2D spring-mass systems in equilibrium

Vector notation preliminaries

First, we summarize 2D vector notation used in the derivations for the spring system.

We use $\|a\|$ to denote the length of a vector $\mathbf{a}$, $\|\mathbf{a}\| = \sqrt{a_x^2 + a_y^2}$. The *outer product* $\mathbf{a}\mathbf{b}^T$ of two vectors $\mathbf{a}$ and $\mathbf{b}$ is a matrix

$$\begin{bmatrix} a_x b_x & a_x b_y \\ a_y b_x & a_y b_y \end{bmatrix}$$

Differentiating a scalar function $f(\mathbf{p})$ of a vector argument $\mathbf{p} = (x, y)$, with respect to $\mathbf{p}$ means calculating the vector

$$\frac{\partial f}{\partial \mathbf{p}} = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right].$$

If we differentiate a vector function with respect to a vector we get a *matrix* (one can think about this as differentiating the vector function component by component, and stacking two resulting row vectors together); if the components of $\mathbf{f}(\mathbf{p})$ are $f_x(x, y)$ and $f_y(x, y)$, then

$$\frac{\partial \mathbf{f}}{\partial \mathbf{p}} = \begin{bmatrix} \frac{\partial f_x}{\partial x} & \frac{\partial f_x}{\partial y} \\ \frac{\partial f_y}{\partial x} & \frac{\partial f_y}{\partial y} \end{bmatrix}$$

This notation makes linear approximations to vector functions look exactly the same way as in the scalar case: instead of

$$f(x) \approx f(x_0) + f'(x)(x - x_0)$$

we get

$$\mathbf{f}(\mathbf{p}) \approx \mathbf{f}(\mathbf{p}_0) + \frac{\partial \mathbf{f}}{\partial \mathbf{p}}(\mathbf{p} - \mathbf{p}_0)$$

where the second term is a matrix-vector product. Several useful identities:

- $$\frac{\partial \mathbf{p}}{\partial \mathbf{p}} = I$$

  that is, if $f_x(x, y) = x$ and $f_y(x, y) = y$, the matrix $\partial \mathbf{f}/\partial \mathbf{p}$ is just the identity matrix.

- For a scalar function g, and vector function $\mathbf{f}$, we get the product rule:

  $$\frac{\partial g\mathbf{f}}{\partial \mathbf{p}} = g\frac{\partial \mathbf{f}}{\partial \mathbf{p}} + \mathbf{f}\frac{\partial g}{\partial \mathbf{p}}^T$$

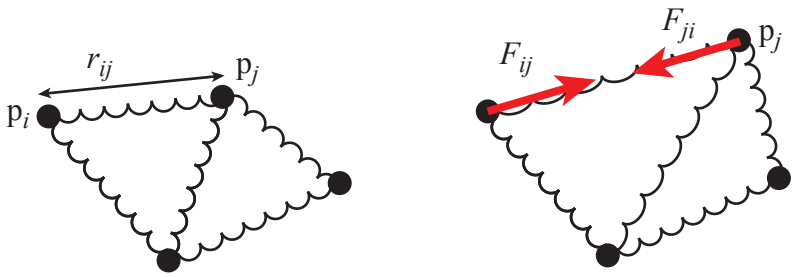
  where the second term is an outer product. This can be easily verified applying the definitions of the derivative of the vector function, the usual scalar derivative product rule, and the outer product definition.

- Chain rule for two vector functions $\mathbf{g}(\mathbf{p})$ and $\mathbf{f}(\mathbf{q})$:

$$\frac{\partial \mathbf{f}(\mathbf{g}(\mathbf{p}))}{\partial \mathbf{p}} = \frac{\partial \mathbf{f}}{\partial \mathbf{q}} \frac{\partial \mathbf{g}}{\partial \mathbf{p}}$$

  where the right-hand side is the matrix product.

Spring equations



Rest configuration *One point moved*

A *spring-mass system* is a collection of point masses m_i with positions $\mathbf{p}_i$ connected by springs. Some of the points are fixed, some are allowed to move. Our goal is to find positions of the moving points for which the total force from springs acting on each point is zero, in other words, the system is in its rest (equilibrium) state to which it relaxes if no external forces are applied.

A spring (i, j) connects two points $\mathbf{p}_i$ and $\mathbf{p}_j$. Each spring has a *rest length*, r_{ij}. If the distance $\|\mathbf{p}_i - \mathbf{p}_j\|$ between spring endpoints is r_{ij}, there there is *no* contribution from the spring to the total force acting on either point. If the distance is different, there is a *spring force* acting on each endpoint: the magnitude of the force acting on $\mathbf{p}_i$, is $k\|\mathbf{p}_i - \mathbf{p}_j\|$, and it acts along the spring, that is, the line connecting spring endpoints; the coefficient k is called *stiffness*. For simplicity, we assume that all springs have the same stiffness. In vector form, the force can be written as

$$\mathbf{F}_{ij} = k(\|\mathbf{p}_i - \mathbf{p}_j\| - r_{ij}) \frac{\mathbf{p}_j - \mathbf{p}_i}{\|\mathbf{p}_i - \mathbf{p}_j\|}$$

In this expression, $\mathbf{p}_j - \mathbf{p}_i / \|\mathbf{p}_i - \mathbf{p}_j\|$ is just a unit vector pointing along the spring *away* from $\mathbf{p}_i$.

We denote $neighbor(i)$ the set of indices of points connected to point i by a spring. Then the total force acting on a point i is

$$\mathbf{F}_i = \sum_{j \in neighbor(i)} \mathbf{F}_{ij} = \sum_{j \in neighbor(i)} k(\|\mathbf{p}_i - \mathbf{p}_j\| - r_{ij}) \frac{\mathbf{p}_j - \mathbf{p}_i}{\|\mathbf{p}_i - \mathbf{p}_j\|} \qquad (1)$$

Let I_f be the set of indices of points that are fixed, and I_m be the set of indices of moving points; we assume that all moving points precede in the list of points all fixed points, that is, the moving points have indices $I_m = \{0 \dots N_m - 1\}$, and fixed points $I_f = \{N_m \dots N_m + N_f - 1\}$. Later we will discuss how to eliminate this assumption.

We build a vector $\mathbf{F}$ of length $2N_m$, which is obtained by concatenating 2D vectors of forces $\mathbf{F}_i$ for all moving points. Similarly, we concatenate all moving point positions $\mathbf{p}_i$ to get a vector $\mathbf{p}$.

Then the equation we need to solve has the form

$$\mathbf{F}(\mathbf{p}) = 0 \tag{2}$$

One can see that for the equilibrium problem, the actual mass of each point does not matter, because the forces only depend on spring stiffnesses (for the dynamic problem, $\mathbf{F}_i = m_i \mathbf{a}_i$, where $\mathbf{a}_i$ is the acceleration of i-th point, the mass is important).

These equations are nonlinear, so we cannot solve this system using, for example, LU decomposition.

Solving the system using Newton's method

The Newton method iteration for (2) (See Chapter 9 of the textbook for details) is

$$\mathbf{p}^{k+1} = \mathbf{p}^k - J[\mathbf{F}](\mathbf{p}^k)^{-1}\mathbf{F}(\mathbf{p}^k)$$

where $J[\mathbf{F}](\mathbf{p}_k)$ is the *Jacobian matrix* of $\mathbf{F}$ evaluated for point positions $\mathbf{p}_k$. The Jacobian matrix is the matix of partial derivatives with entry (s,t) equal to $\partial F_s/\partial p_t$, where $s,t = 0 \dots 2N_m - 1$, are indices in vectors $\mathbf{F}$ and $\mathbf{p}$. Note that this is a more general case of the 2×2 derivative matrix $\partial \mathbf{f}/\partial \mathbf{p}$.

It is convenient to view each Newton step as solving a linear system

$$J[\mathbf{F}]\Delta\mathbf{p}^{k+1} = -\mathbf{F}(\mathbf{p}^k)$$

where $\Delta\mathbf{p}^{k+1} = \mathbf{p}^{k+1} - \mathbf{p}^k$.

We already know how to obtain the vector $\mathbf{F}(\mathbf{p}^k)$; so the only other element that we need is the Jacobian matrix.

Assembling the Jacobian matrix. It is convenient to assemble this matrix from 2 blocks, obtained by differentiating individual spring forces $\mathbf{F}_{ij}$. Instead of treating $\mathbf{p}$ as a flat vector $[p_1^x, p_1^y, p_2^x, p_2^y, \dots]$, we regard it as a vector of points $[\mathbf{p}_1, \mathbf{p}_1, \mathbf{p}_2 \dots]$, and take the same approach to $\mathbf{F}$. Now we build $J[\mathbf{F}]$ as a $N_m \times N_m$ matrix of 2×2 blocks $\partial \mathbf{F}_i/\partial \mathbf{p}_j$, $i,j = 0 \dots N_m - 1$.

An individual block $\partial \mathbf{F}_i/\partial \mathbf{p}_j$ can be interpreted as the rate of change of the force acting on point $\mathbf{p}_i$ due to the change in position of point $\mathbf{p}_j$.

First, we consider the case when $i \neq j$. In this case, in the sum $\mathbf{F}_i = \sum_l \mathbf{F}_{il}$, at most one term depends on $\mathbf{p}_j$: if there is a spring connecting $\mathbf{p}_i$ with $\mathbf{p}_j$, then $\mathbf{F}_{ij}$ depends on j, and the rest do not, and if there is no spring, no terms depend on j. So we get, for $i \neq j$,

$$\frac{\partial \mathbf{F}_i}{\partial \mathbf{p}_j} = \begin{cases} \frac{\partial \mathbf{F}_{ij}}{\partial \mathbf{p}_j}, \text{ if there is a spring } (i,j) \\ 0 \text{ otherwise.} \end{cases}$$

If $i = j$, all terms in the sum depend on $\mathbf{p}_j = \mathbf{p}_i$, so the derivative will be the sum:

$$\frac{\partial \mathbf{F}_i}{\partial \mathbf{p}_i} = \sum_{l \in neighbor(i)} \frac{\partial \mathbf{F}_{il}}{\partial \mathbf{p}_i} \tag{3}$$

On the other hand, we observe that $\mathbf{F}_{il}$ depends only on the difference $\mathbf{d}_{il} = \mathbf{p}_i - \mathbf{p}_l$, so we can write it as a composition of functions $\mathbf{F}(\mathbf{d}_{il}(\mathbf{p}_i - \mathbf{p}_l))$. As $\partial \mathbf{d}_{il}/\partial \mathbf{p}_i = I$ and $\partial \mathbf{d}_{il}/\partial \mathbf{p}_l = -I$, by the chain rule, we can see that $\partial \mathbf{F}_{il}/\partial \mathbf{p}_i = -\partial \mathbf{F}_{il}/\partial \mathbf{p}_l$.

Based on this, we see that we can rewrite the expression (3) as

$$\frac{\partial \mathbf{F}_i}{\partial \mathbf{p}_i} = - \sum_{l \in neighbor(i)} \frac{\partial \mathbf{F}_{il}}{\partial \mathbf{p}_l}$$

There is a subtlety here related to fixed points: some of the points $\mathbf{p}_l$ attached to the moving point $\mathbf{p}_i$ may be fixed; we still need to include terms $\partial \mathbf{F}_{il}/\partial \mathbf{p}_l$ in the summation above, although $\mathbf{p}_l$ for a fixed point is not changing.

So all blocks in the matrix can be obtained as soon as we know $\partial \mathbf{F}_{ij}/\partial \mathbf{p}_j$.

Computing force derivative for an individual spring. To compute this derivative, we rewrite $\mathbf{F}_{ij}$ as sum of two terms

$$\mathbf{F}_{ij} = k(\mathbf{p}_j - \mathbf{p}_i) - kr_{ij} \frac{\mathbf{p}_j - \mathbf{p}_i}{\|\mathbf{p}_i - \mathbf{p}_j\|}$$

The derivative of the first term with respect to $\mathbf{p}_j$ is kI. The derivative of the second term requires a bit more effort. We compute it using the chain rule: denote $\mathbf{d} = \mathbf{p}_j - \mathbf{p}_i$. Then we can write the second term as

$$\mathbf{G}_{ij} = kr_{ij} \frac{\mathbf{d}}{\|\mathbf{d}\|} = kr_{ij} \frac{\mathbf{d}}{\sqrt{\mathbf{d} \cdot \mathbf{d}}}$$

The complete derivative is obtained as $(\partial \mathbf{G}_{ij}/\partial \mathbf{d})(\partial \mathbf{d}/\partial p_j)$. But $\partial \mathbf{d}/\partial p_j = \partial \mathbf{p}_j/\partial p_j - \partial \mathbf{p}_j/\partial p_i = I - 0 = I$, so our derivative is simply $\partial \mathbf{G}_{ij}/\partial \mathbf{d}$. As kr_{ij} is constant, it remains to compute $\partial(\mathbf{d}/\|\mathbf{d}\|)/\partial \mathbf{d}$, which we do using the product rule:

$$\frac{\partial \|\mathbf{d}\|^{-1}\mathbf{d}}{\partial \mathbf{d}} = \|\mathbf{d}\|^{-1} \frac{\partial \mathbf{d}}{\partial \mathbf{d}} + \mathbf{d} \left(\frac{\partial \|\mathbf{d}\|^{-1}}{\partial \mathbf{d}} \right)^T$$

The derivative $\frac{\partial (\mathbf{d} \cdot \mathbf{d})^{1/2}}{\partial \mathbf{d}}$ is $(-1/2)(\mathbf{d} \cdot \mathbf{d})^{-3/2} \frac{\partial \mathbf{d} \cdot \mathbf{d}}{\partial \mathbf{d}} = -\|\mathbf{d}\|^{-3}\mathbf{d}$, so we get the expression

$$\frac{\partial \|\mathbf{d}\|^{-1}\mathbf{d}}{\partial \mathbf{d}} = \frac{\|\mathbf{d}\|^2 I - \mathbf{d}\mathbf{d}^T}{\|\mathbf{d}\|^3}$$

And the final expression for the derivative of the force $\mathbf{F}_{ij}$ with respect to the point position $\mathbf{p}_j$ is

$$J_{ij} = \frac{\partial \mathbf{F}_{ij}}{\partial \mathbf{p}_j} = kI - kr_{ij} \frac{\|\mathbf{d}_{ij}\|^2 I - \mathbf{d}_{ij}\mathbf{d}_{ij}^T}{\|\mathbf{d}_{ij}\|^3}$$

where $\mathbf{d}_{ij} = \mathbf{p}_j - \mathbf{p}_i$.

An additional observation we can make about these expression is that $J_{ij} = J_{ji}$, because $\mathbf{d}_{ij} = -\mathbf{d}_{ji}$, and sign changes of $\mathbf{d}$ cancel in the expressions. This allows us to compute only J_{ij} for $j < i$, and use it for two subblocks of the matrix.

By-spring assembly. The total forces $\mathbf{F}_i$ acting on each point can be computed if we iterate over all points and compute the sums (1) for each. This would require additional data structures for adjacency information, and a loop for each point. Alternatively, we observe that we can have a loop over *springs* (i, j), and add forces due to each spring to the parts of the total force vector corresponding to points $\mathbf{p}_i$ and $\mathbf{p}_j$:

```
for each spring (i,j), i < j
    compute F_ij
    if i < N_m add F_ij to F_i
    if j < N_m add F_ij to F_j.
```

Here we take advantage of the fact that $\mathbf{F}_{ij} = -\mathbf{F}_{ji}$ to compute the force only once.

Similarly, instead of assembling the matrix $J[\mathbf{F}]$ using a double loop over $\mathbf{F}_s$ for rows and $\mathbf{p}_t$ for columns, we can compute $\partial \mathbf{F}_{ij}/\partial \mathbf{p}_j$ for each spring (i, j), and add it to the 2×2 block in the matrix corresponding to the pair of points (i, j). As we know that the coordinates of i-th moving point occupy slots $2i$ and $2i + 1$ in the vectors $\mathbf{F}$ and $\mathbf{p}$, thanks to the assumption that all moving points precede fixed, the block J_{ij} corresponds to the matrix subblock in rows $2i, 2i + 1$ and columns $2j, 2j + 1$.

This leads to the following algorithm that can be implemented with just several lines of code, where we use `numpy` notation for matrix subblock assignment:

```
Initialize the matrix J to 2N_m × 2N_m zero matrix
foreach spring (i, j), i < j
   compute J_ij = ∂F_ij / ∂p_j
   if i < N_m
      # update the sum for the diagonal block, if i is not fixed
      J[2i : 2i + 2, 2i : 2i + 2] += −J_ij
   if i < N_m and j < N_m,
      # set off-diagonal blocks if neither point is fixed
      J[2i : 2i + 2, 2j : 2j + 2] = J_ij
      J[2j : 2j + 2, 2i : 2i + 2] = J_ij
```

An alternative way of handling fixed points. If we fix and unfix points interactively, like we do in our code, it is convenient *not* to assume that indices of all moving points precede indices of fixed points: this would require resorting the points each time.

The idea is to regard all points, including fixed, as variables, but modify the system so that moving points stay in their positions. That is, now the vector $\mathbf{p}$ has all points in it, no longer in any particular order, and its length is $2(N_m + N_f) = 2N$. The force vector $\mathbf{F}$ is assembled in the same way as $\mathbf{p}$.

Then we zero out the entries in the force vector corresponding to fixed points, so that there is no force acting on them; as a consequence, these do not move.

Now the matrix $J[\mathbf{F}]$ is $2N \times 2N$. We start with assembling it as if all points were moving: we use the algorithm above, but eliminate the check if the points $\mathbf{p}_i$ and $\mathbf{p}_j$ are moving or not. Finally, we replace the rows and columns of the matrix corresponding to fixed points with corresponding rows and columns of identity matrix.

One can verify that for this approach the fixed points are not modified by the Newton iteration, and the submatrix for the moving points has exactly the same entries as in the original algorithm.

Here is an illustration of this approach in the case of three points, $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2$, with $\mathbf{p}_0$ and $\mathbf{p}_2$ fixed, and two springs $(0, 1)$ and $(1, 2)$:

1. Initially, we set $\mathbf{F} = [F_{01}, -F_{01} + F_{12}, -F_{12}]$. Then, we set forces on fixed points to zero: $\mathbf{F} = [0, -F_{01} + F_{12}, 0]$.

2. To assemble $J[\mathbf{F}]$, we first assemble the matrix as if all points were moving:

$$\begin{bmatrix} -J_{01} - J_{02} & J_{01} & J_{02} \\ J_{01} & -J_{01} - J_{12} & J_{12} \\ J_{02} & J_{12} & -J_{02} - J_{12} \end{bmatrix}$$

3. Then we set the rows and columns corresponding to fixed points 0 and 2 to the rows and columns of the identity matrix of the same size as $J[\mathbf{F}]$:

$$J[\mathbf{F}] = \begin{bmatrix} I & 0 & 0 \\ 0 & -J_{01} - J_{12} & 0 \\ 0 & 0 & I \end{bmatrix}$$

The inverse of this block-diagonal matrix has the same block form, with diagonal entries inverted:

$$J[\mathbf{F}]^{-1} = \begin{bmatrix} I & 0 & 0 \\ 0 & (-J_{01} - J_{12})^{-1} & 0 \\ 0 & 0 & I \end{bmatrix}$$

When we multiply $J[\mathbf{F}]^{-1}$ by $\mathbf{F}$ for the Newton update step, we get $J[\mathbf{F}]^{-1}\mathbf{F} = [0, (-J_{01} - J_{12})^{-1}(-\mathbf{F}_{01} + \mathbf{F}_{12}), 0]$, so there is no change in positions of the fixed points, and the correct update, that we would have obtained using the first method for matrix assembly for the moving points. This requires maintaining a larger matrix, but eliminates the need to change point indices when points are fixed and unfixed.