Task1_PEC-PMC_Eigenmodes

May 18, 2016

```
In [1]: import numpy as np
        import scipy as sp
        import scipy.sparse.linalg as linalg
        import matplotlib.pyplot as plt
        %matplotlib inline
```

1 Task 1: 1D PEC-PMC Cavity, Eigenmode formulation

1.1 Maxwell Equations (time domain)

$$\nabla \times E = -\mu \frac{\partial H}{\partial t} \tag{1}$$

$$\nabla \times H = \epsilon \frac{\partial E}{\partial t} \tag{2}$$

1.2 Time Domain $\rightarrow$ Frequency Domain

Using the Fourier transform we get:

$$E(r,t) \rightarrow E(r,\omega) \tag{3}$$

$$\frac{\partial E}{\partial t} \rightarrow i\omega E \tag{4}$$

Thus our wave equation transforms into:

$$\nabla E = -i\omega\mu H \tag{5}$$

$$\nabla E = -i\omega\epsilon E \tag{6}$$

1.3 1D Coordinates

$$H_{y_i}, E_{z_{i+\frac{1}{2}}}, i \in \mathcal{N} \tag{7}$$

$$\frac{\partial E_z}{\partial x} = -i\omega\mu H_y \tag{8}$$

$$\frac{\partial H_y}{\partial x} = i\omega\epsilon E_z \tag{9}$$

1.4 Discretization

$$\frac{\partial E_{zi}}{\partial x} \approx \frac{E_{zi+1} - E_{zi-1}}{\Delta x} \tag{10}$$

Forward difference

$$-i\omega\mu E_{z_{i+\frac{1}{2}}} \approx \frac{H_{y_{i+1}} - H_{y_i}}{\Delta x} \tag{11}$$

Backward difference

$$-i\omega\epsilon H_{y_i} \approx \frac{E_{z_{i+\frac{1}{2}}} - E_{z_{i-\frac{1}{2}}}}{\Delta x} \tag{12}$$

1.5 Wave equation

$$H_y = \frac{i}{\omega\mu}\frac{\partial E_z}{\partial x} \tag{13}$$

$$E_z = \frac{-i}{\omega\epsilon}\frac{\partial H_y}{\partial x} \tag{14}$$

$$\frac{\partial}{\partial x}\frac{i}{\omega\mu}\frac{\partial}{\partial x}E_z = i\omega\epsilon E_z \tag{15}$$

$$\frac{1}{\epsilon}\frac{\partial}{\partial x}\frac{1}{\mu}\frac{\partial}{\partial x}E_z = \omega^2 E_z \tag{16}$$

or

$$\frac{1}{\mu}\frac{\partial}{\partial x}\frac{1}{\epsilon}\frac{\partial}{\partial x}H_y = \omega^2 H_y \tag{17}$$

1.6 Matrix form

$$H_y = \frac{i}{\omega\mu}\frac{\partial E_z}{\partial x} \approx \frac{i}{\omega\mu}\frac{E_{z_{i+\frac{1}{2}}} - E_{z_{i-\frac{1}{2}}}}{\Delta x} \tag{18}$$

$$\begin{bmatrix} H_{y_0} \\ H_{y_1} \\ \vdots \\ H_{y_{n-1}} \\ H_{y_n} \end{bmatrix} = \frac{i}{\omega\mu\Delta x}\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix}\begin{bmatrix} E_{z_{0+\frac{1}{2}}} \\ E_{z_{1+\frac{1}{2}}} \\ \vdots \\ E_{z_{n-\frac{1}{2}}} \\ E_{z_{n+\frac{1}{2}}} \end{bmatrix} \tag{19}$$

$$E_z = \frac{-i}{\omega\epsilon}\frac{\partial H_y}{\partial x} \approx \frac{-i}{\omega\epsilon}\frac{H_{y_{i+1}} - H_{y_i}}{\Delta x} \tag{20}$$

$$\begin{bmatrix} E_{z_{0+\frac{1}{2}}} \\ E_{z_{1+\frac{1}{2}}} \\ \vdots \\ E_{z_{n-\frac{1}{2}}} \\ E_{z_{n+\frac{1}{2}}} \end{bmatrix} = \frac{i}{\omega\mu\Delta x}\begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ 0 & 0 & -1 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & \dots & 0 & -1 \end{bmatrix}\begin{bmatrix} H_{y_0} \\ H_{y_1} \\ \vdots \\ H_{y_{n-1}} \\ H_{y_n} \end{bmatrix} \tag{21}$$

$$\omega^2 E_z = \frac{1}{\epsilon}\frac{\partial}{\partial x}\frac{1}{\mu}\frac{\partial}{\partial x}E_z \tag{22}$$

$$\omega^2 \begin{bmatrix} E_{z0+\frac{1}{2}} \\ E_{z1+\frac{1}{2}} \\ \vdots \\ E_{zn-\frac{1}{2}} \\ E_{zn+\frac{1}{2}} \end{bmatrix} = \frac{i}{\mu\Delta x} \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ 0 & 0 & -1 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & \dots & 0 & -1 \end{bmatrix} \frac{i}{\mu\Delta x} \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix} \begin{bmatrix} E_{z0+\frac{1}{2}} \\ E_{z1+\frac{1}{2}} \\ \vdots \\ E_{zn-\frac{1}{2}} \\ E_{zn+\frac{1}{2}} \end{bmatrix} \tag{23}$$

or, simplifing, and replacing $i+\frac{1}{2}$ with i:

$$\omega^2 \begin{bmatrix} E_{z0} \\ E_{z1} \\ \vdots \\ E_{zn-1} \\ E_{zn} \end{bmatrix} = -\frac{1}{\mu\epsilon\Delta x^2} \begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ 0 & 0 & -1 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 1 \\ 0 & 0 & \dots & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \dots & -1 & 1 \end{bmatrix} \begin{bmatrix} E_{z0} \\ E_{z1} \\ \vdots \\ E_{zn-1} \\ E_{zn} \end{bmatrix} \tag{24}$$

1.7 Boundary Conditions

1.7.1 PEC

$$E_{zi} = 0 \tag{25}$$

1.7.2 PMC

$$H_{y_i} = 0 \tag{26}$$

In our equations, they are automatically satisfied at the boundaries.

2 Implementation

2.1 Setup grid

```
In [2]: n   = 100       # Num grid nodes
        dx  = 1. / n    # Step size for overall size of 1

        eps = 1.        #  Vacuum
        mu  = 1.        #  Vacuum

        A   = - 1. / (dx**2 * mu * eps)
```

2.2 Build matrix

In [3]:
```
# Forward

diag = np.ones(n) * -A
up_diag = np.ones(n) * A

M_1 = sp.sparse.dia_matrix(([up_diag, diag], [1, 0]), [n,n])

# Backward

diag = np.ones(n) * 1
up_diag = np.ones(n) * -1

M_2 = sp.sparse.dia_matrix(([up_diag, diag], [-1, 0]), [n,n])

M = M_1.dot(M_2)
```

2.3 Solve for eigenmodes

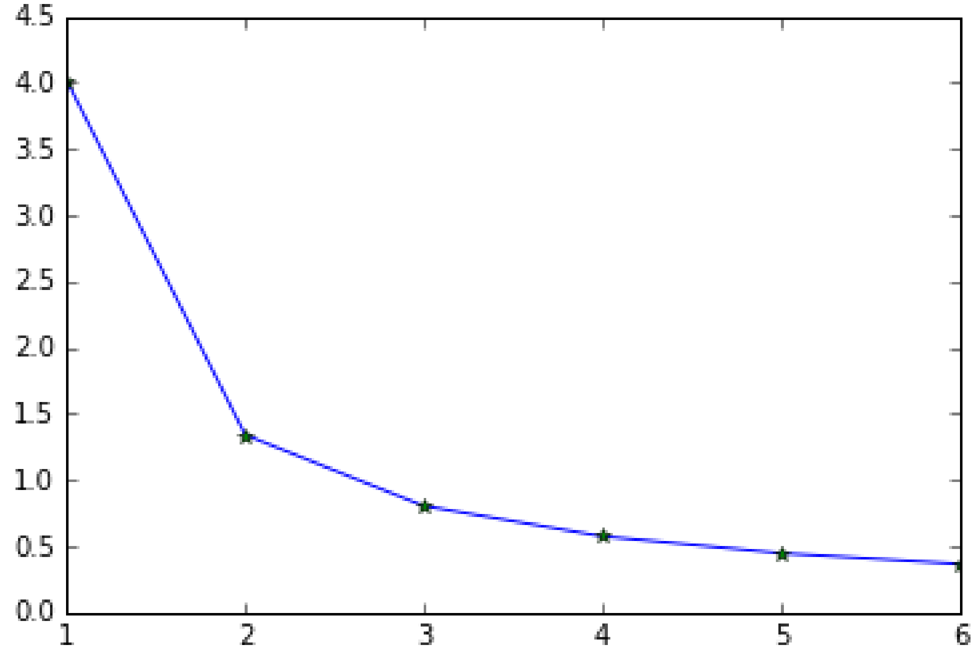
In [4]:
```
kt = 2*sp.pi*1                                          # wave vector target to f
num_eigs = 6
k2, V = linalg.eigs(M, k=num_eigs, M=None, sigma=kt**2)  # solve for eigenv

V_indicies = np.linspace(0,num_eigs-1, num_eigs, dtype=np.int)
k, V_indicies = (list(t) for t in zip(*sorted(zip(np.sqrt(k2), V_indicies))

# k = np.sort(np.sqrt(k2))                               # sort

lam = 2*sp.pi/np.real(k)                                # wavelength
# Q = np.real(k)/(2*np.imag(k))                          # quality factor
```

2.4 Plot eigenmodes

2.4.1 Wavelengths

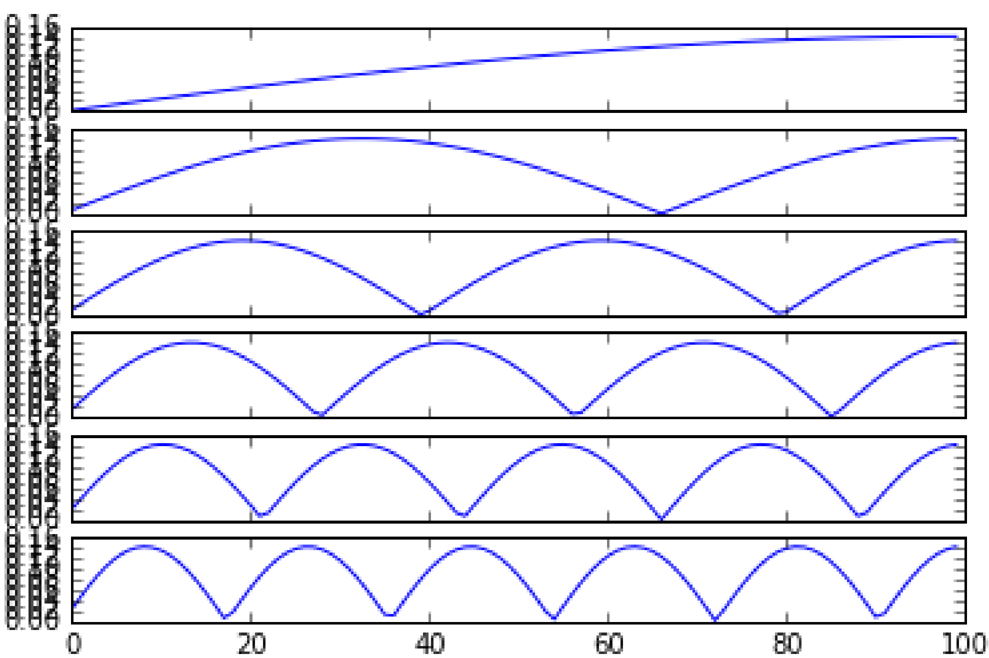
In [5]:
```
plt.plot(np.linspace(1,num_eigs, num_eigs), lam, '-')
plt.plot(np.linspace(1,num_eigs, num_eigs), 4/(2*np.linspace(0,5, num_eigs)
```

Out[5]:
```
[<matplotlib.lines.Line2D at 0x7faade30e898>]
```



2.4.2 Field distribution

```
In [6]: f, ax = plt.subplots(6,1, sharex=True, sharey=True)
        for n in np.arange(V.shape[1]):
            ax[n].plot(np.abs(V[:,V_indicies[n]]),'-')
```



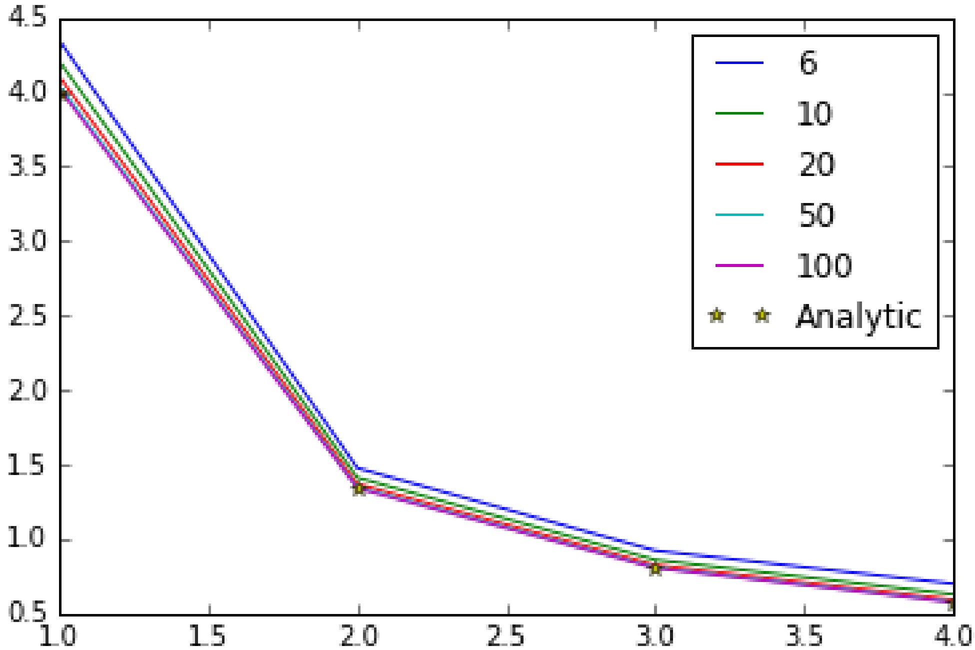
2.5 Discretization artefacts

2.5.1 Solve

In [7]:
```
discret = [6, 10, 20, 50, 100]
eigs = []
field = []
num_eigs = 4

for num in discret:
    # Grid
    n   = num     # Num grid nodes
    dx  = 1. / num  # Step size for overall size of 1
    eps = 1.      #  Vacuum
    mu  = 1.      #  Vacuum
    A   = - 1. / (dx**2 * mu * eps)

    # Matrix
    # Forward

    diag = np.ones(n) * -A
    up_diag = np.ones(n) * A

    M_1 = sp.sparse.dia_matrix(([up_diag, diag], [1, 0]), [n,n])

    # Backward

    diag = np.ones(n) * 1
    up_diag = np.ones(n) * -1

    M_2 = sp.sparse.dia_matrix(([up_diag, diag], [-1, 0]), [n,n])

    M = M_1.dot(M_2)

    # Solve
    kt = 2*sp.pi*1                                    # wave vector target
    k2, V = linalg.eigs(M, k=num_eigs, M=None, sigma=kt**2)  # solve for ei

    V_indicies = np.linspace(0,num_eigs-1, num_eigs, dtype=np.int)
    k, V_indicies = (list(t) for t in zip(*sorted(zip(np.sqrt(k2), V_indici

    lam = 2*sp.pi/np.real(k)                          # wavelength

    eigs.append(lam)
    field.append(V[:,V_indicies])
```

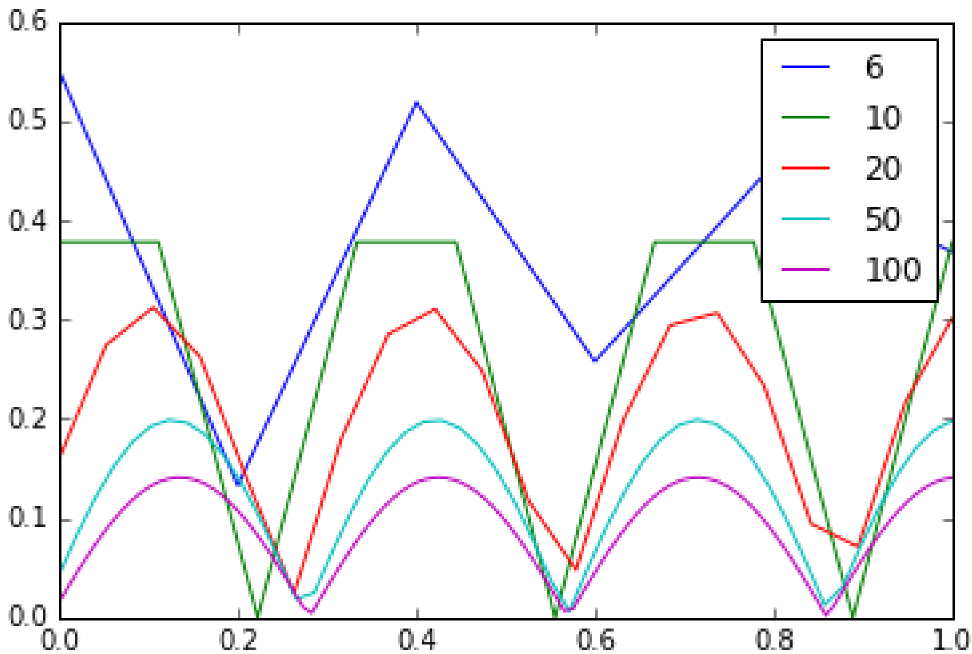
2.5.2 Wavelength

```
In [8]: for i in range(len(eigs)):
            plt.plot(np.linspace(1,num_eigs, num_eigs), eigs[i], '-')

        plt.plot(np.linspace(1,num_eigs, num_eigs), 4/(2*np.linspace(0,num_eigs-1,
        plt.legend([str(s) for s in discret] + ["Analytic"])

Out[8]: <matplotlib.legend.Legend at 0x7faade06e0b8>
```



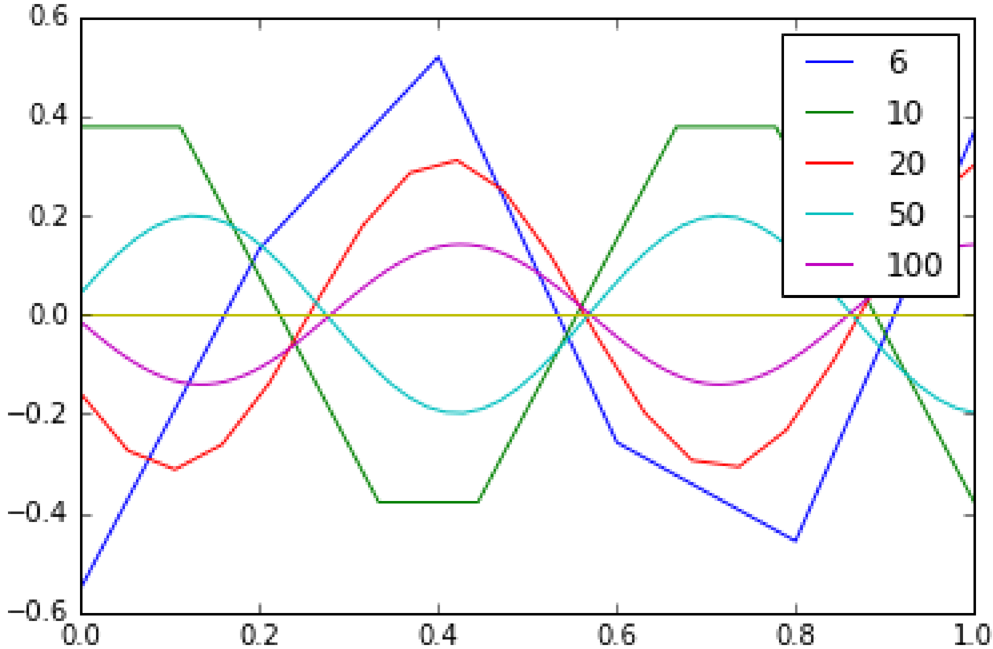
2.5.3 No normalization, amplitude changes

```
In [9]: for f in field:
            #norm = np.max(np.abs(f[:,5]))
            norm = 1.
            plt.plot(np.linspace(0, 1, len(f[:,num_eigs-1])), np.divide(np.abs(f[:,

        plt.legend([str(n) for n in discret])

Out[9]: <matplotlib.legend.Legend at 0x7faade1dfd68>
```



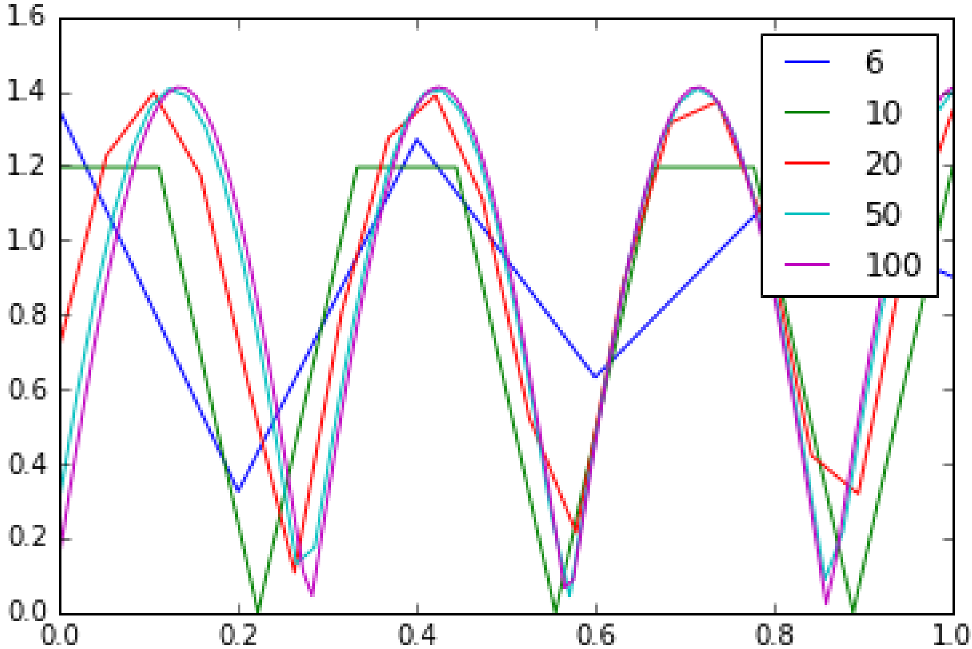
2.5.4 No normalization, phase changes

```
In [10]: for f in field:
             #norm = np.max(np.abs(f[:,5]))
             norm = 1.
             plt.plot(np.linspace(0, 1, len(f[:,num_eigs-1])), np.divide(np.real(f[

         plt.plot([0, 1], [0., 0.])
         plt.legend([str(n) for n in discret])

Out[10]: <matplotlib.legend.Legend at 0x7faade038400>
```



2.5.5 Normalization

```
In [11]: for f, n in zip(field, discret):
             #norm = np.max(np.abs(f[:,5]))
             norm = 1/np.sqrt(n)
             plt.plot(np.linspace(0, 1, len(f[:,num_eigs-1])), np.divide(np.abs(f[:

         plt.legend([str(n) for n in discret])

Out[11]: <matplotlib.legend.Legend at 0x7faadde2e518>
```




```
In [ ]:
```