1. $proj_{\vec{v}_1}\vec{y} = \frac{\vec{v}_1 \cdot \vec{y}}{|\vec{v}_1|^2}\vec{v}_1$ $\quad proj_{\vec{v}_2}\vec{y} = \frac{\vec{v}_2 \cdot \vec{y}}{|\vec{v}_2|^2}\vec{v}_2$

a)

b) $\cos^{-1}\frac{\vec{y}\cdot\vec{y}_i}{|\vec{y}|\,|\vec{y}_i|} = \cos^{-1}\frac{\vec{y}\cdot\frac{\vec{v}_i\cdot\vec{y}}{|\vec{v}_i|^2}\vec{v}_i}{|\vec{y}|\,\left|\frac{\vec{v}_i\cdot\vec{y}}{|\vec{v}_i|^2}\vec{v}_i\right|}$

c) $(\vec{y}-\vec{y}_i)\cdot(\vec{y}-\vec{y}_i) = \left(\vec{y}-\frac{\vec{v}_i\cdot\vec{y}}{|\vec{v}_1|^2}\vec{v}_1\right)^2$

d) $\left(\vec{y}-\frac{\vec{v}_1\cdot\vec{y}}{|\vec{v}_1|^2}\vec{v}_1\right)^2 < \left(\vec{y}-\frac{\vec{v}_2\cdot\vec{y}}{|\vec{v}_2|^2}\vec{v}_2\right)^2$

2. a) $\text{Span}\left\{\begin{bmatrix}1\\3\\1\end{bmatrix},\begin{bmatrix}-1\\-3\\-1\end{bmatrix},\begin{bmatrix}-3\\-5\\-1\end{bmatrix},\begin{bmatrix}4\\8\\2\end{bmatrix}\right\}$

dimension = 4

c) $\text{span}\{(1\ -1\ -3\ 4), (3\ -3\ -5\ 8), (1\ -1\ -1\ 2)\}$

dimension = 3

b) $\left[\begin{array}{cccc|c}1 & -1 & -3 & 4 & 0\\ 3 & -3 & -5 & 8 & 0\\ 1 & -1 & -1 & 2 & 0\end{array}\right] \sim \Big[\quad\Big]$

2. $\begin{bmatrix} 1 & -1 & -3 & 4 \\ 3 & -3 & -5 & 8 \\ 1 & -1 & -1 & 2 \end{bmatrix} \sim \begin{bmatrix} 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$

a) $\text{span}\left\{ \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}, \begin{bmatrix} -3 \\ -5 \\ -1 \end{bmatrix} \right\}$ $\dim = 2$

c) $\text{span}\{ (1\ -1\ -3\ 4), (3\ -3\ -5\ 8) \}$ $\dim = 2$

b) $x_1 = x_2 - x_4$

$x_3 = x_4$

$$\begin{bmatrix} x_2 - x_4 \\ x_2 \\ x_4 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} x_2 + \begin{bmatrix} -1 \\ 0 \\ 1 \\ 1 \end{bmatrix} x_4$$

$\text{span}\left\{ \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \right\}$ $\dim = 2$

d) $A^T = \begin{bmatrix} 1 & 3 & 1 \\ -1 & -3 & -1 \\ -3 & -5 & -1 \\ 4 & 8 & 2 \end{bmatrix} \sim \begin{bmatrix} 1 & 3 & 1 \\ 0 & 0 & 0 \\ 0 & 4 & 2 \\ 0 & -4 & -2 \end{bmatrix} \sim \begin{bmatrix} 1 & 3 & 1 \\ 0 & 2 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

$\sim \begin{bmatrix} 1 & 1 & 0 \\ 0 & 2 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ $x_1 = -x_2$

$x_3 = -2x_2$ $\begin{bmatrix} -1 \\ 1 \\ -2 \end{bmatrix}$

$\text{span}\left\{ \begin{bmatrix} -1 \\ 1 \\ -2 \end{bmatrix} \right\}$ $\dim = 1$

3. a) $\begin{bmatrix} \vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_i \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & & \\ \vdots & & \ddots & \\ 0 & & & \lambda_i \end{bmatrix} \begin{bmatrix} \vec{v}_1, \vec{v}_2 \dots \vec{v}_i \end{bmatrix}^T$

b) 400 eigen vectors are required

c) ipython

d) 15

4. a) $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

b) 1 one-hop
0 two-hops
1 three-hop

c) ~~[illegible]~~ both are $\frac{1}{2}$

d) $\begin{bmatrix} 0 & 1 & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{3} \\ 1 & 0 & \frac{1}{3} & 0 \end{bmatrix}$

e) 1 two-hop path
1 three-hop path

f) 0.33, 0.17, 0.12, 0.38

g) $\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 & 0 \end{bmatrix}$

h) 0 paths

i) 0.2, 0.2, 0.4, 0.1, 0.1

In graph c some nodes don't relate to each other at all

5. Gary Li
3031920100

EE16A Homework 4

Image Compression

In [1]:
```
%pylab inline
```

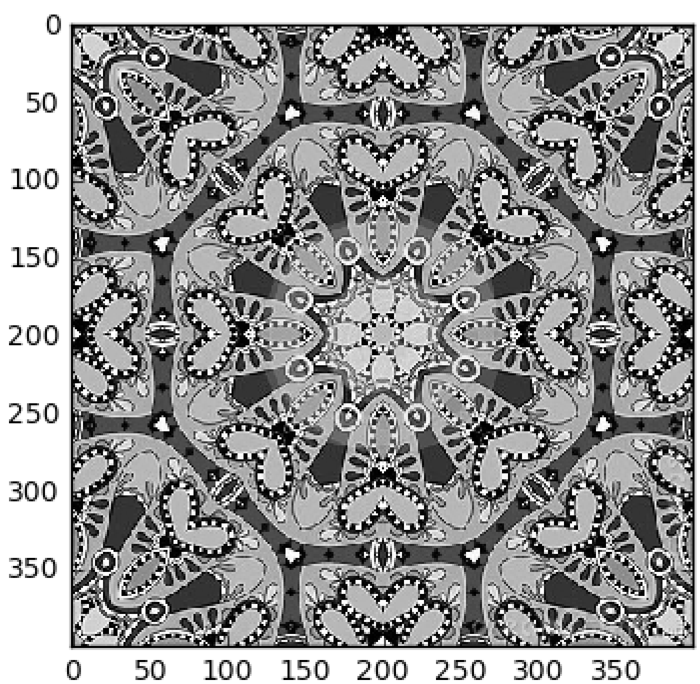
```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:
```
import numpy as np
from scipy import ndimage as nd
from scipy import misc
from scipy import io
```

Part b

In [23]:
```
#Load Pattern Image
pattern = np.load('pattern.npy')
plt.imshow(pattern, cmap='gray', interpolation='nearest')
```

Out[23]:
```
<matplotlib.image.AxesImage at 0x110b27eb8>
```



Use the command shape (http://docs.scipy.org/doc/numpy/reference/generated/numpy.ndarray.shape.html) to find the dimensions of the image. How many eigenvalues do you expect?

Run the code below to find the eigenvector and eigenvalues of `pattern` and sort them in descending order (first eigenvalue/vector corresponds to the largest eigenvalue)

In [28]:
```
eig_vals, eig_vectors = np.linalg.eig(pattern)
idx = (abs(eig_vals).argsort())
idx = idx[::-1]
eig_vals = eig_vals[idx]
eig_vectors = eig_vectors[:,idx]
print(np.shape(pattern))
```

```
(400, 400)
```

Part c

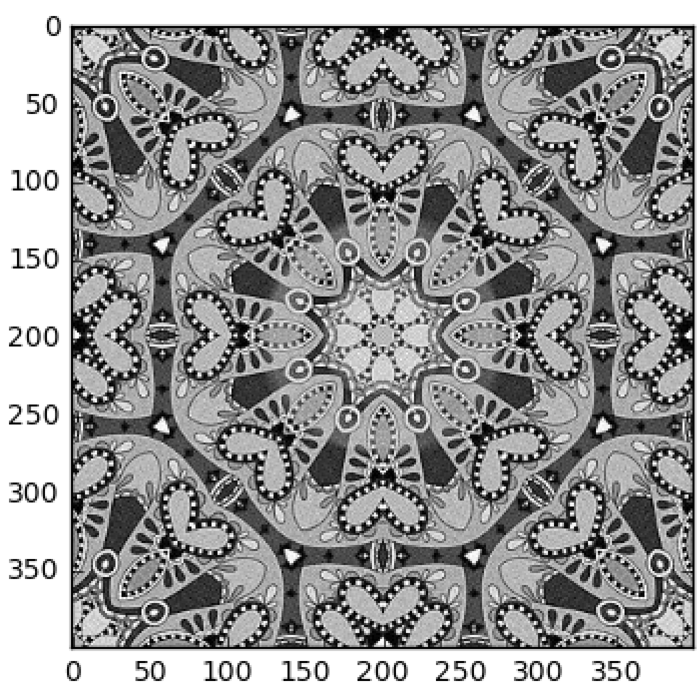
Find the pattern approximation using 100 largest eigenvalues/eigenvectors.

- Index into above variables to choose the first 100 eigenvalues and eigenvectors.
- You can use the command np.outer (http://docs.scipy.org/doc/numpy/reference/generated/numpy.outer.html) to find the outer product of two vectors

In [29]:
```
rank = 100
S = np.zeros(pattern.shape)
for i in range(rank):
    vec_i = eig_vectors[:,i]  # i-th largest eigenvector
    val_i = eig_vals[i]       # i-th largest eigenvalue
    S += np.outer(np.outer(vec_i, val_i), vec_i.T)

plt.imshow(S, cmap='gray', vmin=0, vmax=255)
```

Out[29]: <matplotlib.image.AxesImage at 0x111c552e8>



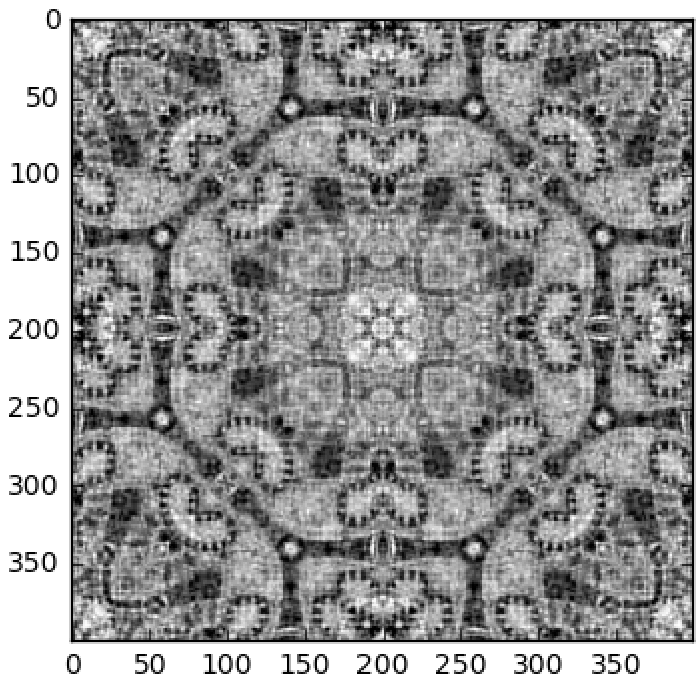
Part d

Find the pattern approximation using 50 largest eigenvalues/eigenvectors

In [36]:
```
rank = 15
S = np.zeros(pattern.shape)
for i in range(rank):
    vec_i = eig_vectors[:,i]  # i-th largest eigenvector
    val_i = eig_vals[i]       # i-th largest eigenvalue
    S += np.outer(np.outer(vec_i, val_i), vec_i.T)

plt.imshow(S, cmap='gray', vmin=0, vmax=255)
```

Out[36]: <matplotlib.image.AxesImage at 0x1171ddc50>



Paths of a Surfer

In [22]:
```
# There is no required ipython component, but you may wish to use iPython fo
B = np.array([[0, 1, 1/3, 1/3],
               [0, 0, 1/3, 1/3],
               [0, 0, 0, 1/3],
               [1, 0, 1/3, 0]])
vbi = np.array([1/4, 1/4, 1/4, 1/4])
print(np.dot(B, np.dot(B, np.dot(B, np.dot(B, np.dot(B, np.dot(B, np.dot(B,

C = np.array([[0, 1, 0, 0, 0],
              [1, 0, 0, 0, 0],
              [0, 0, 0, 1, 1],
              [0, 0, 1/2, 0, 0],
              [0, 0, 1/2, 0, 0]])
vci = np.array([1/5, 1/5, 1/5, 1/5, 1/5])
final = np.dot(C, vci)
for _ in range(100):
    final = np.dot(C, final)
print(final)
```

```
[ 0.33373554  0.1663576   0.12448983  0.37541703]
[ 0.2  0.2  0.4  0.1  0.1]
```

In [ ]: