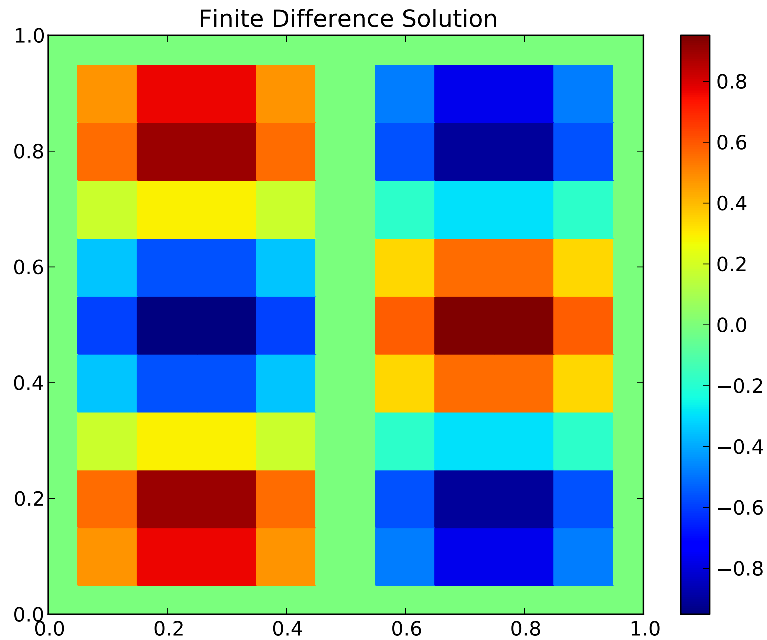

Figure 1: $a = 2$, $b = 3$, error $= 2.24662632e - 16$

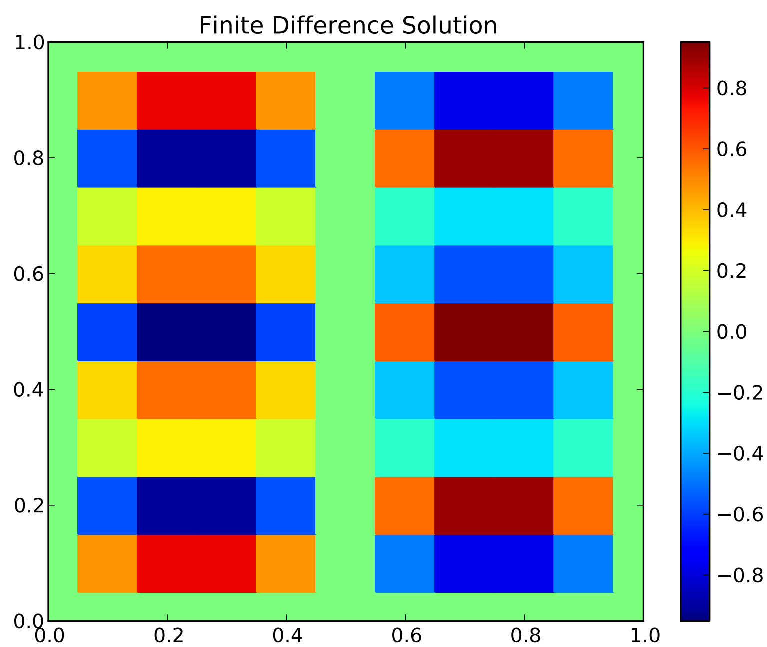

Figure 2: $a = 2$, $b = 7$, error $= 6.56173233e - 16$

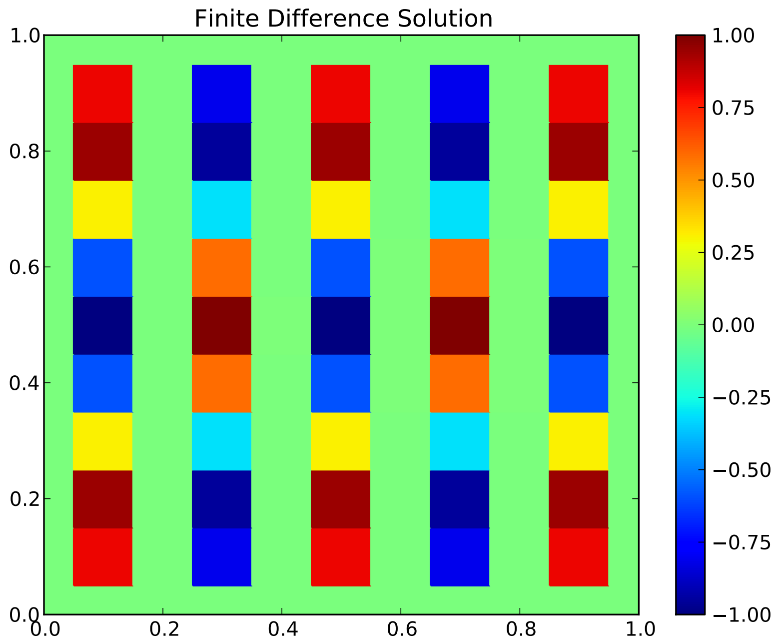

Figure 3: $a = 5$, $b = 3$, error $= 5.12525209e - 16$

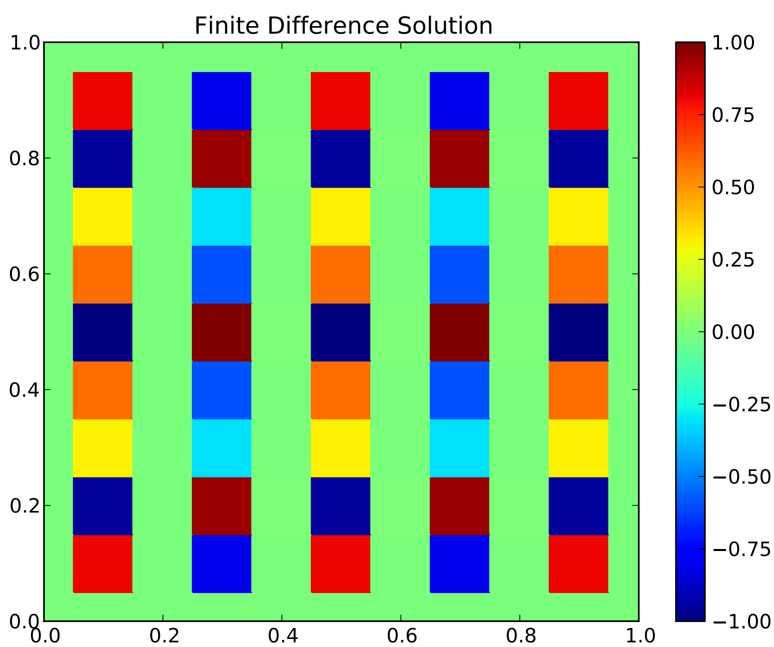

Figure 4: $a = 5$, $b = 7$, error $= 4.37301726e - 16$

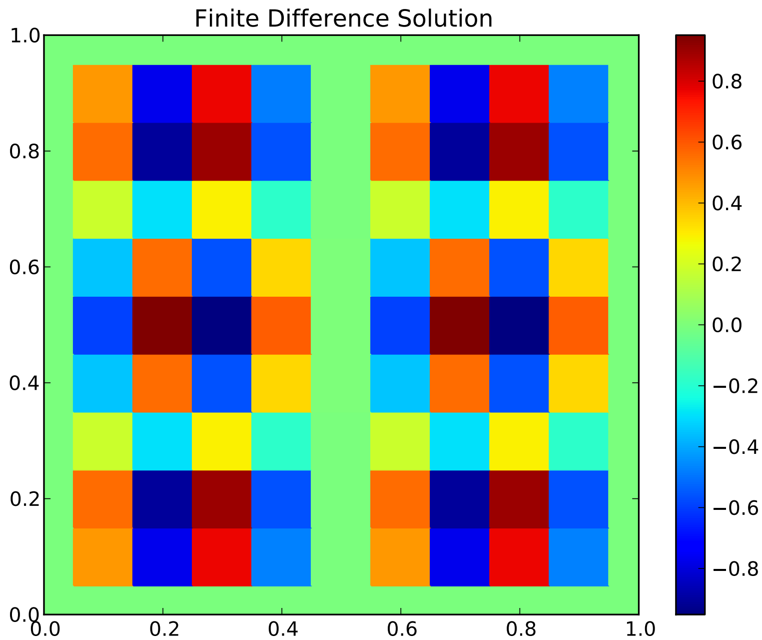

Figure 5: $a = 8$, $b = 3$, error $= 6.98042995e - 16$

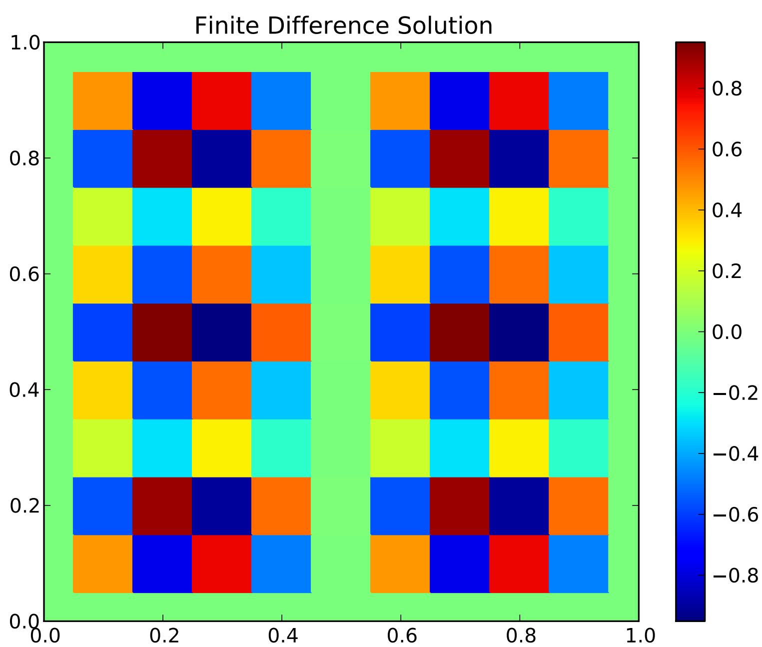

Figure 6: $a = 8$, $b = 7$, error $= 3.56630134e - 16$

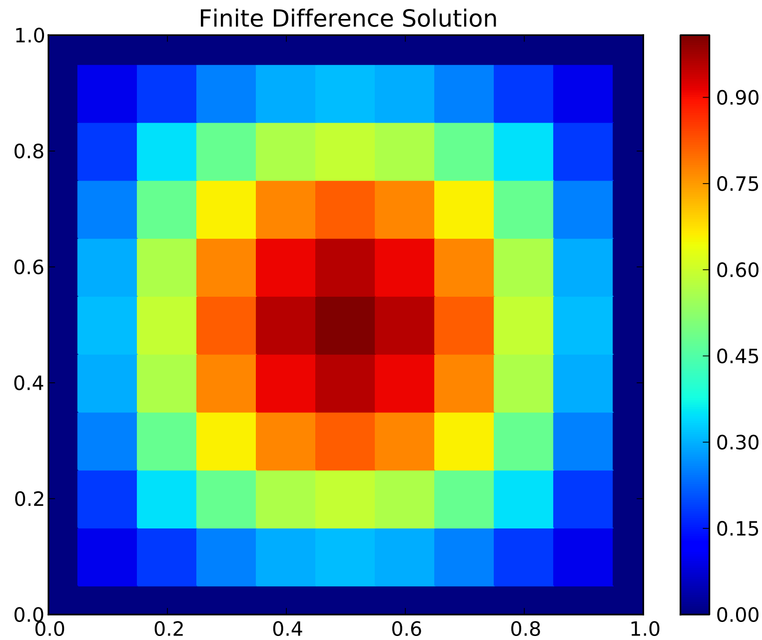

Figure 7: $a = 1$, $b = 1$: Eigenvector

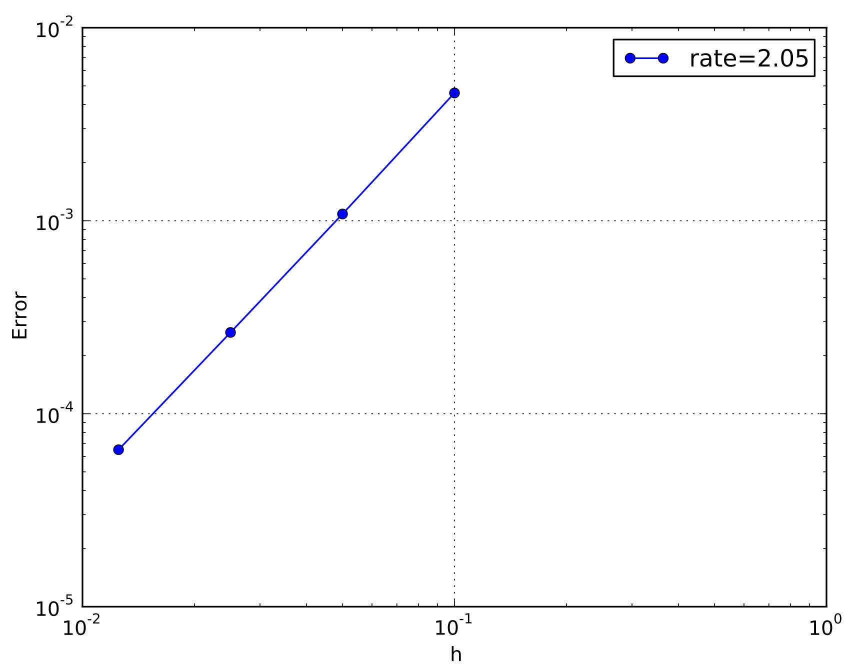

Figure 8: $a = 1$, $b = 1$: Convergence

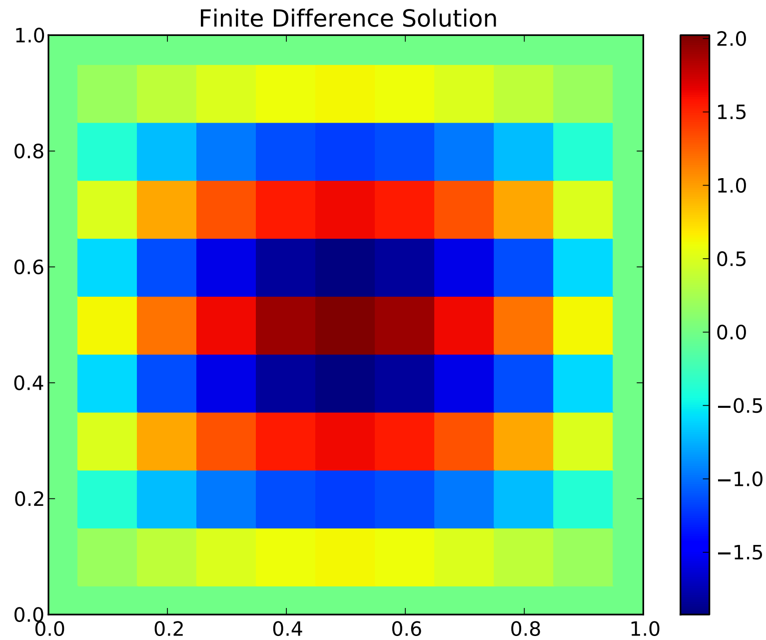

Figure 9: $a = 1$, $b = 9$: Eigenvector

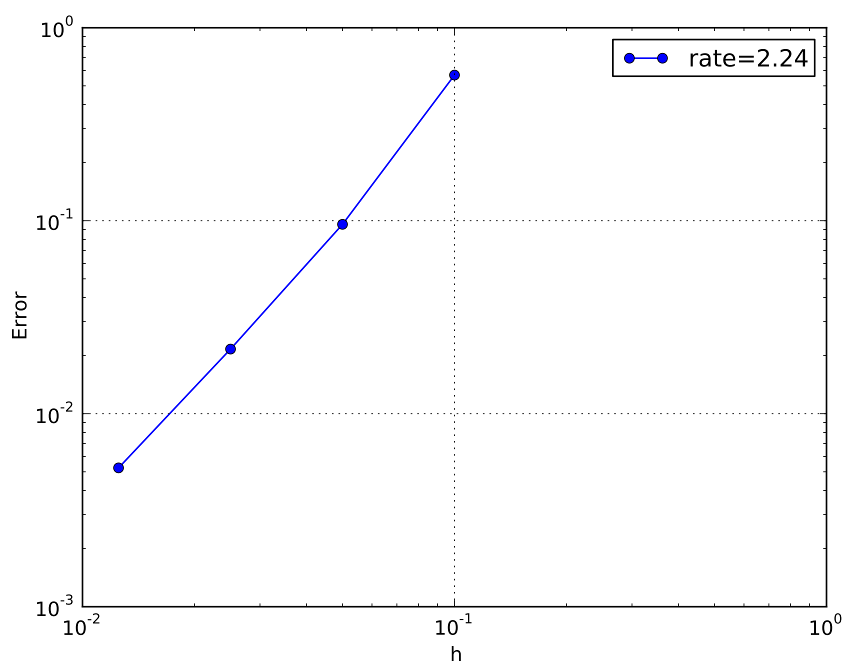

Figure 10: $a = 1$, $b = 9$: Convergence

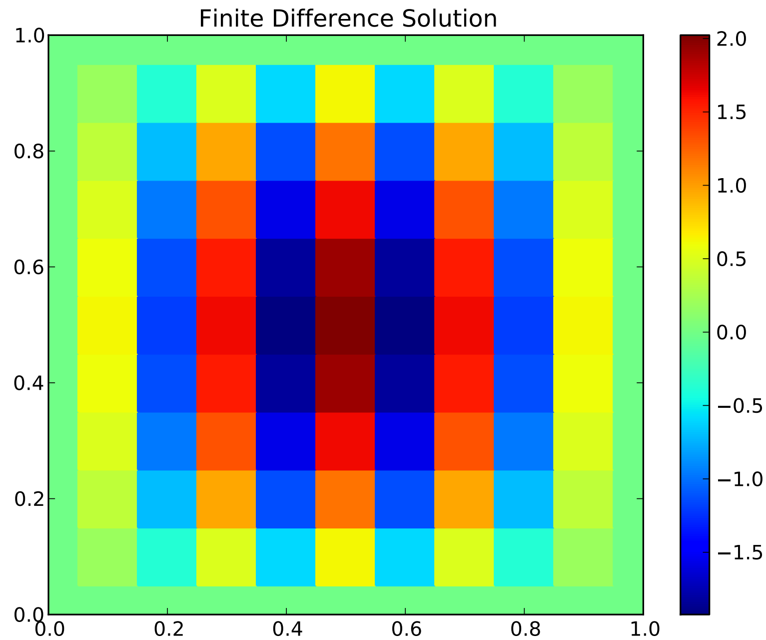

Figure 11: $a = 9$, $b = 1$: Eigenvector

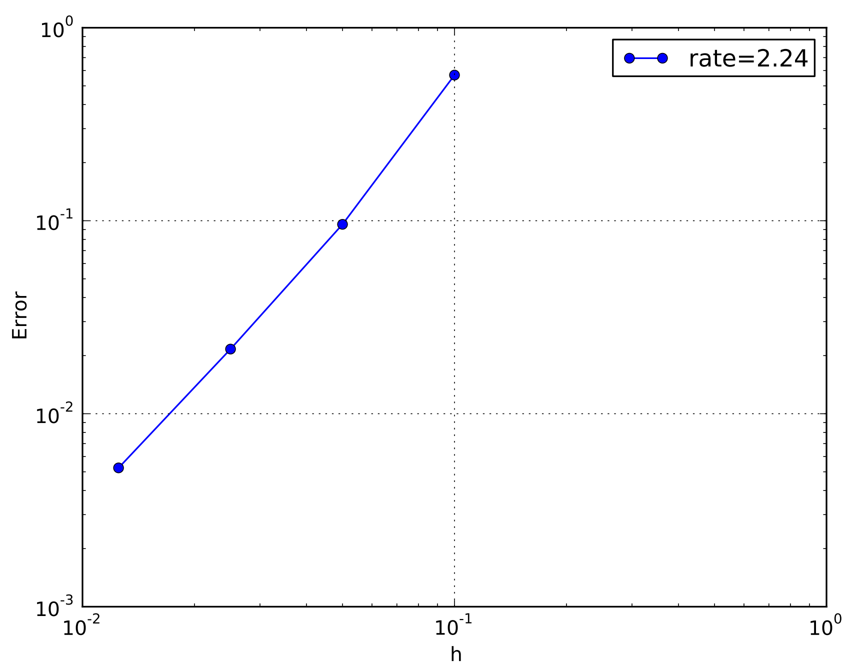

Figure 12: $a = 9$, $b = 1$: Convergence

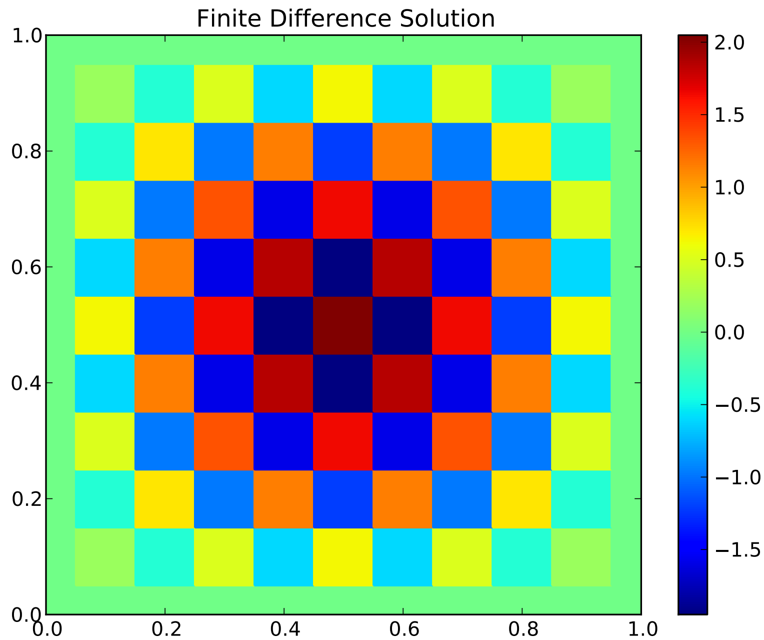

Figure 13: $a = 9$, $b = 9$: Eigenvector

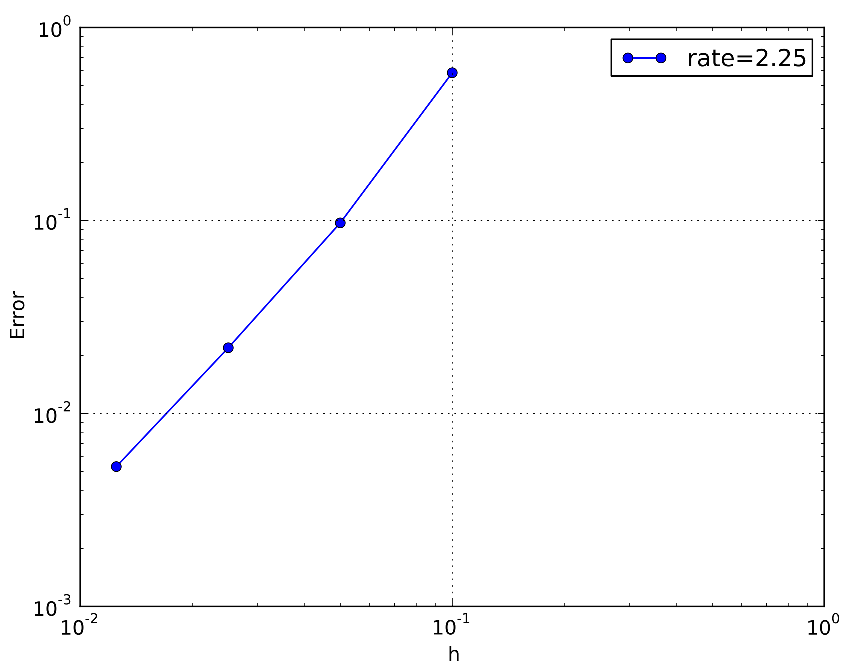

Figure 14: $a = 9$, $b = 9$: Convergence

```
# 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')

# Set eigenstate variables
aa = 9.
bb = 9.

# Define Initial Refinement
nx = 9
ny = 9

# 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):
    # Lambda for discrete operator
    #L = 4./hx**2*(sin(aa*pi*hx/2)**2 + sin(bb*pi*hx/2)**2)
    # Lambda for continuous operator
    L = (aa**2 + bb**2)*pi**2
    return L*sin(aa*pi*x)*sin(bb*pi*y)

def a(x,y):
    return 0

def d(x,y):
    return 1

# Set functions for exact solution
def u_ex(x,y):
    return sin(aa*pi*x)*sin(bb*pi*y)

# 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 = 4
error = zeros(nref)
ndofs = zeros(nref)
hs = zeros(nref)
for r in range(0,nref):
    if r > 0:
        nx = 2*(nx+1)-1
        ny = 2*(ny+1)-1
    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 == 0:
        X[1:] -= hx/2
        Y[1:] -= hy/2
        X = append(X,xmax)
        Y = append(Y,ymax)
        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)
loglog(hs,error,'-o')
(m,b) = polyfit(log(hs),log(error),1)
xlabel('h')
ylabel('Error')
grid()
legend(('rate=%.2f' % (m,),), loc='best')
```