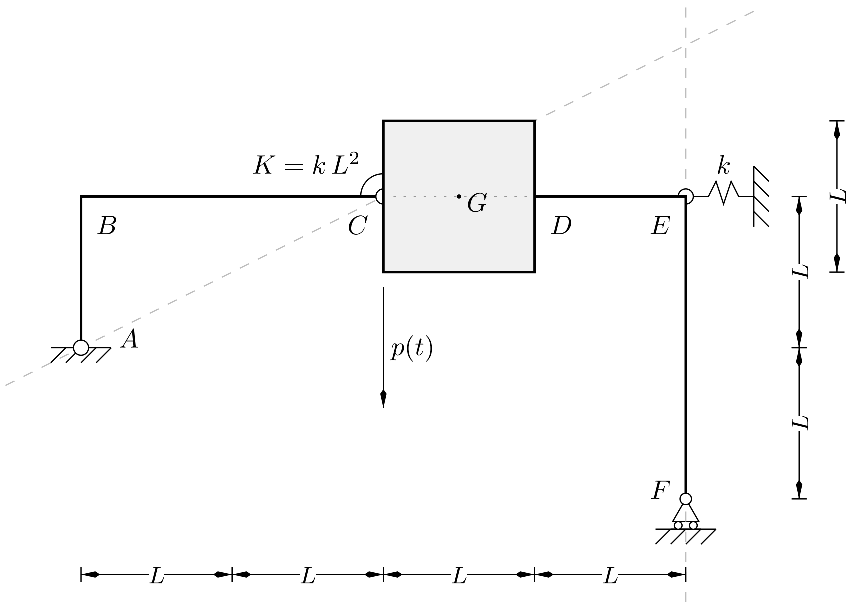
03_Assemblage

May 23, 2014

```
# import some utility functions from IPython
from IPython.display import display, HTML, Latex, SVG
```

1 Statement of the problem

```
from fractions import Fraction as frac
display(SVG(filename="tbodies.svg"))
```



The left body being 1 and the right one being 2, the Centres of Instantaneous Rotations Ω_1 and Ω_2 are the hinge in A and the intersection of the dotted lines.

2 Characterization of the model

2.1 Masses

The mass and the length are just a unit mass and a unit length, the rotatory inertia has to be computed.

Note that we use the `frac` class do to all our computations using exact integer fractions.

```
L = frac(1)
M = frac(1)
J = M*(L**2+L**2)/12
```

2.2 Stiffnesses and external load

The stiffness is a unit stiffness, the rotational stiffness is specified in the figure, the load is a unit load.

```
k   = frac(1)
K   = k*L**2
p   = 1
```

2.3 Displacements and rotations

With the centres of instantaneous rotation as shown in figure, using the horizontal component of the mass displacement as the unit of displacement (hence `xg=1`), we compute the rotations of the two bodies and from the rotations all the interesting components of generalized displacements.

```
xg = frac(1)
# rotations
t2 = xg/L
t1 = -t2
# displacements, x rightwards and y downwards
# extensional spring
xs = t2*L
# center of mass and point of application of p
yg = t2*(L*3/2)
yp = t2*(2*L)
# relative rotation
tK = t1 - t2
```

2.4 Increments and accelerations

We define a function to represent the virtual increment of a displacement component, that simply returns the displacement component, and a function to represent the acceleration, that simply returns the displacement component.

```
def inc(x): return x
def acc(x): return x
```

3 Equation of Motion using the PVD

Let's write the virtual work done by the spring forces, by the inertial forces and by the external forces, then, by the PVD

$$\delta W_S + \delta W_I + \delta W_P = 0$$

```
dws = + (-xs*k) * inc(xs) + (-tK*K) * inc(tK)
dwi = + (-acc(xg)*M) * inc(xg) + (-acc(yg)*M) * inc(yg) + (-acc(t2)*J) * inc(t2)
dwp =    + p * inc(yp)
```

Symplifying δx_G and rearranging, using a overly nice output format we have eventually

```
fmt = r'$$\frac{%d}{%d}\,m\,\ddot{u}_G + \frac{%d}{%d}\,k\,u_G = \frac{%d}{%d}\,p(t).$$'
display(Latex('The equation of motion is\n'))
display(Latex(fmt%(-dwi.numerator, dwi.denominator,
                   -dws.numerator, dws.denominator,
                   +dwp.numerator, dwp.denominator)))
```

The equation of motion is

$$\frac{41}{12}\,m\,\ddot{u}_G + \frac{5}{1}\,k\,u_G = \frac{2}{1}\,p(t).$$