Ex2_MDOF

April 23, 2015

```
In [1]: %matplotlib inline
        import matplotlib.pyplot as pl
        from scipy import array, linspace, matrix, sin, sqrt, set_printoptions
        from scipy.linalg import eigh
        nl = '\n'
```

0.1 Multi Degrees of Freedom System

The system and the load are described by the structural matrices, the load vector and the frequency of the sinusoidal excitation.

Everything is adimensionalised, `K` and `M` with respect to a unit stiffness and a unit mass `k` and `m`, `p` with respect to a reference load `p_0` and `w` with respect to `w_0 = sqrt(k/m)`.

```
In [2]: K = matrix("3 -2; -2 2")
        M = matrix("2  0;  0 1")
        p = matrix("0;1")
        w = 2.0
```

The eigenvalues and the eigenvectors, the eigenvalues are adimensionalised with respect to `w_0^2 = k/m`, the eigenvectors are normalised with respect to the mass matrix, `Psi.T*M*Psi = I`, `I` being the unit matrix. The eigenvectors are returned sorted from lowest to highest.

Immediatly we compute the starred matrices and the starred load, and also the ratio of the load to the modal stiffnesses.

```
In [3]: e_vals, Psi = eigh(K, M)

        M_star = Psi.T*M*Psi
        K_star = Psi.T*K*Psi
        p_star = Psi.T*p
        p_efff = K_star.I*p_star
```

Let's look at `M_star`

```
In [4]: set_printoptions(suppress=False)
        print M_star,nl
        print 150E9*5E-17
```

```
[[  1.00000000e+00  -5.55111512e-17]
 [ -5.55111512e-17   1.00000000e+00]]

7.5e-06
```

As you can see, `M_star` is not exactly the unit matrix, but to put in perspective the amount of error I've printed also the error multiplied by the Sun Earth distance, in metres. The error stays to unity as an 1/100 of mm stays to the distance between Sun and Earth.

Let's use an option to suppress the printing of such small values and print all the relevant quantities we've just computed.

```
In [5]: set_printoptions(suppress=True)
        print M_star,nl
        print K_star,nl
        print p_star,nl
        print p_efff
```

```
[[ 1. -0.]
 [-0.  1.]]

[[ 0.31385934  0.        ]
 [ 0.          3.18614066]]

[[-0.64262055]
 [ 0.76618459]]

[[-2.04747947]
 [ 0.24047419]]
```

The modal frequencies, the modal frequencies ratio and the modal dynamic amplification factors

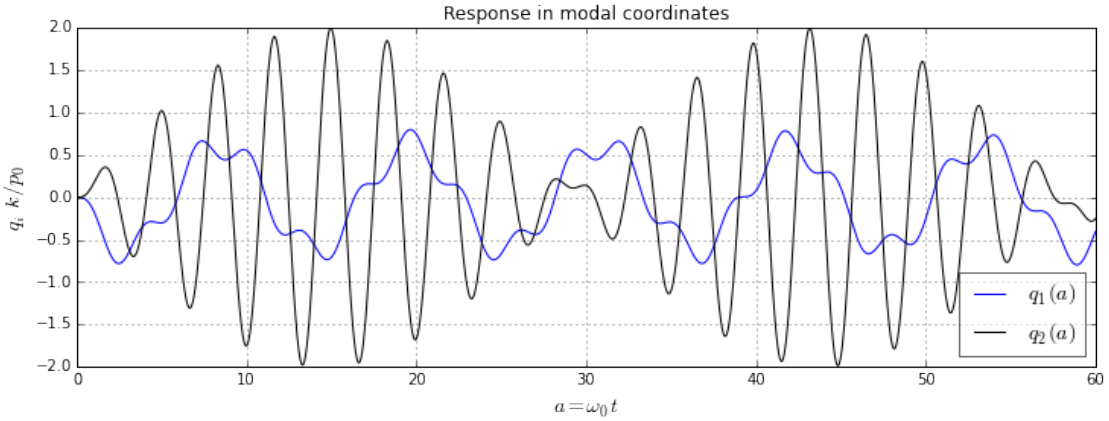
```
In [6]: ww1 = K_star[0,0]   ; ww2 = K_star[1,1]
        w1  = sqrt(ww1)     ; w2  = sqrt(ww2)
        b1  = w/w1          ; b2  = w/w2
        D1  = 1./(1.-b1*b1) ; D2  = 1./(1.-b2*b2)
```

The response, first in modal coordinates and next in structural coordinates, computed for an adimensional time that's equal to `w_0*t`.

```
In [7]: a = linspace(0.,60.,2001)
        q = array((p_efff[0,0]*(sin(w*a)-b1*sin(w1*a))*D1,
                   p_efff[1,0]*(sin(w*a)-b2*sin(w2*a))*D2))
        x = Psi.dot(q)
```

Eventually we plot the responses.

```
In [8]: pl.figure(figsize=(12,4))
        pl.grid()
        pl.title('Response in modal coordinates')
        pl.xlabel("$a = \\omega_0t$",size=14)
        pl.ylabel("$q_i\\,k/p_0$",size=14)
        pl.plot(a, q[0,:], label="$q_1(a)$", color="blue")
        pl.plot(a, q[1,:], label="$q_2(a)$", color="black")
        pl.legend(fontsize=14, framealpha=0.6, loc='best')
        pl.show()
        pl.savefig('pippo1.pdf')
```




```
<matplotlib.figure.Figure at 0x7f37ce5af110>
```

```
In [9]: pl.figure(figsize=(12,4))
        pl.grid()
        pl.title('Response in structural coordinates')
        pl.xlabel("$a = \\omega_0t$",size=14)
        pl.ylabel("$x_i\\,k/p_0$",size=14)
        pl.plot(a,x[0,:], label="$x_1(a)$", color="blue")
        pl.plot(a,x[1,:], label="$x_2(a)$", color="black")
        pl.legend(fontsize=14, framealpha=0.6, loc='best')
        pl.savefig('pippo2.pdf')
```