Projecting chains of constrained points

Morten Silcowitz

Feb 2026

1 Formulation

We're going to solve

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

where

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

We want to be working only on the difference vectors in $\mathbf{z}$. To do that, we need to eliminate $\mathbf{x}$, and we do this by holding $\mathbf{z}$ constant and deriving the optimality conditions for (1). We first state the Lagrangian:

$$\mathcal{L}(\mathbf{x},\lambda)=\frac{1}{2}(\mathbf{x}-\mathbf{p})^{\mathsf{T}}\mathbf{M}(\mathbf{x}-\mathbf{p})+\lambda^{\mathsf{T}}(\mathbf{z}-\mathbf{R}\mathbf{x})$$

with the optimality conditions:

$$\begin{aligned} \mathbf{M}(\mathbf{x}-\mathbf{p})-\mathbf{R}^{\mathsf{T}}\lambda &= 0 \\ \mathbf{z}-\mathbf{R}\mathbf{x} &= 0 \end{aligned}$$

solving for λ gives us:

$$\lambda = (\underbrace{\mathbf{R}\mathbf{M}^{-1}\mathbf{R}^\mathsf{T}}_{S})^{-1}(\mathbf{z} - \mathbf{R}\mathbf{p})$$

where $\mathbf{S} = \mathbf{R}\mathbf{M}^{-1}\mathbf{R}^\mathsf{T} \in \mathbb{R}^{3m \times 3m}$ is a block tridiagonal SPD matix. We use this to give us an expression for $\mathbf{x}$:

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

Inserting this into (1) eliminates $\mathbf{z} = \mathbf{R}\mathbf{x}$ and we are left with

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

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. Note that it is trivial to project $\mathbf{z}$ to satisfy the unit length constraints (under the regular $\mathbf{I}$-norm, known as normalization), 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 change the directions of each $\mathbf{z}_i$ vector, never their lengths.

2 Solution

We want to solve this system using a newton-like method, so to make sense of that we first derive the full newton step. The Lagrangian of (2) is

$$\mathcal{L}_{(\mathbf{z},\lambda)} = \frac{1}{2}(\mathbf{z} - \mathbf{R}\mathbf{p})^\mathsf{T}\mathbf{S}^{-1}(\mathbf{z} - \mathbf{R}\mathbf{p}) + \frac{1}{2}\sum_{i=1}^{m}\lambda_i(\mathbf{z}_i^\mathsf{T}\mathbf{z}_i - 1) \tag{3}$$

The optimality conditions for (3) are:

$$\begin{aligned}\mathbf{S}^{-1}(\mathbf{z}-\mathbf{Rp})+\mathbf{Q_z}^{\mathsf{T}}\lambda &= 0\\ \mathbf{z}_i^{\mathsf{T}}\mathbf{z}_i-1 &= 0 \quad i=1,\dots,m\end{aligned}$$

and the newton step to find a stationary point $(\mathbf{z},\lambda)$ for (3) is

$$\underbrace{\begin{bmatrix}\mathbf{S}^{-1}+\mathbf{D}_\lambda & \mathbf{Q}_z^{\mathsf{T}}\\ \mathbf{Q}_z & \end{bmatrix}}_{H\mathcal{L}_{(\mathbf{z},\lambda)}}\begin{bmatrix}\Delta\mathbf{z}\\ \Delta\lambda\end{bmatrix}=\underbrace{\begin{bmatrix}\mathbf{S}^{-1}(\mathbf{z}-\mathbf{Rp})+\mathbf{Q}_z^{\mathsf{T}}\lambda\\ \mathbf{Q_z z}-\mathbf{1}\end{bmatrix}}_{\nabla\mathcal{L}_{(\mathbf{z},\lambda)}} \tag{4}$$

where

$$\mathbf{Q}_z=\begin{bmatrix}\mathbf{z}_1^{\mathsf{T}} & 0 & \cdots & 0\\ 0 & \mathbf{z}_2^{\mathsf{T}} & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \mathbf{z}_m^{\mathsf{T}}\end{bmatrix}\in\mathbb{R}^{m\times 3m}\ \mathbf{D}_\lambda=\begin{bmatrix}\mathbf{I}_{\lambda_1} & 0 & \cdots & 0\\ 0 & \mathbf{I}_{\lambda_2} & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \mathbf{I}_{\lambda_m}\end{bmatrix}\in\mathbb{R}^{3m\times 3m}$$

There are many interesting ways to go about solving this system, but in our case the goal is the exploint the tri-diagonal structure of $\mathbf{S}$, in order to use fast direct solves. Instead of solving the full step, we first solve just for $\boldsymbol{\Delta}\lambda$ at $\lambda=\mathbf{0}$, effectively resetting the multipliers in each iteration. After that we solve for $\boldsymbol{\Delta}z$ to update the solution variables.

2.1 Solving for λ

To derive the solution for $\Delta\lambda$ we eliminate Δz by using $\lambda=\mathbf{0}$ and solving the first row in (4). Interestingly, because we will project the vectors in $\mathbf{z}$ in each iteration the $\mathbf{Q_z z}-\mathbf{1}$ term is always 0, which will simplify things a bit:

$$\Delta\mathbf{z}=\mathbf{S}(\mathbf{Q}_z^{\mathsf{T}}\Delta\lambda+\mathbf{S}^{-1}(\mathbf{z}-\mathbf{Rp}))$$

Substitute $\Delta\mathbf{z}$ into the second row in (4):

$$\mathbf{Q}_z(\mathbf{S}(\mathbf{Q}_z^\mathsf{T}\Delta\lambda + \mathbf{S}^{-1}(\mathbf{z} - \mathbf{Rp}))) = \underbrace{\mathbf{Q_z z - 1}}_{\mathbf{0}}$$

Simplify to solve for $\Delta\lambda$ we get:

$$\mathbf{Q}_z\mathbf{S}\mathbf{Q}_z^\mathsf{T}\Delta\lambda = -\mathbf{Q}_z(\mathbf{z} - \mathbf{Rp})$$

This system is tridiagonal because the operator $\mathbf{Q_z}(\cdot)\mathbf{Q_z^\mathsf{T}}$ collapses each 3×3 block in $\mathbf{S}$ to a scalar value $\mathbf{z_i^T S_{(i,j)} z_j}$. This operation effectively reduces the block tridiagonal structure of $\mathbf{S}$ to the plain tridiagonal structure in $\mathbf{Q_z S Q_z^\mathsf{T}} \in \mathbb{R}^{m\times m}$. Also note that $\mathbf{Q}_z z = 1$ appears in the RHS and so it could be further reduced to $\mathbf{Q_z Rp - 1}$ but since the vector $\mathbf{z - Rp}$ is needed elsewhere this has neglectable practical benefit.

2.2 Solving for $\Delta\mathbf{z}$

To solve for $\Delta\mathbf{z}$, we substitute $\Delta\lambda = 0$ into the first row of (4):

$$(\mathbf{S^{-1}} + \mathbf{D}_\lambda)\mathbf{\Delta z} = \mathbf{S^{-1}}(\mathbf{z - Rp}) - \mathbf{Q_z^\mathsf{T}}\lambda \tag{5}$$

To avoid having to solve the dense system $\mathbf{S^{-1}} + \mathbf{D}_\lambda$ we exploit that we can cheaply compute the cholesky decomposition $\mathbf{LL^\mathsf{T}} = \mathbf{S}$ and make use of the identity

$$\mathbf{S^{-1}} + \mathbf{D}_\lambda = \mathbf{L^{-T}}(\mathbf{I} + \mathbf{L^T D}_\lambda \mathbf{L})\mathbf{L^{-1}} \tag{6}$$

we insert (6) into (5)

$$\mathbf{L^{-T}}(\mathbf{I} + \mathbf{L^T D}_\lambda \mathbf{L})\mathbf{L^{-1}}\mathbf{\Delta z} = \mathbf{L^{-T}L^{-1}}(\mathbf{z - Rp}) - \mathbf{Q_z^\mathsf{T}}\lambda$$

then multiplying with the invertible $\mathbf{L}^\mathsf{T}$

$$(\mathbf{I} + \mathbf{L}^\mathbf{T}\mathbf{D}_\lambda\mathbf{L})\mathbf{L}^{-1}\boldsymbol{\Delta}\mathbf{z} = \mathbf{L}^{-1}(\mathbf{z} - \mathbf{R}\mathbf{p}) - \mathbf{L}^\mathbf{T}\mathbf{Q}_\mathbf{z}^\mathsf{T}\lambda$$

means we can solve for $\mathbf{u} = \mathbf{L}^{-\mathsf{T}}\Delta z$ and then obtain $\boldsymbol{\Delta} z$ like so

$$(\mathbf{I} + \mathbf{L}^\mathbf{T}\mathbf{D}_\lambda\mathbf{L})\mathbf{u} = \mathbf{L}^{-1}(\mathbf{z} - \mathbf{R}\mathbf{p}) - \mathbf{L}^\mathbf{T}\mathbf{Q}_\mathbf{z}^\mathsf{T}\lambda$$

and then finally

$$\Delta\mathbf{z} = \mathbf{L}^\mathsf{T}\mathbf{u}$$

The great advantage here is that $(\mathbf{I} + \mathbf{L}^\mathbf{T}\mathbf{D}_\lambda\mathbf{L})$ remains block tridiagonal and so it's fast to solve. The disadvantage is that we need one extra solve of the block bidiagonal $\mathbf{L}$ to compute $\mathbf{L}^{-1}(\mathbf{z} - \mathbf{R}\mathbf{p})$. This is annoying but still greatly outweighs the pain of solving a dense system using iterative methods. Note also that in the trivial case where $\lambda = \mathbf{0}$ the system reduces to the identity. In an effort to keep the eigenvalues of $(\mathbf{I} + \mathbf{L}^\mathbf{T}\mathbf{D}_\lambda\mathbf{L})$ positive, we clamp λ to positive values before we compute $\mathbf{D}_\lambda$. This is a hack but seems to work well in practice. We now proceed to list the full algorithm.

3 Algorithm

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

```
01  L = cholesky(S)
02  z = Rx
03  while true do
04      Q,z = normalize(z)
05      y = Rp − z
06      λ = solve(QSQ^T, Qy)
```

```
07      b_z = solve(L, y)
08      b_λ = L^T Q^T λ
09      if ||bz − bl||^2 < ε then
10          s = solve(L^T, b_z)
11          x = p − M^{-1} R^T s
12          return x
13      D = diagonal(max(0, λ))
14      Δz = L solve(I + L^T D L, b_z − b_λ)
15      z = z + Δz
```

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 $\mathbf{L}\mathbf{L}^\mathsf{T} = \mathbf{S}$ 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 $\mathbf{Q} \in \mathbb{R}^{3m \times m}$, 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