Elasto-Plastic_Free_Vibrations

April 1, 2015

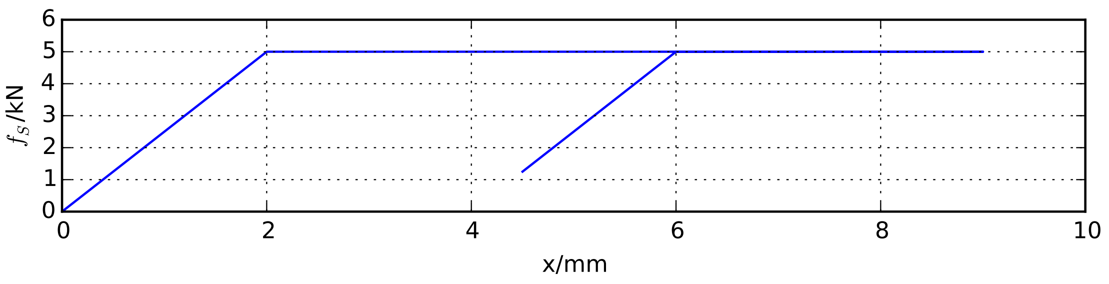
```
In [1]: %pylab inline
        %config InlineBackend.figure_format = 'svg'
        figsize(8,1.5)
```

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

0.1 Elasto-plastic response

In our SDOF system we have a non linear spring, that exhibits zero stiffness in loading after a yielding force is developed, and that unloads with the inelastic stiffness

```
In [2]: plot((0,2,9,6,4.5),(0,5,5,5,1.250))
        axis((0,10,0,6))
        grid()
        xlabel('x/mm')
        ylabel(r'$f_S$/kN');
```



We are going to study such a system under free vibrations, i.e., to study its free vibrations response following given initial conditions.

0.1.1 The system parameters

The parameters of the system, *m, k, z,* and the yielding force are given, all the remaining quantities are computed.

```
In [3]: mass = 16e3
        stif = 2.5e6
        zeta = 0.05

        fy   = 5e3
        uy   = fy/stif
```

```
wn   = sqrt(stif/mass)
wd   = wn*sqrt(1-zeta**2)
damp = 2*wn*mass*zeta

print 'm =', mass, '\tc =',damp, '\tk =', stif
print 'fy =', fy, '\tuy =', uy
print 'wn =',wn, '\twd =', "%8.4f"%(wd,), '\tTn =', 2*pi/wn
```

```
m = 16000.0          c = 20000.0          k = 2500000.0
fy = 5000.0          uy = 0.002
wn = 12.5          wd =  12.4844          Tn = 0.502654824574
```

0.1.2 Initial conditions

The initial conditions have to be assigned

```
In [4]: x0 =  1e-3
        v0 = 25e-3
```

0.1.3 Constants of integration

At start, we are in the elastic range and the elastic homogeneous solution must satisfy the initial conditions.

```
In [5]: # x(0) = A ; v(0) = - zeta wn A + wd B => v(0) + zeta wn x(0) = wd B
        A = x0 ; B = (v0+zeta*wn*A)/wd
        def x_el0(t, A=A, B=B):
            return exp(-zeta*wn*t)*(A*cos(wd*t)+B*sin(wd*t))
        def v_el0(t, A=A, B=B):
            return exp(-zeta*wn*t)*((B*wd-A*wn*zeta)*cos(wd*t)-(B*wn*zeta+A*wd)*sin(wd*t))
```

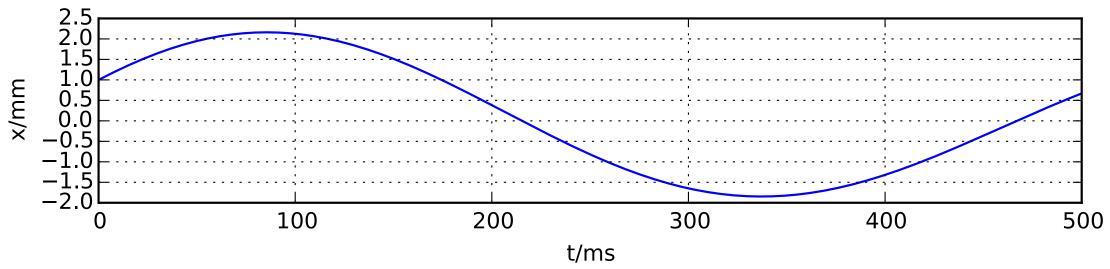
Just to be sure, let's verify that the initial conditions are OK

```
In [6]: x_el0(0), v_el0(0)

Out[6]: (0.001, 0.025000000000000001)
```

Now, let's plot our elastic response and take note that $x_{\text{max}} > 2$
,mm $= u_{\text{y}}$.

```
In [7]: t = linspace(0,0.5, 1001)
        plot(t*1000,1000*x_el0(t))
        grid();xlabel('t/ms');ylabel('x/mm');
```



0.1.4 Limit of validity for elastic response

The solution that we have is valid **only** for $x \leq u_\text{y}$, so we have to find the time t_1 such that $x = u_\text{y}$, and then find x_1 and v_1, the last valid values for our current solution and the initial values for our next solution.

```
In [8]: from scipy.optimize import newton
        t1 = newton(lambda x:x_el0(x)-.002, 0.0)
        x1 = uy
        v1 = v_el0(t1)
        print x1, v1
```

```
0.002 0.0103901221923
```

0.1.5 Equation of motion in plastic range

In the plastic range, until unloading (i.e., for $v \geq 0$), the equation of motion is

$$m\ddot{x} + c\dot{x} + f_\text{S} = 0$$

and being $f_\text{S} = f_\text{y}$ we have

$$m\ddot{x} + c\dot{x} = -f_\text{y};$$

the particular integral is $\xi = -f_\text{y}t/c$ and the general integral can be written

$$x(t) = A\exp(-ct/m) - f_\text{y}t/c + B,$$

where, as usual, the constants of integraton are determined by the initial conditions.

Knowing the constants of integration, you can define the functions that give the displacement and the velocity in the interval from yielding to unloading.

```
In [9]: # x2(t-t1) = x2(t2) = A exp(-c t2 / m) - fy t2 / c + B ; v2(t2) = - c A / m exp(...) - fy/c
        # x2(0) = A+B = x1 ; v2(0) = - A c/m -fy/c = v1
        A = - ( v1 + fy/damp ) * mass / damp
        B = x1 - A
        print A, B
        def x_pl(t, A=A, B=B): return A*exp(-damp*t/mass)-fy*t/damp+B
        def v_pl(t, A=A, B=B): return -damp*A*exp(-damp*t/mass)/mass -fy/damp
```

```
-0.208312097754 0.210312097754
```

0.1.6 Limit of validity for plastic response

Because the initial velocity is positive, the displacements grows until the velocity is, er, positive, and when the velocity equals zero the system changes its state from plastic to elastic and the spring force is $f_S = k\,(x - x_pl)$ (i.e., it's decreasing following the unloading branch).

Using the newton function we find *t2*, the time (NB measured starting from *t1*) at which the velocity is equal to zero,

```
In [10]: t2 = newton(v_pl,t1)
```

so that we can compute the position and the velocity at the end of the plastic phase or, equivalently, at the beginning of the unloading. Also, in the following we'll need the total plastic deformation, so it is computed as the difference between the position at the end and the position at the beginning of the plastic phase.

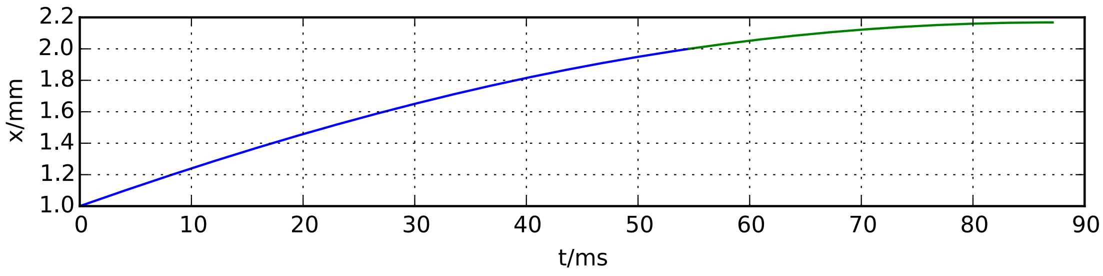
```
In [11]: x2 = x_pl(t2)
         v2 = v_pl(t2)
         dp = x2-x1
         print t1+t2, x2,v2,dp
```

```
0.0871411993189 0.00216808604516 6.66133814775e-16 0.000168086045161
```

To plot the first two phases of the response we have to be careful with time ranges, but it's relatively simple: the E-P response is computed starting from zero, so a vector `t_2` ranging from zero is used in the computation and a vector `t_2+t1` to plot a continuous function of time.

In blue the elastic, loading phase, in green the plastic phase.

```
In [12]: t_1 = linspace(0,t1,1001)
         t_2 = linspace(0,t2,1001)
         plot(1000*t_1,x_el0(t_1)*1000, 1000*(t_2+t1), x_pl(t_2)*1000) ;
         xlabel('t/ms');ylabel('x/mm');
         grid() ;
```



0.1.7 Unloading

Now for the elastic unloading, the elastic force is now $f_\text{S} = k(x - \Delta_\text{pl})$ so that the equation of motion can be written

$m\ddot{x} + c\dot{x} + kx = k\Delta_\text{pl}$

whose general integral is

$x(t) = \exp(-\zeta\omega_n t)\,(A\cos(\omega_\text{D} t) + B\sin(\omega_\text{D}) + \Delta_\text{pl}$.

Next, by the initial conditions we have the constants of integration, and finally the definition of the displacements and velocities in the unloading phase.

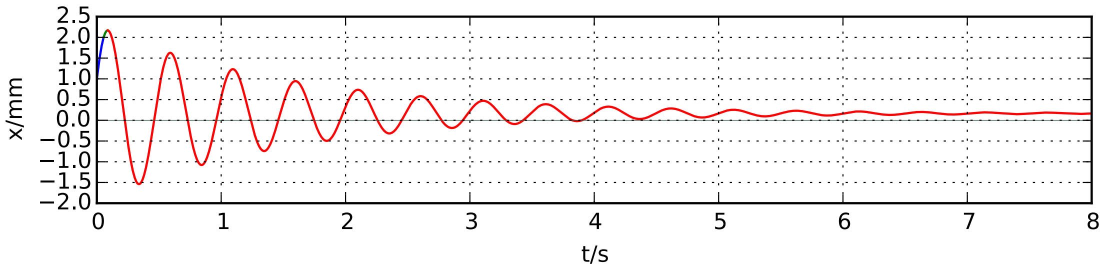
```
In [13]: # x(t3) = exp A cos + exp B sin + dp ; v(t3) = ...
         # x3(0) = A + dp = x2 ; v3(0) = B wd - A z wn = v2
         A = x2-dp
         B = (v2+A*zeta*wn)/wd

         def x_el1(t, A=A, B=B):
             return exp(-zeta*wn*t)*(A*cos(wd*t)+B*sin(wd*t))+dp
         def v_el1(t, A=A, B=B):
             return exp(-zeta*wn*t)*((B*wd-A*wn*zeta)*cos(wd*t)-(B*wn*zeta+A*wd)*sin(wd*t))
```

Eventually, we can plot the free vibrations of our elastic-perfectly plastic system… note that I've stretched the time axis to let the response damp out almost completely, so that you can readily appreciate that the system is not oscillating around the initial equilibrium position, i.e., zero, but is asymptotically reaching a ***new*** position of equilibrium, Δ_pl.

```
In [14]: t_3 = linspace(0,8-t1-t2,2001)
         plot(t_1,x_el0(t_1)*1000,
              t_2+t1, x_pl(t_2)*1000,
              t1+t2+t_3, x_el1(t_3)*1000)
```

```
ylabel('x/mm')
xlabel('t/s')
axhline(linewidth=0.3,color='#335544')
grid() ;
```


x/mm
t/s