NumericalIntegration

Giacomo Boffi

```
from math import *
from IPython.display import HTML
```

Numerical Integration

The data of the problem and a few derived quantities:

```
# system parameters
m =   1315. # kg
c =  15700. # N s / m
k = 325000. # N / m

# p(t) = P sin pi t / t0
P =   1155. # Note that 1155*sin(pi/3) = 1155*sin(2pi/3) ~= 1000
t0 = 0.060 # s
w = pi/t0
Ds = P/k

wn2 = k/m
wn = sqrt(wn2)
z = c/2/m/wn
wd = wn*sqrt(1-z**2)
b = w/wn
```

Analytical Solution

To have a benchmark of our solution we compute the analytical response

$$x(t) = \exp(-\zeta\omega_n t)(A\sin\omega_D t + B\cos\omega_D t) + C\sin\omega t + D\cos\omega t.$$

```
obb, tzb = 1-b**2, 2*z*b
det = obb**2 + tzb**2
# z = C sin wt + D cos wt
C = +Ds*obb/det
D = -Ds*tzb/det
# x = exp(-z wn t) (A sin wd t + B cos wd t) + C sin wt + D cos wt
# x0 = 0, \dot x0 = 0
B = -D
```

```
A = -(z*wn*D+w*C)/wd
# v = exp(-z wn t) [-z wn(A sin wd t + B cos wd t) + wd( A cos wd t - B sin wd t)]
# + w C cos wt - w D cos wt
xt0  = exp(-z*wn*t0)*(A*sin(wd*t0)+B*cos(wd*t0))+C*sin(w*t0)+D*cos(w*t0)
vt0  = exp(-z*wn*t0)*(A*sin(wd*t0)+B*cos(wd*t0))*(-z)*wn
vt0 += exp(-z*wn*t0)*(A*cos(wd*t0)-B*sin(wd*t0))*wd
vt0 += w*C*cos(w*t0)-w*D*sin(w*t0)
print('x(t0) =', xt0*1000, 'mm,   v(t0) =', vt0*1000, 'mm/s')
```

```
x(t0) = 0.7772075829151299 mm,   v(t0) = 20.573849496340785 mm/s
```

```
display(HTML('<h3>Analytical Solution, details</h3>'))
print('Natural period of vibration', 2*pi/wn, 's')
print(' Damped period of vibration', 2*pi/wd, 's')
print('   Period of the excitation', 2*pi/w,  's')
print()
print('Circular frequency  w', w, 'rad/s')
print('Circular frequency wn', wn, 'rad/s')
print('        Damping ratio', z*100, '%')
print(' Frequency ratio beta', b)
print()
print('Static displacement', '%+.3g'%(Ds*1000), 'mm')
print('x(t) = exp(%+.3gt)*[%+.3g*sin(%.3gt)%+.3g*cos(%.3gt)]%+.3g*sin(%.3gt)%+.3g*cos(%.
     (-z*wn, A*1000, wd, B*1000, wd, C*1000, w, D*1000, w))
print()
print('Final displacement', '%+.3g'%(xt0*1000), 'mm')
print('    Final velocity', '%+.3g'%(vt0*1000), 'mm/s')
```

```
<IPython.core.display.HTML object>
```

```
Natural period of vibration 0.39966955254302083 s
 Damped period of vibration 0.4320280745201883 s
   Period of the excitation 0.12 s

Circular frequency  w 52.35987755982989 rad/s
Circular frequency wn 15.720950638348308 rad/s
        Damping ratio 37.972142311087445 %
 Frequency ratio beta 3.330579604525174

Static displacement +3.55 mm
x(t) = exp(-5.97t)*[+1.23*sin(14.5t)+0.083*cos(14.5t)]-0.331*sin(52.4t)-0.08

Final displacement +0.777 mm
    Final velocity +20.6 mm/s
```

Numerical Solution

First the time step depending parameters and the initial conditions.

```
h = 0.02
kn = k + 2*c/h + 4*m/h/h
cn = 2*c + 4*m/h
mn = 2*m

dp01, dp12, dp23 = 1000., 0., -1000.
x0, v0, a0 = 0, 0, 0
```

Next, the computations

```
dx01 = (dp01+v0*cn+a0*mn)/kn
dv01 = 2*(dx01/h-v0)

x1, v1 = x0+dx01, v0+dv01
a1 = (1000-c*v1-k*x1)/m

dx12 = (dp12+v1*cn+a1*mn)/kn
dv12 = 2*(dx12/h-v1)

x2, v2 = x1+dx12, v1+dv12
a2 = (1000-c*v2-k*x2)/m

dx23 = (dp23+v2*cn+a2*mn)/kn
dv23 = 2*(dx23/h-v2)

x3, v3 = x2+dx23, v2+dv23

display(HTML('<h2>Numerical Solution</h2>'))
print('x(t0) =', x3*1000, 'mm,   v(t0) =', v3*1000, 'mm/s')
```

```
<IPython.core.display.HTML object>
```

```
x(t0) = 0.6799599388288293 mm,   v(t0) = 18.745274787261852 mm/s
```

```
print(k/1000, 2*c/h/1000, 4*m/h/h/1000, kn/1000)
print('t [ms]\t\t   x [mm]\t v [mm/s] ')
for t, x, v in ((0, x0, v0), (20, x1, v1), (40, x2, v2), (60, x3, v3)):
    print('%6.3f\t\t%9.3f\t%9.3f'%(t, x*1000, v*1000))
```

```
325.0 1570.0 13150.0 15045.0
t [ms]                     x [mm]         v [mm/s]
 0.000                      0.000            0.000
20.000                      0.066            6.647
40.000                      0.313           17.979
60.000                      0.680           18.745
```