week14–springs

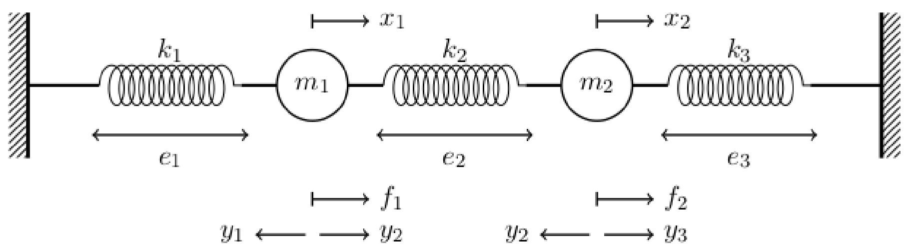
April 18, 2024

1 Modeling spring networks

Examples in "Structural modeling"

1.1 One-dimensional models

Let's consider a linear network of 3 springs and 2 masses:



Here are the variables:

- f_j = applied load or force to mass (in `N` = `Newtons`), for $j = 1, 2$
- k_i = spring constant (in `N/m` = `Newtons per meter`), for $i = 1, 2, 3$
- e_i = elongation of spring i from equilibrium (in `m` = `meters`)
- x_j = displasement of mass j from equilibrium (in `m` = `meters`)
- y_i = restoring force on spring i (in `N` = `Newtons`)

The *"inputs"* are the applied forces f_j which cause the masses to move, resulting in elongation of springs.

We'll take "movement to the right" to be *positive*, and a stretch as *positive* elongation.

Thus we have the equations:

$$e_1 = x_1, \quad e_2 = x_2 - x_1, \quad e_3 = -x_2.$$

(This third equation reflects the fact that spring 3 compresses when m_2 moves to the right.)

Let's put this in matrix form:

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = B\mathbf{x}.$$

Now, let's recall that according to Hooke's Law, the elongation of the spring causes a restoring force on the mass, determined by the *spring constant* $k_i > 0$. Thus we get equations

$$y_j = k_j e_j \quad \text{for } j = 1, 2, 3.$$

In matrix form, these equations read:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = K\mathbf{e}.$$

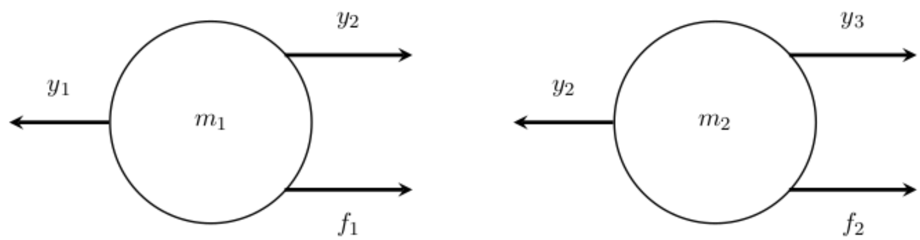
Combining these euqations gives

$$\mathbf{y} = K\mathbf{e} = KB\mathbf{x}.$$

[19]:
```
import numpy as np
import sympy as sp

def sbv(i,n):
    return np.array([1 if j == i else 0 for j in range(n)])

def makeK(kar):
    n = len(kar)
    return np.array([kar[i]*sbv(i,n) for i in range(n)])

def makeB(m,n):
    return np.array([(-1)*sbv(i-1,m) + sbv(i,m) for i in range(n)])

k1, k2, k3 = sp.symbols('k1 k2 k3')

K = makeK([k1,k2,k3])
B = makeB(2,3)
(K,B)
```

[19]:
```
(array([[k1, 0, 0],
        [0, k2, 0],
        [0, 0, k3]], dtype=object),
 array([[ 1,  0],
        [-1,  1],
        [ 0, -1]]))
```

Next, we assume that the system is at rest after the loads are applied (i.e. the forces f_i).



Looking at the diagram, we see that the following equations must hold:

(The first diagram gives:)

$$y_1 = y_2 + f_1 \implies$$
$$y_1 - y_2 = f_1$$

(The second diagram gives:)

$$y_2 = y_3 + f_2 \implies$$
$$y_2 - y_3 = f_2$$

In matrix form this reads

$$\begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix},$$

i.e.

$$B^T \mathbf{y} = \mathbf{f}$$

Combined with our earlier equation

$$\mathbf{y} = K\mathbf{e} = KB\mathbf{x}$$

we now see

$$B^T KB\mathbf{x} = \mathbf{f}.$$

[26]:
```
A=B.transpose() @ K @ B
A
```

[26]:
```
array([[k1 + k2, -k2],
       [-k2, k2 + k3]], dtype=object)
```

Thus we have

$$A = B^T KB = \begin{bmatrix} k_1 + k_2 & -k_2 \\ -k_2 & k_2 + k_3 \end{bmatrix}.$$

[32]:
```
def findDisplacements(kar,far):
    # kar = array of spring constants
    # far = array of initial forces.
```

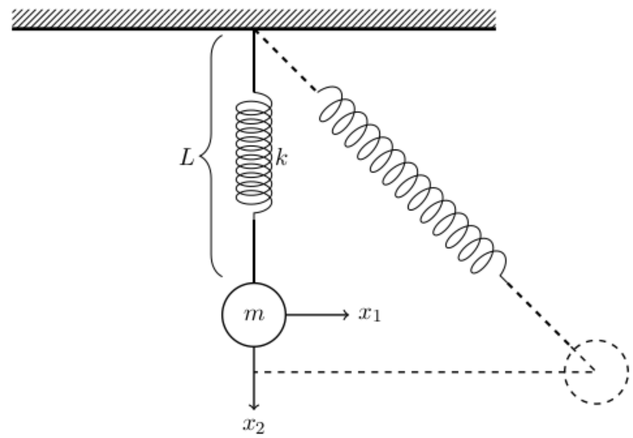
```
    m = len(far)
    n = len(kar)
    B = makeB(m,n)
    K = makeK(kar)
    A = B.transpose() @ K @ B
    f = np.array(far)
    return np.linalg.solve(A,f)

# Let's find the displacements for spring constants `k = [1,1,1]`
# and forces `f = [3,-3]`

findDisplacements([1,1,1],[3,-3])
```

[32]: array([1., -1.])

[37]:
```
findDisplacements([1,1,1],[3,-2])
```

[37]: array([1.33333333, -0.33333333])

1.2 Two dimensional models



Now let's allow the mass to move in two dimensions:

We see in this case that the elongation e satisfies

$$e = \sqrt{x_1^2 + (L + x_2)^2} - L$$

Since this does not express a linear relationship between e and the displacements x_1, x_2, we can't express this relationship using a matrix.

But we can *linearize*. Recall that the linearization (first-order Taylor polynomial) about $t = 0$ of the function $y = \sqrt{1+t}$ is given by

$$(\clubsuit) \quad \sqrt{1+t} \approx 1 + \frac{t}{2} + O(t^2).$$

Let's use this linearization to rewrite the expression for e given above.

We first rewrite

$$\begin{aligned} x_1^2 + (L+x_2)^2 &= x_1^2 + L^2 + 2Lx_2 + x_2^2 \\ &= L^2\left(\frac{x_1^2}{L^2} + 1 + \frac{2x_2}{L} + \frac{x_2^2}{L^2}\right) \\ &= L^2\left(1 + \frac{2x_2}{L} + \frac{x_1^2}{L^2} + \frac{x_2^2}{L^2}\right) \end{aligned}$$

so that

$$\begin{aligned} e &= \sqrt{x_1^2 + (L+x_2)^2} - L \\ &= \sqrt{L^2\left(1 + \frac{2x_2}{L} + \frac{x_1^2}{L^2} + \frac{x_2^2}{L^2}\right)} - L \\ &= L\sqrt{1 + \frac{2x_2}{L} + \frac{x_1^2}{L^2} + \frac{x_2^2}{L^2}} - L \end{aligned}$$

Now taking $t = \frac{2x_2}{L} + \frac{x_1^2}{L^2} + \frac{x_2^2}{L^2}$ the approximation ($\clubsuit$) gives

$$\begin{aligned} e &\approx L\left(1 + \frac{1}{2}t\right) - L \\ &= L\left(1 + \frac{1}{2}\frac{2x_2}{L} + \frac{x_1^2}{L^2} + \frac{x_2^2}{L^2}\right) - L \\ &= x_2 + \frac{x_1^2 + x_2^2}{2L} \end{aligned}$$

Now, this is of course still not a linear relationship between e and x_1, x_2. Note that the approximation ($\clubsuit$) depends on the assumption that $t = \frac{2x_2}{L} + \frac{x_1^2}{L^2} + \frac{x_2^2}{L^2} \approx 0$.

If we suppose that the displacements x_1, x_2 are small compared to the resting length L of the spring, then $x_1^2 + x_2^2$ is even smaller compared to L, so making one more approximation, we eliminate the quadratic term and so we get

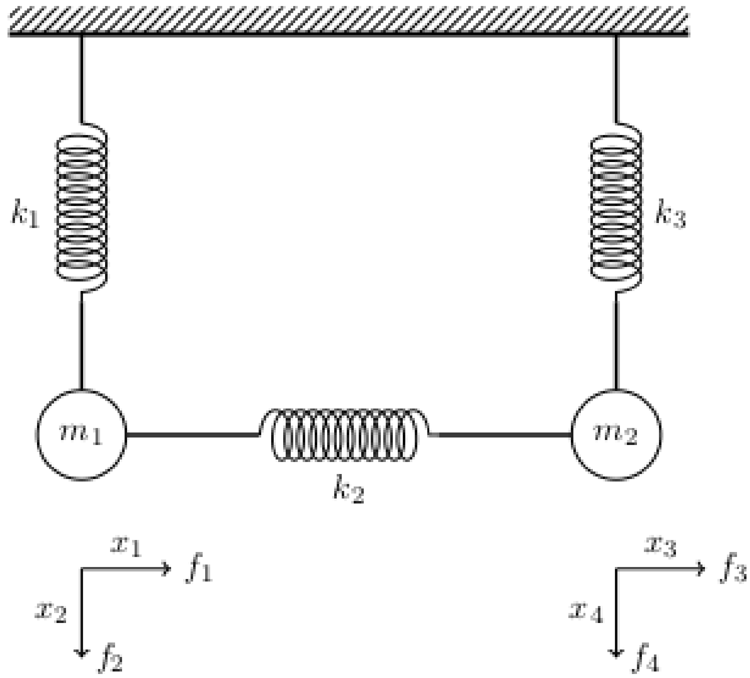
$$e \approx x_2.$$

Remark The approximation $e \approx x_2$ is equivalent to a small-angle approximation. It essentially says that the horizontal displacements are negligible in the elongation, but vertical displacements are important.

Remark This assumption is of course **"wrong"**, but can still be useful. Especially, it is now *linear*

1.3 Spring networks

Let's consider a network of springs in the two-dimensional setting.



$$
\begin{aligned}
e_1 &= x_2 \\
e_2 &= x_3 - x_1 \\
e_3 &= x_4
\end{aligned}
$$

The *linearization* discussed in the previous section says that the elongation is determined by the displacement in the "1 dimensional direction of the spring".

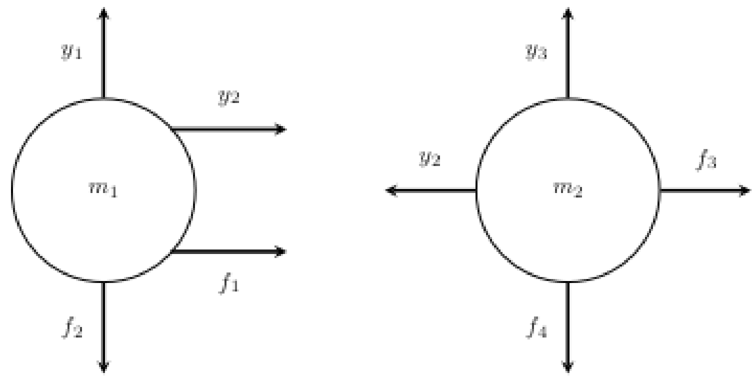
We get the matrix equation

$$
\mathbf{e} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = B\mathbf{x}.
$$

As before, Hooke's Law gives

$$
\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} k_1 & 0 & 0 \\ 0 & k_2 & 0 \\ 0 & 0 & k_3 \end{bmatrix} \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = K\mathbf{e}.
$$

We balance the forces:



$$
\begin{aligned}
y_1 &= f_2 \\
-y_2 &= f_1 \\
y_2 &= f_3 \\
y_3 &= f_4
\end{aligned}
$$

This gives

$$
\begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{y} = \mathbf{f}
$$

i.e.

$$
B^T \mathbf{y} = \mathbf{f}.
$$

[ ]: