example_MDOF

April 23, 2015

```
In [1]: %matplotlib inline
        import matplotlib.pyplot as pl
        from scipy import *
        from scipy.linalg import eigh
```

```
In [2]: %config InlineBackend.figure_format = 'svg'
        import matplotlib as mp
        mp.rcParams['figure.figsize'] = 9,4
```

Our system is a two DOF system, its structural matrices are

$$\boldsymbol{M} = m \begin{bmatrix}2 & 0\\0 & 1\end{bmatrix}, \qquad \boldsymbol{K} = k \begin{bmatrix}3 & -2\\-2 & 2\end{bmatrix}.$$

The system is harmonically loaded,

$$\boldsymbol{p}(t) = p_o \begin{Bmatrix}0\\1\end{Bmatrix} \sin\omega t = p_o\,\boldsymbol{r}\sin\omega t,$$

where we have introduced an adimensional *load shape* vector.

A particular integral can be $\boldsymbol{x}_{ss} = \boldsymbol{\xi}\sin\omega t$, substituting in the equation of motion and simplifying the time dependency

$$(k\boldsymbol{K} - \omega^2 m\boldsymbol{M})\,\boldsymbol{\xi} = p_o\,\boldsymbol{r}.$$

Introducing the unit frequency, defined in terms of unit stiffness and unit mass, $\omega_o = \sqrt{\frac{k}{m}}$, with $\omega = 2\omega_o$, dividing both members by k we have

$$(\boldsymbol{K} - 4\boldsymbol{M})\,\boldsymbol{\xi} = \frac{p_o}{k}\,\boldsymbol{r} = \Delta\,\boldsymbol{r}.$$

Substituting the numerical values, solving for $\boldsymbol{\xi}$ and substituting in $\boldsymbol{x}_s$

$$\boldsymbol{x}_s(t) = \Delta\frac{1}{6}\begin{Bmatrix}+2\\-5\end{Bmatrix}\sin\omega t.$$

If our systems starts from rest conditions, the steady state solution has a non zero initial velocity, so we have to superpose a homogeneous solution to get the respect of all initial conditions.

To find the homogeneous solution we use separation of variables, $\boldsymbol{x} = \boldsymbol{\psi}\sin\omega t$, substituting $x(t)$ in the equation of free vibrations and simplifying the time dependency we have the following homogeneous equation

$$(\boldsymbol{K} - \omega^2\boldsymbol{M})\boldsymbol{\psi} = \boldsymbol{0}.$$

The non trivial solutions can be found using the library function `eigh`, that computes an 1-D array of eigenvalues and a 2-D array of (mass-normalized) eigenvectors.

```
In [3]: K = matrix('3 -2;-2 2') ; M = matrix('2 0;0 1')
        evals, evecs = eigh(K,M)
        print evals
        print
        print evecs
```

```
[ 0.31385934  3.18614066]

[[-0.54177432 -0.45440135]
 [-0.64262055  0.76618459]]
```

Note that both `evals` and `evecs` are adimensional, the dimensional eigenvalues can be obtained multiplying `evals` by ω_o^2.

While we are at it, we compute also the adimensional frequencies, ω_i/ω_o, and the inverses of the adimensional frequencies.

```
In [4]: freqs = sqrt(evals)
        invfreq = 1/freqs
        print freqs
        print invfreq
```

```
[ 0.5602315   1.78497638]
[ 1.78497638  0.5602315 ]
```

The first eigenvector has both components negative, I don't like that...

```
In [5]: evecs[:,0] *= -1
```

Now, the load vector divided p_o and the excitation frequency, ω_f/ω_o:

```
In [6]: p  = matrix('0;1')
        wf = 2.0
```

First, we compute $\boldsymbol{\xi}/\Delta$, then its components in the modal coordinates...

```
In [7]: xi = (K-wf*wf*M).I*p
        print xi
        qs = evecs.T*M*xi
        print qs
```

```
[[ 0.33333333]
 [-0.83333333]]
[[-0.17433425]
 [-0.94142139]]
```

The steady state response, in modal coordinates, is

$$x_s(t) = \Delta\,(\boldsymbol{\psi}_1 q_{s,1} + \boldsymbol{\psi}_2 q_{s,2}) \sin\omega t$$

and for initial rest conditions we have

$$q_i(t) = \Delta\left(\sin\omega t - \frac{\omega}{\omega_i}\sin\omega_i t\right) q_{s,i}.$$

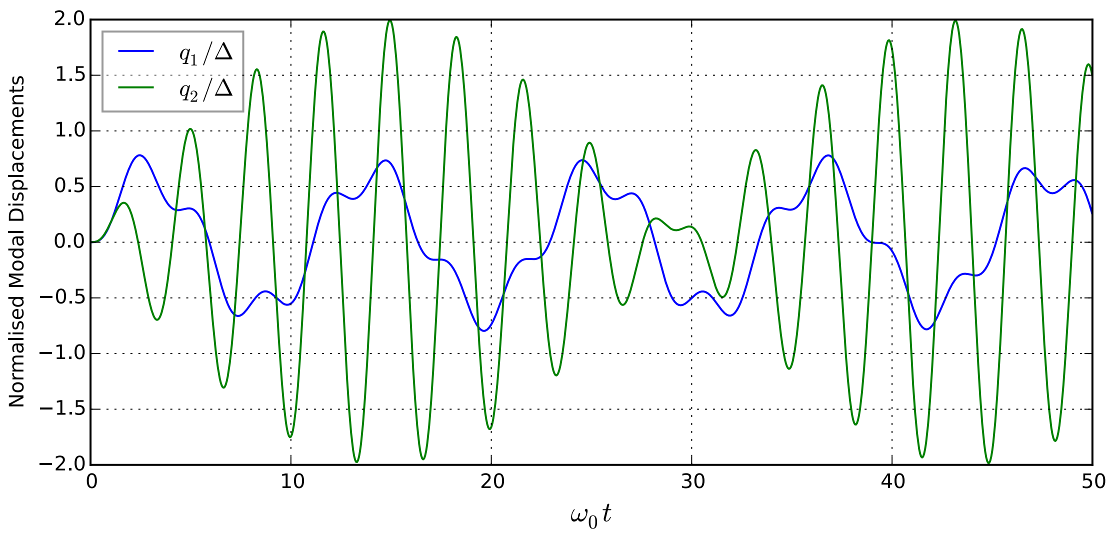
In code:

```
In [8]: def q1(t): return qs[0,0]*(sin(wf*t) - wf/freqs[0]*sin(freqs[0]*t))
        def q2(t): return qs[1,0]*(sin(wf*t) - wf/freqs[1]*sin(freqs[1]*t))
```

It's high time to represent our results, here the response is plotted in terms of adimensional modal coordinates vs adimensional time.

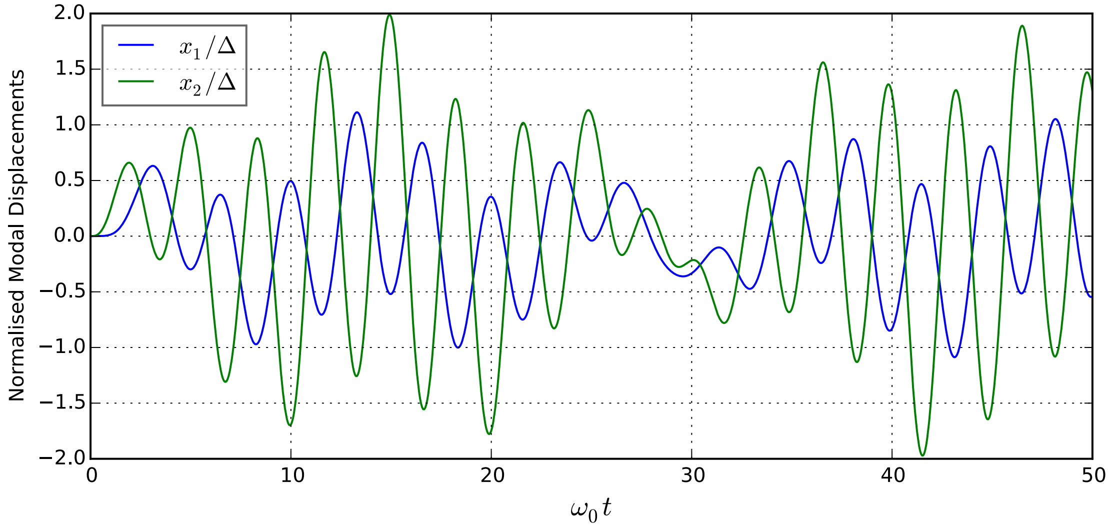
It is worth noting that the second frequency is rather close to the frequency of the harmonmic loading, and this is clearly reflected in the *beating* behaviour of the second modal coordinate.

In [12]:
```
t = linspace(0,50,1001)

pl.xlabel(r'$\omega_0t$',size=14)
pl.ylabel(r'Normalised Modal Displacements')
pl.plot(t,q1(t), label=r'$q_1/\Delta$')
pl.plot(t,q2(t), label=r'$q_2/\Delta$')
pl.legend(framealpha=0.4,loc=0) ; pl.grid()
```



Now, we define quite naively the response in natural coordinates and proceed to plotting in a manner similar to our previous graph.

In [13]:
```
def x1(t): return evecs[0,0]*q1(t)+evecs[0,1]*q2(t)
def x2(t): return evecs[1,0]*q1(t)+evecs[1,1]*q2(t)

pl.xlabel(r'$\omega_0t$',size=14)
pl.ylabel(r'Normalised Modal Displacements')
pl.plot(t,x1(t), label=r'$x_1/\Delta$')
pl.plot(t,x2(t), label=r'$x_2/\Delta$')
pl.legend(framealpha=0.6,loc=0);
pl.grid();
```




```
In []:
```