Rod Linkage Optimization

Julian Panetta

December 6, 2018

1 Optimization Framework

Optimizing our linkage structures is an interesting challenge, not only since simulating their deformations is already a highly nonlinear optimization problem in itself, but also because we care about (at least) two states of the structure: the flat assembly state and the curved deployed state. We want the flat state to be truly flat, the deployed state to produce a desired shape, and both states to experience reasonable stresses/elastic energy. Denoting our design parameters as a vector $\mathbf{p}$ and collecting our linkage state variables into a vector $\mathbf{x}$, we can pose the design problem as minimizing the objective:

$$J(\mathbf{p}) = \frac{\gamma}{E_0} E(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}), \mathbf{p}) + \frac{(1-\gamma)}{E_0} E(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}), \mathbf{p}) + \frac{\beta}{2l_0^2} \|\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}) - \mathbf{x}_{\mathrm{tgt}}\|^2_W, \tag{1}$$

$$\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}) \coloneqq \operatorname*{argmin}_{\mathbf{x}} E(\mathbf{x}, \mathbf{p}) \qquad \text{(flat configuration state vars)},$$

$$\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}) \coloneqq \operatorname*{argmin}_{\substack{\mathbf{x} \\ \mathbf{a}\cdot\mathbf{x} = \alpha_{\mathrm{tgt}}}} E(\mathbf{x}, \mathbf{p}) \quad \text{(deployed configuration state vars)},$$

subject to the constraint:

$$c(\mathbf{p}) = \|S_z \mathbf{x}^*_{2\mathrm{D}}(\mathbf{p})\|^2 = 0. \tag{2}$$

Here $\gamma \in [0, 1]$ trades off between preferring low energy in the deployed or flat state, $\beta > 0$ controls how hard we try to fit the deployed structure's joints to user-provided target positions $\mathbf{x}_{\mathrm{tgt}}$, diagonal matrix W holds user-provided weights summing to 1 that indicate how important fitting each individual joint is (only entries corresponding to joint position state variables are nonzero), and E_0 and l_0 are normalization constants (the initial deployed structure's energy and the length of its bounding box diagonal, respectively). Finally, $\mathbf{a}$ is a vector implementing the linear functional $\mathbf{a}^T\mathbf{x} = \frac{1}{N}\sum_i \alpha_i$ (i.e., averaging the opening angles), and S_z is a matrix selecting the joints' z coordinates from the full vector of state variables (this `numJoint` $\times$ `numDoF` matrix has a single 1 in each row and zeros everywhere else).

Note that the deployed structure stores significantly more elastic energy than the flat structure, so adjusting γ may not be fully intuitive; we may want to normalize each energy term separately. Also, we could chose to apply $c(\mathbf{p}) = 0$ as a soft constraint if having a completely flat assembly configuration is not so essential.

2 Applications

This minimization framework can be used to achieve several goals (though since it's a non-convex optimization, it's probably only appropriate for fairly small design modifications):

(a) Improve an existing structure that produces a desirable shape but requires excessive forces to assemble/deploy or does not lie flat in its closed configuration. This is done by setting $\mathbf{x}_{\mathrm{tgt}}$ equal to the current deployed positions, setting W equal to the identity, and tuning γ and β.

(b) Support interactive design where the user drags the $\mathbf{x}_{\mathrm{tgt}}$ positions around and paints weights W.

(c) Fit to a known surface by periodically updating $\mathbf{x}_{\mathrm{tgt}}$ to be the projection of $\mathbf{x}^*_{3\mathrm{D}}$ onto the surface (local-global iteration). This gives another possible approach for interactive design: the user can apply smooth deformations to the target surface and rerun the surface-fitting optimization.

3 Gradients of the Objective and Constraints

To run gradient-based optimization, we need to compute derivatives of (1) and (2) efficiently:

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{p}} &= \frac{\gamma}{E_0}\left(\frac{\partial E}{\partial \mathbf{x}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p})\frac{\partial \mathbf{x}^*_{2\mathrm{D}}}{\partial \mathbf{p}} + \frac{\partial E}{\partial \mathbf{p}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p})\right)\\ &\quad + \frac{(1-\gamma)}{E_0}\left(\frac{\partial E}{\partial \mathbf{x}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p})\frac{\partial \mathbf{x}^*_{3\mathrm{D}}}{\partial \mathbf{p}} + \frac{\partial E}{\partial \mathbf{p}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p})\right)\\ &\quad + \frac{\beta}{l_0^2}\Big(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}) - \mathbf{x}_{\mathrm{tgt}}\Big)^T W\frac{\partial \mathbf{x}^*_{3\mathrm{D}}}{\partial \mathbf{p}}\\ \frac{\partial c}{\partial \mathbf{p}} &= \mathbf{x}^*_{2\mathrm{D}}(\mathbf{p})^T S_z^T S_z \frac{\partial \mathbf{x}^*_{2\mathrm{D}}}{\partial \mathbf{p}}.\end{aligned} \tag{3}$$

First, we recall the local optimality conditions that determine $\mathbf{x}^*_{2\mathrm{D}}$ and $\mathbf{x}^*_{3\mathrm{D}}$:

$$\frac{\partial E}{\partial \mathbf{x}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p}) = 0, \qquad \frac{\partial E}{\partial \mathbf{x}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p}) = \lambda \mathbf{a}^T, \qquad \mathbf{a}^T\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}) = \alpha_{\mathrm{tgt}}. \tag{4}$$

From these, we see that $\frac{\partial E}{\partial \mathbf{x}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p})\frac{\partial \mathbf{x}^*_{2\mathrm{D}}}{\partial \mathbf{p}} = 0$ and $\frac{\partial E}{\partial \mathbf{x}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p})\frac{\partial \mathbf{x}^*_{3\mathrm{D}}}{\partial \mathbf{p}} = \lambda \mathbf{a}^T\frac{\partial \mathbf{x}^*_{3\mathrm{D}}}{\partial \mathbf{p}} = 0$, (as we would expect according to the envelope theorem). So the objective's gradient simplifies to:

$$\frac{\partial J}{\partial \mathbf{p}} = \frac{\gamma}{E_0}\frac{\partial E}{\partial \mathbf{p}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p}) + \frac{(1-\gamma)}{E_0}\frac{\partial E}{\partial \mathbf{p}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p}) + \frac{\beta}{l_0^2}\Big(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}) - \mathbf{x}_{\mathrm{tgt}}\Big)^T W\frac{\partial \mathbf{x}^*_{3\mathrm{D}}}{\partial \mathbf{p}}. \tag{5}$$

However, we still must find $\frac{\partial \mathbf{x}^*_{3\mathrm{D}}}{\partial \mathbf{p}}$ and $\frac{\partial \mathbf{x}^*_{2\mathrm{D}}}{\partial \mathbf{p}}$ to evaluate the constraint gradient and the last term in the objective gradient. These are determined by differentiating the optimality conditions (4):

$$\frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{x}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p})\frac{\partial \mathbf{x}^*_{2\mathrm{D}}}{\partial \mathbf{p}} + \frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{p}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p}) = 0 \quad \Longrightarrow H_{2\mathrm{D}}\frac{\partial \mathbf{x}^*_{2\mathrm{D}}}{\partial \mathbf{p}} = -\frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{p}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p}), \tag{6}$$

$$\frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{x}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p})\frac{\partial \mathbf{x}^*_{3\mathrm{D}}}{\partial \mathbf{p}} + \frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{p}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p}) = \mathbf{a}\frac{\partial \lambda}{\partial \mathbf{p}} \quad \Longrightarrow \begin{bmatrix} H_{3\mathrm{D}} & \mathbf{a} \\ \mathbf{a}^T & 0 \end{bmatrix}\begin{bmatrix} \frac{\partial \mathbf{x}^*_{3\mathrm{D}}}{\partial \mathbf{p}} \\ \frac{\partial \lambda}{\partial \mathbf{p}} \end{bmatrix} = \begin{bmatrix} -\frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{p}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p}) \\ 0 \end{bmatrix}, \tag{7}$$

where we defined $H_{2\mathrm{D}}$ and $H_{3\mathrm{D}}$ to be the Hessian of elastic energy with respect to state variables evaluated at $\mathbf{x}^*_{2\mathrm{D}}$ and $\mathbf{x}^*_{3\mathrm{D}}$, respectively. The last row of the equations for $\frac{\partial \mathbf{x}^*_{3\mathrm{D}}}{\partial \mathbf{p}}$ comes from differentiating the constraint in (4).

In principle, to compute each gradient component, we could solve (6),(7) for the derivative of the equilibrium state variables with respect to the parameter in question, then plug these derivatives into (3) and (5). But this requires solving $2 \times$ `numParams` linear systems (albeit only two distinct system matrices). Instead, we can apply the adjoint method and solve only two systems for the adjoint state vectors $\mathbf{y}$ and $\mathbf{w}$:

$$\begin{aligned} H_{2\mathrm{D}}\mathbf{y} &= S_z^T S_z \mathbf{x}^*_{2\mathrm{D}},\\ \begin{bmatrix} H_{3\mathrm{D}} & \mathbf{a} \\ \mathbf{a}^T & 0 \end{bmatrix}\mathbf{w} &= \begin{bmatrix} W\Big(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}) - \mathbf{x}_{\mathrm{tgt}}\Big) \\ 0 \end{bmatrix}. \end{aligned}$$

After finding these adjoint state vectors, we can compute our gradients with just some inner products:

$$\begin{aligned}\frac{\partial J}{\partial \mathbf{p}} &= \frac{\gamma}{E_0}\frac{\partial E}{\partial \mathbf{p}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p}) + \frac{(1-\gamma)}{E_0}\frac{\partial E}{\partial \mathbf{p}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p}) - \frac{\beta}{l_0^2}\mathbf{w}^T\begin{bmatrix} \frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{p}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p}) \\ 0 \end{bmatrix},\\ \frac{\partial c}{\partial \mathbf{p}} &= -\mathbf{y}^T\frac{\partial^2 E}{\partial \mathbf{x}\partial \mathbf{p}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p}).\end{aligned}$$

4 Optimizing

The gradient formulas above are all we need to run a BFGS-based optimization algorithm. For faster convergence, we might want to use Hessian information from the objective/constraints. Unfortunately, computing the full Hessian with respect to $\mathbf{p}$ is expensive (it will require solving several linear systems for each parameter, and no adjoint method-like trick can be used). However, Hessian-vector products can be computed efficiently (at a cost of four backsolves per Hessian application) as detailed in the next section. So we could use a Newton CG method as implemented in `Knitro` or `SciPy`.

One concern that we have regardless of the chosen optimization algorithm is preventing the optimization from taking excessively large steps during its line search: large changes to the design parameters will slow down the equilibrium solves for $\mathbf{x}^*_{2\mathrm{D}}$ and $\mathbf{x}^*_{3\mathrm{D}}$. On the other hand, terminating the equilibrium solves early during the line search will compute an upper bound on the true energy. So a backtracking line search looking for a sufficient decrease will automatically shorten its step length if we are forced to terminate early at a bad configuration.

5 Hessian-vector Products

We now derive a formula for the objective and constraint Hessians applied to a particular step, $\delta\mathbf{p}$:

$$\begin{aligned}
\frac{\partial^2 J}{\partial\mathbf{p}\partial\mathbf{p}}\delta\mathbf{p} =& \frac{\gamma}{E_0}\left(\frac{\partial^2 E}{\partial\mathbf{p}\partial\mathbf{x}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{x}^*_{2\mathrm{D}} + \frac{\partial^2 E}{\partial\mathbf{p}\partial\mathbf{p}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{p}\right)\\
&+\frac{(1-\gamma)}{E_0}\left(\frac{\partial^2 E}{\partial\mathbf{p}\partial\mathbf{x}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{x}^*_{3\mathrm{D}} + \frac{\partial^2 E}{\partial\mathbf{p}\partial\mathbf{p}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{p}\right)\\
&-\frac{\beta}{l_0^2}\begin{bmatrix}\frac{\partial^2 E}{\partial\mathbf{p}\partial\mathbf{x}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p})\\ 0\end{bmatrix}\delta\mathbf{w} - \frac{\beta}{l_0^2}\begin{bmatrix}\frac{\partial^3 E}{\partial\mathbf{p}\partial\mathbf{x}\partial\mathbf{x}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{x}^*_{3\mathrm{D}} + \frac{\partial^3 E}{\partial\mathbf{p}\partial\mathbf{x}\partial\mathbf{p}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{p}\\ 0\end{bmatrix}\mathbf{w},\\
\frac{\partial^2 c}{\partial\mathbf{p}\partial\mathbf{p}}\delta\mathbf{p} =& -\frac{\partial^2 E}{\partial\mathbf{p}\partial\mathbf{x}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{y} - \left(\frac{\partial^3 E}{\partial\mathbf{p}\partial\mathbf{x}\partial\mathbf{x}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{x}^*_{2\mathrm{D}} + \frac{\partial^3 E}{\partial\mathbf{p}\partial\mathbf{x}\partial\mathbf{p}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{p}\right)\mathbf{y}.
\end{aligned}$$

Evaluating this expression requires solving four linear systems for the state and adjoint state perturbations:

$$\begin{aligned}
H_{2\mathrm{D}}\delta\mathbf{x}^*_{2\mathrm{D}} &= -\frac{\partial^2 E}{\partial\mathbf{x}\partial\mathbf{p}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{p},\\
\begin{bmatrix}H_{3\mathrm{D}} & \mathbf{a}\\ \mathbf{a}^T & 0\end{bmatrix}\begin{bmatrix}\delta\mathbf{x}^*_{3\mathrm{D}}\\ \delta\lambda\end{bmatrix} &= \begin{bmatrix}-\frac{\partial^2 E}{\partial\mathbf{x}\partial\mathbf{p}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{p}\\ 0\end{bmatrix},\\
H_{2\mathrm{D}}\delta\mathbf{y} &= S_z^T S_z\delta\mathbf{x}^*_{2\mathrm{D}} - \left(\frac{\partial^3 E}{\partial\mathbf{x}\partial\mathbf{x}\partial\mathbf{x}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{x}^*_{2\mathrm{D}} + \frac{\partial^3 E}{\partial\mathbf{x}\partial\mathbf{x}\partial\mathbf{p}}(\mathbf{x}^*_{2\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{p}\right)\mathbf{y},\\
\begin{bmatrix}H_{3\mathrm{D}} & \mathbf{a}\\ \mathbf{a}^T & 0\end{bmatrix}\delta\mathbf{w} &= \begin{bmatrix}W\delta\mathbf{x}^*_{3\mathrm{D}}\\ 0\end{bmatrix} - \begin{bmatrix}\left(\frac{\partial^3 E}{\partial\mathbf{x}\partial\mathbf{x}\partial\mathbf{x}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{x}^*_{3\mathrm{D}} + \frac{\partial^3 E}{\partial\mathbf{x}\partial\mathbf{x}\partial\mathbf{p}}(\mathbf{x}^*_{3\mathrm{D}}(\mathbf{p}),\mathbf{p})\delta\mathbf{p}\right)\mathbf{w}\\ 0\end{bmatrix}.
\end{aligned}$$

The catch is that we need to compute the elastic energy Hessian's directional derivatives $\frac{\partial^3 E}{\partial\mathbf{x}\partial\mathbf{x}\partial\mathbf{p}}\delta\mathbf{p}$, $\frac{\partial^3 E}{\partial\mathbf{p}\partial\mathbf{x}\partial\mathbf{x}}\delta\mathbf{x}$ and $\frac{\partial^3 E}{\partial\mathbf{x}\partial\mathbf{x}\partial\mathbf{x}}\delta\mathbf{x}$. The first two are not so difficult since the Hessian's dependence on design parameters (rest lengths, elastic moduli) is relatively simple. However, computing $\frac{\partial^3 E}{\partial\mathbf{x}\partial\mathbf{x}\partial\mathbf{x}}\delta\mathbf{x}$ requires computing third derivatives of the elastic energy with respect to the elastic rod state variables, which is more involved. Thankfully, since we only need directional derivatives of the Hessian, we can calculate them efficiently with forward mode automatic differentiation (at roughly the cost of two additional Hessian evaluations).

6 Remarks

This formulation assumes no actuation forces are applied in the flat state and that a target angle is known. Because the flat state can easily self-collide/invert, we possibly will want bound constraints to keep the angles above some minimal value. Or maybe we want to apply a small actuation force tuned to keep the smallest angle positive. While these choices will complicate the solve for the flat equilibrium, at least they do not make the outer optimization proposed here any more difficult.

We could replace the elastic energy terms in (1) with some other function of the deformed configuration (e.g., an L^p norm of bending stress to prevent bending from concentrating at a few points). This modification would mean the derivative of these terms is now sensitive to $\frac{\partial \mathbf{x}^*_{\text{2D}}}{\partial \mathbf{p}}$ and $\frac{\partial \mathbf{x}^*_{\text{3D}}}{\partial \mathbf{p}}$, but the increase in derivative evaluation time is negligible since we already needed to solve adjoint problems for the flatness constraint and fitting terms.

We could also introduce a term explicitly trying to make the structure easier to actuate (some function of the actuation force λ that is holding the deployed structure open). Note that (7) computes the derivative of this actuation force, but our objective currently discards this information. However, if our design variables include the profile geometry, this will mostly just try to make the profile less stiff.

Finally, it is not conceptually more difficult to improve the performance of the structure at multiple instants along the deployment path (not just the beginning and end). For instance, if we wish to minimize the energy at multiple states $\mathbf{x}_i(\mathbf{p})$ along the path, we simply add terms $\frac{\partial E}{\partial \mathbf{p}}(\mathbf{x}_i(\mathbf{p}), \mathbf{p})$ for each $\mathbf{x}_i$ to the objective gradient (recall the envelope theorem). However, this does involve solving/updating these intermediate equilibria at each evaluation of J.