Projecting chains of constrained points

Morten Silcowitz

Feb 2026

1 Formulation

We're going to solve

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}\,(\mathbf{x}-\mathbf{p})^{\mathsf{T}}\mathbf{M}(\mathbf{x}-\mathbf{p}) \\ \text{s.t} \quad & \mathbf{z} = \mathbf{R}\mathbf{x} \quad \text{and} \quad \mathbf{z}_i^{\mathsf{T}}\mathbf{z}_i = 1 \end{aligned}$$

where

$\mathbf{x} \in \mathbb{R}^{3n}$	is the solution vector,
$\mathbf{p} \in \mathbb{R}^{\mathrm{3n}}$	is the vector of target/free fall positions,
$\mathbf{z} \in \mathbb{R}^{\mathrm{3(n-1)}}$	is the vector of difference vectors in the chain,
$\mathbf{z}_i \in \mathbb{R}^{3}$	is the ith 3-vector in $\mathbf{z}$,
$\mathbf{R} \in \mathbb{R}^{\mathrm{3(n-1)\times 3n}}$	is the per vector finite-difference matrix so $(\mathbf{R}\mathbf{x})_i = \mathbf{x}_i - \mathbf{x}_{i+1}$
$\mathbf{M} \in \mathbb{R}^{\mathrm{3n\times 3n}}$	is the mass matrix, simply having the mass (or weight) of each particle in each 3x3 block diagonal

We want to be working only on the difference vectors in $\mathbf{z}$. To do that, we skip the details and jump directly to the minimizer for $\mathbf{x}$, treating $\mathbf{z}$ as a constant for now:

$$\mathbf{x} = \mathbf{p} - \mathbf{M}^{-1}\mathbf{R}^{\mathrm{T}}\mathbf{S}^{-1}(\mathbf{R}\mathbf{p} - \mathbf{z})$$

where $\mathbf{S} = \mathbf{R}\mathbf{M}^{-1}\mathbf{R}^{T}$ is a block tridiagonal matix. Inserting this into the main problem eliminates $\mathbf{z} = \mathbf{R}\mathbf{x}$ and we are left with

$$\begin{aligned} \min_{\mathbf{x}} \quad & \frac{1}{2}(\mathbf{z} - \mathbf{R}\mathbf{p})^{\mathsf{T}}\mathbf{S}^{-1}(\mathbf{z} - \mathbf{R}\mathbf{p}) \\ \text{s.t} \quad & \mathbf{z}_i^{\mathsf{T}}\mathbf{z}_i = 1 \quad i = 1 \ldots n \end{aligned}$$

In this reformulated problem we must find a $\mathbf{z}$ that consist of unit length 3-vectors that minimize the distance to $\mathbf{R}\mathbf{p}$ under the $\mathbf{S}^{-1}$ norm. It worth noting that it is trivial to project $\mathbf{z}$ to satisfy the unit length constraints, and likewise it is trivial to obtain $\mathbf{x}$ given any $\mathbf{z}$. This means that whenever we take a newton step on $\mathbf{z}$ we can trivially reign it in to its feasible solution, and in effect only changing the directions of each $\mathbf{z}_i$ vector, never their length.

2 Algorithm

Here we'll outline the algorithm in full before explaining each step in detail:

```
$\mathbf{L}$ = cholesky($\mathbf{S}$)
$\mathbf{z}$ = $\mathbf{R}\mathbf{x}$
while true do
    $\mathbf{Q}$,$\mathbf{z}$ = normalize($\mathbf{z}$)
    $\mathbf{y}$ = $\mathbf{R}\mathbf{p} - \mathbf{z}$
```

```
$\lambda$ = solve($\mathbf{QSQ^T}$, $\mathbf{Qy}$)
$\mathrm{b}_{\mathbf{z}}$ = solve($\mathbf{L}$, $\mathbf{y}$)
$\mathrm{b}_{\lambda}$ = $\mathbf{L^T Q^T}\lambda$
if $\|\mathbf{bz} - \mathbf{bl}\|^2 < \epsilon$ then
    $\mathbf{s}$ = solve($\mathbf{L^T}$, $\mathrm{b}_{\mathbf{z}}$)
    $\mathbf{x} = \mathbf{p} - \mathbf{M}^{-1}\mathbf{R^T s}$
    return $\mathbf{x}$
$\mathbf{D}$ = diagonal(max(0,$\lambda$))
$\mathbf{\Delta z}$ = $\mathbf{L}$ solve($\mathbf{I} + \mathbf{L^T D L}$, $\mathrm{b}_{\mathbf{z}} - \mathrm{b}_{\lambda}$)
$\mathbf{z} = \mathbf{z} + \mathbf{\Delta z}$
```

Each iteration the algorithm does these main steps:

1. Normalize the 3-vectors in $\mathbf{z}$
2. Compute the $n-1$ Lagrange multipliers of the system at $\mathbf{z}$
3. Check the solution residual, if below threshold compute and return $\mathbf{x}$
4. Compute the Newton step on $\mathbf{z}$ and go to step 1

Note the following:

- we compute the cholesky decomposition $\mathbf{LL^T} = \mathbf{S}$ to keep the systems we need to solve sparse and symmetric. See more details bellow. Computing cholesky for a block-tridiagonal SPD matrix is simple and fast, and if $\mathbf{M}$ is fixed it can be done off-line/once.
- every `solve()` is at least a block tri-diagonal system (or simpler), which are fast and easy to solve
- to compute the Lagrange multipliers we build the Jacobian matrix $\mathbf{Q} \in \mathbb{R}^{3(n-1)\times(n-1)}$, which is simply the $\mathbf{z}_i$ vectors vertically stacked

- the matrix $\mathbf{D}$ is the contribution to the Hessian stemming from the non-linear unit length constraints. We clamp it to be positive to keep the Hessian PD. This is a hack but seems to play out well in practice

2.1 Lagrange multipliers

2.2 Hessian system