Newton's Method in Optimization

Course ID: COMPENG 3SK3
Instructor: Dr. Xiaolin Wu
Due date: February 26, 2023

1 Background

To meet safety requirements transportation authorities need to monitor the conditions of highway and railway tunnels and detect any structural changes. One can use LiDAR (light detection and ranging) to measure and model the tunnel interior surface. For this purpose, laser reflection markers are painted on the tunnel surface so that LiDAR can measure their 3D positions. The challenge is that the LiDAR readings are noisy, which affects the precision of the 3D measurements. One way to suppress the sensor noises is to scan the markers with LiDAR for $K > 1$ times, and combine the data in an optimal fashion to improve the measurement precision.

In this project, you are to apply Newton's method to perform the above optimization task. A detailed derivation of the algorithm is presented in the next section.

2 Method

2.1 Notations

- $\mathbf{p}_i = (x_i, y_i, z_i)^T$ is the ground-truth position of the i-th marker on the tunnel surface.
- $\widetilde{\mathbf{p}}_i$ is the estimate of $\mathbf{p}_i$ made by the proposed algorithm.
- $\mathbf{q}_k = (a_k, b_k, c_k)^T$ is the position of the LiDAR detector in the k-th observation.
- $\hat{\mathbf{p}}_{ik} = (\hat{x}_{ik}, \hat{y}_{ik}, \hat{z}_{ik})^T$ is the noisy measurement made in the k-th observation.
- $d_{ik} = ||\hat{\mathbf{p}}_{ik} - \mathbf{q}_k||_2 = \sqrt{(\hat{\mathbf{p}}_{ik} - \mathbf{q}_k)^T(\hat{\mathbf{p}}_{ik} - \mathbf{q}_k)}$ is the measured distance between the i-th marker and the LiDAR position in the k-th observation.
- N is the number of markers.
- K is the number of observations.

2.2 Illustation

Figure 1 illustrates how our LiDAR-based tunnel monitoring system works. There are N markers $\{\mathbf{p}_i\}_{i=1}^N$ on the tunnel surface. The LiDAR observation positions $\{\mathbf{q}_k\}_{k=1}^K$ are evenly spaced on a straight line. These positional data are stored in a $K \times 3$ matrix (given to you in '**observation_R5_L40_N100_K21.mat**'). In the k-th observation, the LiDAR makes noisy measurements of the N markers, denoted by $\{\hat{\mathbf{p}}_{ik}\}_{i=1}^N$. The assemble of these K measurements are recorded in a $K \times N \times 3$ data array (given to you in '**pts_R5_L40_N100_K21.mat**'). Besides, the distances $\{\mathbf{d}_{ik}\}_{i=1,k=1}^{N,K}$ between the i-th marker and the k-th LiDAR observation point are stored in an $N \times K$ matrix (given to you in '**dist_R5_L40_N100_K21.mat**').

For each marker, K noisy coordinates and K distances, one for each of the K observations, are available to estimate the position of this marker. Simply averaging K noisy coordinates can help reducing the noise to some extent, but this is not optimal. In this project, Newton's method will be applied to obtain an optimal estimate of marker. In next subsection, how to solve this problem with Newton's method is discussed in detail.

2.3 Formulation

For simplicity, we drop the index of marker i from the subscript of $\mathbf{p}_i$, $\widetilde{\mathbf{p}}_i$, $\hat{\mathbf{p}}_{ik}$ and d_{ik} in the following discussions.

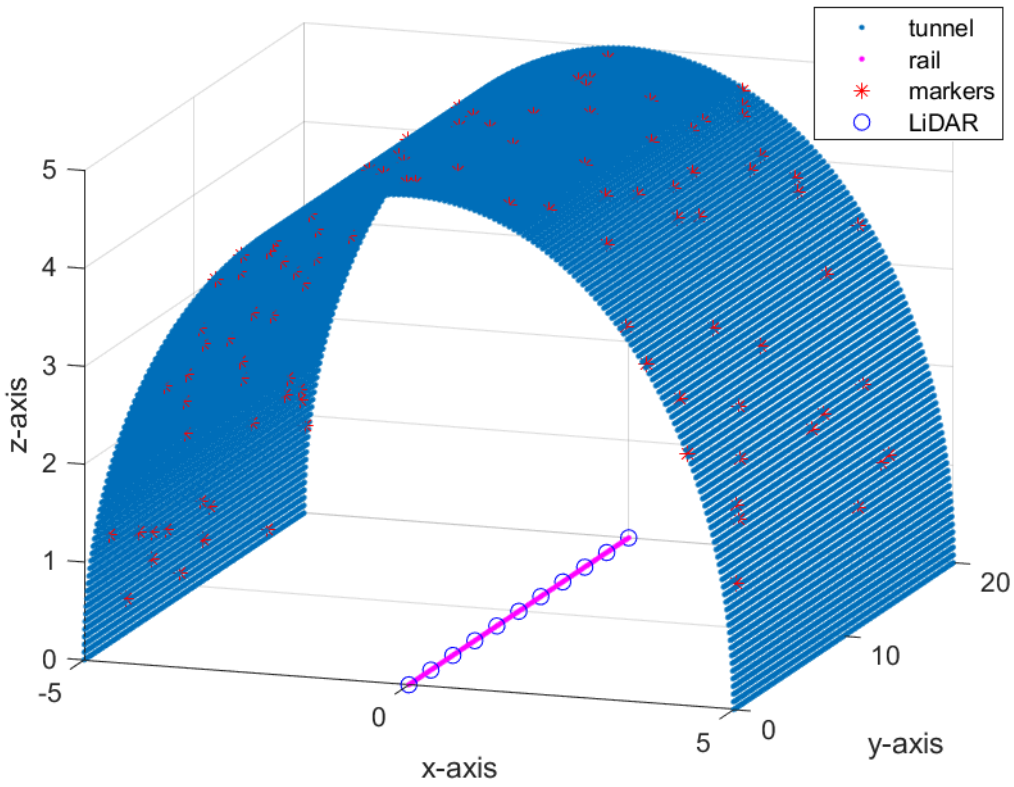

Figure 1: A demonstration for the system prototype.

Given the measured distances $\{d_k\}_{k=1}^K$ between $\{\mathbf{q}_k\}_{k=1}^K$ and the marker $\mathbf{p}$ and the noisy coordinates $\{\hat{\mathbf{p}}_k\}_{k=1}^K$, the error can be represented as

$$E(\mathbf{p}) = \frac{1}{2}\sum_{k=1}^{K}(||\mathbf{p}-\mathbf{q}_k||_2 - d_k)^2 + \lambda\sum_{k=1}^{K}||\mathbf{p}-\hat{\mathbf{p}}_k||_2^2 \tag{1}$$

where λ is a constant to control the importance of the two error terms.

The optimal coordinate of the marker is

$$\widetilde{\mathbf{p}} = \arg\min_{\mathbf{p}} \frac{1}{2}\sum_{k=1}^{K}(||\mathbf{p}-\mathbf{q}_k||_2 - d_k)^2 + \lambda\sum_{k=1}^{K}||\mathbf{p}-\hat{\mathbf{p}}_k||_2^2. \tag{2}$$

Let $\mathbf{r} \in \mathbb{R}^K$ in Eq.(3) denote the vector of residuals.

$$\mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_K \end{bmatrix} = \begin{bmatrix} ||\mathbf{p}-\mathbf{q}_1||_2 - d_1 \\ ||\mathbf{p}-\mathbf{q}_2||_2 - d_2 \\ \vdots \\ ||\mathbf{p}-\mathbf{q}_K||_2 - d_K \end{bmatrix} = \begin{bmatrix} \sqrt{(\mathbf{p}-\mathbf{q}_1)^T(\mathbf{p}-\mathbf{q}_1)} - d_1 \\ \sqrt{(\mathbf{p}-\mathbf{q}_2)^T(\mathbf{p}-\mathbf{q}_2)} - d_2 \\ \vdots \\ \sqrt{(\mathbf{p}-\mathbf{q}_K)^T(\mathbf{p}-\mathbf{q}_K)} - d_K \end{bmatrix}. \tag{3}$$

Then, Eq.(1) can be rewritten into

$$E(\mathbf{p}) = \frac{1}{2}\sum_{k=1}^{K} r_k^2 + \lambda\sum_{k=1}^{K}||\mathbf{p}-\hat{\mathbf{p}}_k||_2^2. \tag{4}$$

The Jacobian matrix of $\mathbf{r}$ with respect to $\mathbf{p}$ can be represented as

$$\mathbf{J} = \begin{bmatrix} \frac{\partial r_1}{\partial x} & \frac{\partial r_1}{\partial y} & \frac{\partial r_1}{\partial z} \\ \frac{\partial r_2}{\partial x} & \frac{\partial r_2}{\partial y} & \frac{\partial r_2}{\partial z} \\ \vdots & \vdots & \vdots \\ \frac{\partial r_K}{\partial x} & \frac{\partial r_K}{\partial y} & \frac{\partial r_K}{\partial z} \end{bmatrix} \tag{5}$$

where the k-th row of $\mathbf{J}$ is

$$\begin{bmatrix} \frac{\partial r_k}{\partial x} & \frac{\partial r_k}{\partial y} & \frac{\partial r_k}{\partial z} \end{bmatrix} = \frac{(\mathbf{p}-\mathbf{q}_k)^T}{\sqrt{(\mathbf{p}-\mathbf{q}_k)^T(\mathbf{p}-\mathbf{q}_k)}} = \begin{bmatrix} \frac{x-a_k}{r_k+d_k} & \frac{y-b_k}{r_k+d_k} & \frac{z-c_k}{r_K+d_k} \end{bmatrix}. \tag{6}$$

As for $||\mathbf{p}-\hat{\mathbf{p}}_k||_2^2$, its gradient with respect to $\mathbf{p}$ can be obtained by the rule [1]

$$\frac{\partial ||\mathbf{x}||_2^2}{\partial \mathbf{x}} = 2\mathbf{x}.$$

Next, according to the chain rule in calculus, the gradient of the loss function Eq.(4) with respect to $\mathbf{p}$ is

$$\mathbf{g} = \sum_{k=1}^{K} r_k \begin{bmatrix} \frac{\partial r_k}{\partial x} \\ \frac{\partial r_k}{\partial y} \\ \frac{\partial r_k}{\partial z} \end{bmatrix} + \lambda \sum_{k=1}^{K} 2(\mathbf{p}-\hat{\mathbf{p}}_k). \tag{7}$$

The first item in Eq.(7) is equivalent to

$$\begin{bmatrix} \frac{\partial r_1}{\partial x} & \frac{\partial r_2}{\partial x} & \cdots & \frac{\partial r_K}{\partial x} \\ \frac{\partial r_1}{\partial y} & \frac{\partial r_2}{\partial y} & \cdots & \frac{\partial r_K}{\partial y} \\ \frac{\partial r_1}{\partial z} & \frac{\partial r_2}{\partial z} & \cdots & \frac{\partial r_K}{\partial z} \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_K \end{bmatrix} = \mathbf{J}^T\mathbf{r} \tag{8}$$

Then, Eq.(7) can be rewritten into a more concise form

$$\mathbf{g} = \mathbf{J}^T\mathbf{r} + \lambda \sum_{k=1}^{K} 2(\mathbf{p}-\hat{\mathbf{p}}_k). \tag{9}$$

Regarding the Hessian matrix of loss function Eq.(4), $\mathbf{J}^T\mathbf{J}$ can approximate the Hessian matrix of $\frac{1}{2}\sum_{k=1}^{K} r_k^2$ [2] and the Hessian of $||\mathbf{p}-\hat{\mathbf{p}}_k||_2^2$ is 3×3 identity matrix $\mathbf{I}_{3\times 3}$. Then, the Hessian of loss function Eq.(4) is

$$\mathbf{H} \approx \mathbf{J}^T\mathbf{J} + 2\lambda K\mathbf{I}_{3\times 3} \tag{10}$$

With $\mathbf{g}$ and $\mathbf{H}$, we can obtain the descent direction for Newton's method

$$\Delta\mathbf{p} = -\mathbf{H}^{-1}\mathbf{g}. \tag{11}$$

In each iteration, the coordinate will be updated according to

$$\mathbf{p}^{(j+1)} = \mathbf{p}^{(j)} - \alpha\mathbf{H}^{-1}\mathbf{g} \tag{12}$$

where α is used to control the step size along the descent direction. α can be either calculated by line search algorithm [3] or set to a constant. Line search is beyond the scope of this course, so simply set α to a constant (e.g., $\alpha = 0.1$).

3 Work to do

Your task is to implement Newton's method by writing your own program according to the above formulation. This Newton's method based denoising algorithm is point-wise, i.e., it takes the corresponding measurements of one marker as input and predicts the optimal coordinate of this marker. So you should process each marker one at a time using the same algorithm[1], and put these predictions together into an $N\times 3$ matrix.

Requirements:

- Implement the above-mentioned Newton's method based denoising algorithm with Matlab. **Don't** use any built-in functions such as *fminunc* and *cvx*. **Please use only transpose, norm, addition, subtraction, multiplication, and division (including inv).** Python is also acceptable. If you choose Python, please use basic operators of numpy to write your own programs and **do not** use any advanced library functions such as *scipy.optimize* and *pytorch*.
- Submit your code with detailed documentation.

[1] Use the same λ, initialization scheme $\mathrm{p}^{(0)}$ and termination condition to process each marker.

- Use pseudo code to describe all necessary details of your implementation.
- Evaluate the quality of estimated coordinates of markers by computing the root mean square error (RMSE) [4] between your result $\{\widetilde{\mathbf{p}}_i\}_{i=1}^N$ and the ground-truth markers $\{\mathbf{p}_i\}_{i=1}^N$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}||\widetilde{\mathbf{p}}_i - \mathbf{p}_i||_2^2}. \tag{13}$$

- Fine-tune λ
  - Find λ_{op} that can make the converged RMSE as small as possible[2];
  - Plot the curve of converged RMSE against λ. The range of λ is $[\frac{1}{100}\lambda_{op}, 100\lambda_{op}]$[3].
- Fine-tune the initialization $\mathbf{p}^{(0)}$
  - Let $\lambda = \lambda_{op}$ and keep λ fixed, sketch three curves of RMSE against iterations given the following initialization schemes:
    - average the K measured coordinates;
    - choose one of the measured coordinates;
    - set it as a random vector.
  - Comment on the needed number of iterations to converge for the above three cases.
- Individual work.

Comments:

- The ground-truth coordinates of markers are organized as a $N \times 3$ matrix in '**gt_R5_L40_N100_K21.mat**'. The given ground-truth markers can only be used to compute the RMSE of your result $\{\widetilde{\mathbf{p}}_i\}_{i=1}^N$ for the purpose of self-evaluation.
- Your code will be verified on another dataset which is unknown to students.
- A template for your implementation is shown as follows[4]

Algorithm 1: Newton's method based denoising algorithm

input : $\{\hat{\mathbf{p}}_{ik}\}_{i=1,k=1}^{N,K}$, $\{d_{ik}\}_{i=1,k=1}^{N,K}$, $\{\mathbf{q}_k\}_{k=1}^{K}$
output: $\{\widetilde{\mathbf{p}}_i\}_{i=1}^{N}$

```
for i ← 1 to N do
    Extract measured coordinates and distances corresponding to the i-th marker;
    Initialization;
    while not converged do
        update the coordinate of the i-th marker according to the Newton's method;
    Record p̃_i;
```

References

[1] Kaare Brandt Petersen, Michael Syskind Pedersen, et al. The matrix cookbook. *Technical University of Denmark*, 7(15):510, 2008.

[2] Wikipedia contributors. Gauss–newton algorithm — Wikipedia, the free encyclopedia, 2023. [Online; accessed 2-February-2023].

[2] Use the average of K measured coordinates (i.e., $\frac{1}{K}\sum_{k=1}^{K}\hat{\mathrm{p}}_k$) as the initialization $\mathrm{p}^{(0)}$. Make sure that the converged RMSE is smaller than the RMSE obtained by simply average $\frac{1}{K}\sum_{k=1}^{K}\hat{\mathrm{p}}_k$, and the reference for RMSE reduction is 0.0068.

[3] $\lambda \neq 0$ and $0 < \lambda_{op} < 1$.

[4] The while loop can be replaced by a for loop (for it=1:max_iterations) where max_iterations is the maximum number of iterations and defined by you. If the termination condition for Newton's method is satisfied, break this for loop.

[3] Wikipedia contributors. Line search — Wikipedia, the free encyclopedia, 2022. [Online; accessed 2-February-2023].

[4] Wikipedia contributors. Root-mean-square deviation — Wikipedia, the free encyclopedia, 2023. [Online; accessed 2-February-2023].