COMPENG 3SK3 - Project 02: Newton's Method in Optimization

Professor: Dr. Xiaolin Wu

Abdulrahman Hamideh, hamideha, 400170395
Due: February 27, 2022

As a future member of the engineering profession, the student is responsible for performing the required work in an honest manner, without plagiarism and cheating. Submitting this work with my name and student number is a statement and understanding that this work is my own and adheres to the Academic Integrity Policy of McMaster University and the Code of Conduct of the Professional Engineers of Ontario. Submitted by Abdulrahman Hamideha, hamideha, 400170395.

1. Newton's Optimization Algorithm Implementation

Pseudocode:

```
% Load data matrices
% Define constants alpha and lambda

N = Number of markers on tunnel surface
K = Number of LiDAR markers on ground

% Initialize matrix to store estimated marker positions
p_opt = zeros(N, 3)

for i = 1:N
	% Noisy position of the i-th marker seen by K LiDARs
	p_hat = pts_markers(i)

	% Distance between i-th marker and K LiDARs
	d = dist(i)

	% Initialize p0 as average of the noisy coordinates. For the other two schemes
	% either take a single sample from p_hat or create a random vector
	p = mean(p_hat) or p_hat(1,:) or rand(1,3)

	% Descent direction
	delta_p = Inf(3,1)

	% Check for convergence by a given tolerance
	while iteration < maximum number of iterations
		% Vector of residuals
		r = ||p - pts_o||^2 - d
		% Jacobian
		J = (p - pts_o) / ||p - pts_o||^2 - d

		% Gradient of loss function/error function
		g = J' * r + lambda * sum(2 * (p - p_hat))
		% Hessian of loss function
		H = J' * J + 2 * lambda * K * I;
		% Descent direction for Newton's method
		delta_p = -H^-1*g
		% Updated iteration of p
		p = p + alpha * delta_p

		iter +=1
		if converged
			break;
	% Place optimized coordinates in matrix
	p_opt(i) = p
```

Algorithm implementation:

In this project, Newton's Method was used to optimize readings of N markers on the surface of a tunnel. The marker readings are performed by K LiDARs situated on the ground. However, the LiDAR sensors introduce noise. Thus, Newton's method is used to suppress the noise in the readings and optimize the scanned locations of each marker. The noisy readings performed by the K sensors of the N markers are mapped in a three-dimensional xyz space.

There are two important constants used in this project. α and λ. α is used to control the step size along the direction of descent, and in this application was set to $\alpha = 0.1$. Alternatively, α could be found using a line search algorithm. The second constant, λ, is used to control the importance of the two error terms:

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

In this relationship, r_k is the residual, p is the position of the marker and $\hat{p}_k$ is the noisy position of the marker. Ultimately, this program aims to find an optimal λ_{op} such that RMSE between the optimized marker positions and the ground truth marker positions is minimized.

The optimization algorithm involves looping through every marker to optimize its position, and applying Newton's method until a convergence criteria is met. This is shown by the while-loop nested in the for-loop from 1 to N. It is important to note that this algorithm acts point-wise, i.e. it predicts the optimal coordinates of a marker one at a time. The criteria used to determine convergence was that a maximum number of iterations of the algorithm is reached or the magnitude of the descent direction, Δp, is less than 10^{-6}. Another way to think about this is that the magnitude between consecutive estimates of p is less than 10^{-6}.

Newton's optimization method involves selecting an initialization, $p^{(0)}$, to begin optimization. In this project, three different initialization schemes were used: 1. Averaging the K noisy coordinates to get an initial starting coordinate. 2. Using a single coordinate from the K noisy coordinates. 3. Using a randomly generated coordinate (this was randomly generated as `rand(1,3)`). Newton's method is sensitive to the initialization point, thus, the results for each scheme were different and will be discussed in Section 3.

During optimization, the following values were used to find the optimal coordinates for the marker:

Firstly, the vector of residuals r was found. The vector of residuals is a $K \times 1$ vector storing the error in the noisy distance and the actual distance between the marker and the sensor. The k-th row of r is given by:

$$r_k = ||p - q_k|| - d_k$$

d_k is given and is the distance between the i-th marker and k-th sensor and q_k is the position of the k-th sensor.

Secondly, the Jacobian of r with respect to p was found, where the k-th row of J is as follows:

$$J_k = \frac{(p-q_k)^T}{||p-q_k||}$$

Thirdly, using the found J and r, the gradient, g, of the loss function as well as the Hessian matrix were found:

$$g = J^T r + \lambda \sum_{k=1}^{K} 2(p - \widehat{p_k})$$

$$H = J^T J + 2\lambda K I_{3\times 3}$$

In the Hessian function, $I_{3\times 3}$ is the 3×3 identity matrix.

Finally, Δp was updated for the purpose of checking convergence and the coordinate $p^{(i+1)}$ was also updated with the optimized values for that iteration:

$$\Delta p = - H^{-1} g$$

$$p^{(i+1)} = p^i + \alpha \Delta p$$

The logic for this algorithm was implemented in a function called `optimize` in MATLAB. This function takes five parameters: *pts_o,* the coordinates of the LiDAR sensors; *pts_markers*, the noisy coordinate measurements of the markers as seen by the sensors; *dist*, the actual distance between k-th sensor and i-th marker; *lambda*; and *max_iter*, the maximum number of iterations before terminating Newton's method, even if the method has not converged yet. The function returns a $N \times 3$ matrix of the optimized coordinates of the markers.

Code:

```
clc; clear;
load('observation_R5_L40_N100_K21.mat', 'pts_o');
load('pts_R5_L40_N100_K21.mat', 'pts_markers');
load('dist_R5_L40_N100_K21.mat', 'dist');
load('gt_R5_L40_N100_K21.mat', 'pts_marks_gt')

lambda = 0.0168318035;
% ---------------- Apply Newton Optimization -----------------
p_opt = optimize(pts_o, pts_markers, dist, lambda, 200);
rmse = calculate_rmse(p_opt, pts_marks_gt);
fprintf("Lambda = %.10f, RMSE = %.10f\n", lambda, rmse);

function p_opt = optimize(pts_o, pts_markers, dist, lambda, max_iter)
	alpha = 0.1;
	% Number of markers on tunnel surface
	N = size(pts_markers, 2);
	% Number of LiDAR markers on ground
	K = size(pts_o, 1);

	% Initialize matrix to store estimated marker positions
	p_opt = zeros(N, 3);
	for i = 1:N
		% Noisy position of the i-th marker seen by K LiDARs
		p_hat = squeeze(pts_markers(:, i, :));
		% Distance between i-th marker and K LiDARs
		d = dist(i, :)';

		% Initialize marker position as average of the noisy coordinates
		p = mean(p_hat);

		% Initialize marker as single coordinate from the noise measurements
		% p = p_hat(1,:);

		% Initialize marker position as random vector
		% p = rand(1,3);

		% Keep track of iterations of Newton's method
		iter = 0;

		% While Newton's method is not converged apply Newton's method
		while iter < max_iter
			% Vector of residuals
			r = vecnorm(p - pts_o, 2, 2) - d;
			% Jacobian
			J = (p - pts_o) ./ vecnorm(p - pts_o, 2, 2);
```

```
                    % Gradient of loss function/error function
                    g = J' * r + lambda * sum(2 * (p - p_hat))';
                    % Hessian of loss function
                    H = J' * J + 2 * lambda * K * eye(3);

                    % Descent direction for Newton's method
                    delta_p = -inv(H)*g;
                    % Updated iteration of p
                    p = p + alpha * delta_p';
                    % Place optimized coordinates in matrix
                    p_opt(i, :) = p;

                    iter = iter + 1;

                    if norm(delta_p) < 1e-6
                           break;
                    end
             end
       end
end

function rmse = calculate_rmse(p_opt, p)
       rmse = sqrt(norm(p_opt - p, 'fro').^2/numel(p));
end
```

2. **Fine-tuning** λ

In order to find the optimal λ that minimizes RMSE, the following MATLAB code was used:

```
lambda_values = logspace(-5, 1, 200);
rmse_values = zeros(size(lambda_values));
for j = 1:length(lambda_values)
    p_opt = optimize(pts_o, pts_markers, dist, lambda_values(j), 200);
    rmse = sqrt(norm(p_opt - pts_marks_gt, 'fro').^2/numel(p_opt));
    rmse_values(j) = rmse;
end
semilogx(lambda_values, rmse_values, '.-');
title('RMSE vs \lambda');
xlabel('\lambda');
ylabel('RMSE');
[~, index] = min(rmse_values);
lambda = lambda_values(index);
fprintf("Minimum lambda, lambda_op = %.10f\n", lambda);
```

The code above creates a vector of 200 logarithmically spaced values for λ. This vector is looped through, the optimization algorithm is applied for each λ, and the respective RMSE is

calculated. The value of λ that was found to minimize converged RMSE is $\lambda = 0.0168318035$. The plot of the lambda vs. corresponding RMSE is shown in Figure 1.

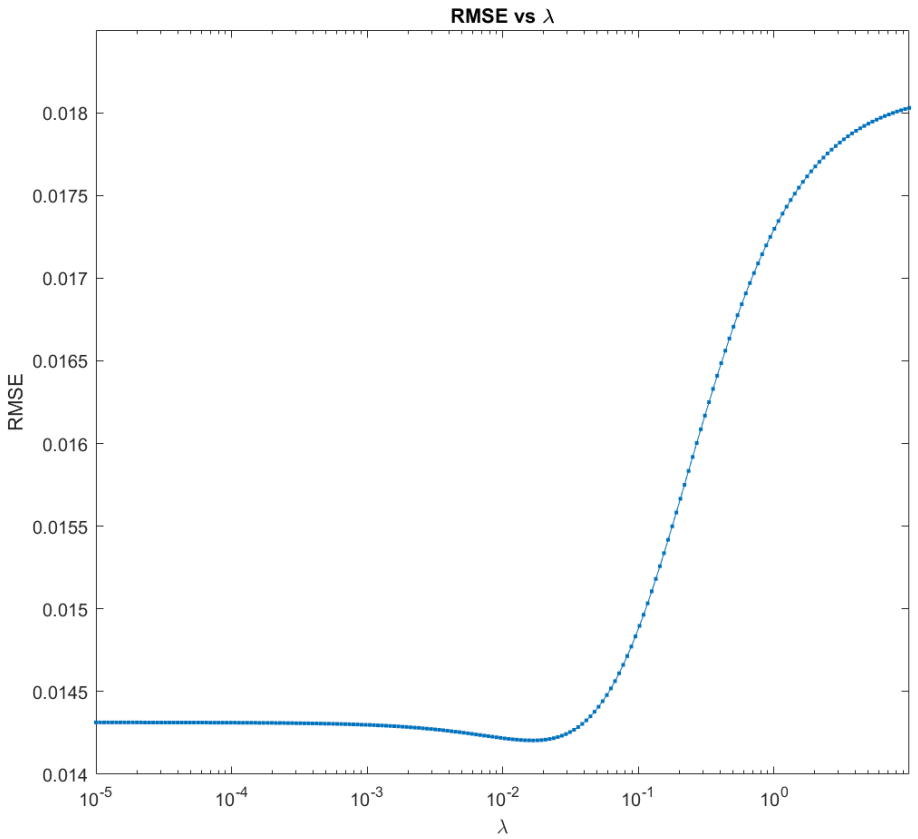

Figure 1: Plot of λ vs. RMSE

Figure 1 shows that the optimal value for λ is between 0 and 1, as expected. However, for very small values the RMSE does not change significantly and is relatively flat (between $\lambda = 10^{-5}$ and $\lambda = 10^{-3}$. Then, RMSE decreases to a minimum and as λ approaches 10^{-1} and 10^{0}, RMSE increases rapidly.

3. Fine-tuning Initialization $P^{(0)}$

Keeping λ fixed as $\lambda_{op} = 0.0168318035$, the three following initialization schemes were considered:

1. p^0 is the average K measured noisy coordinates
2. p^0 is a single measured noisy coordinate
3. p^0 is a random coordinate

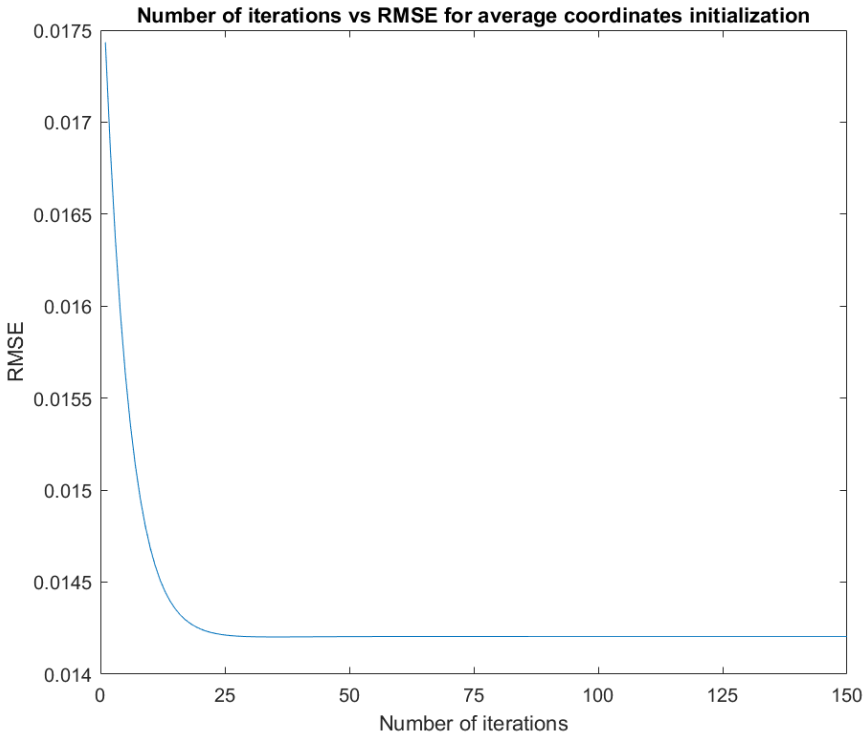

Figure 2: Number of iterations vs RMSE for P^0 as average of noisy coordinates

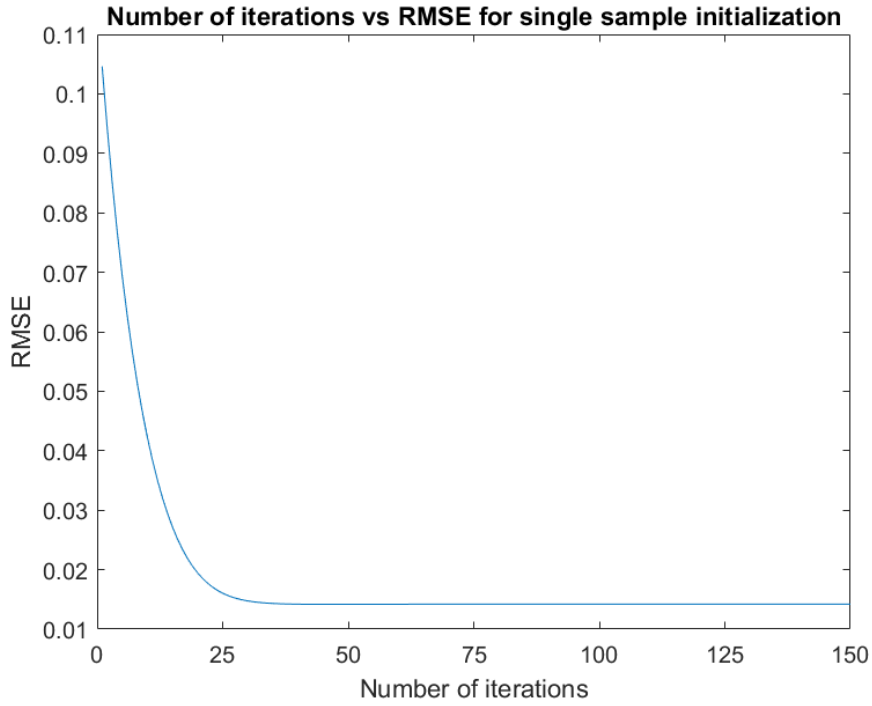

Figure 3: Number of iterations vs RMSE for P^0 as single noisy coordinate

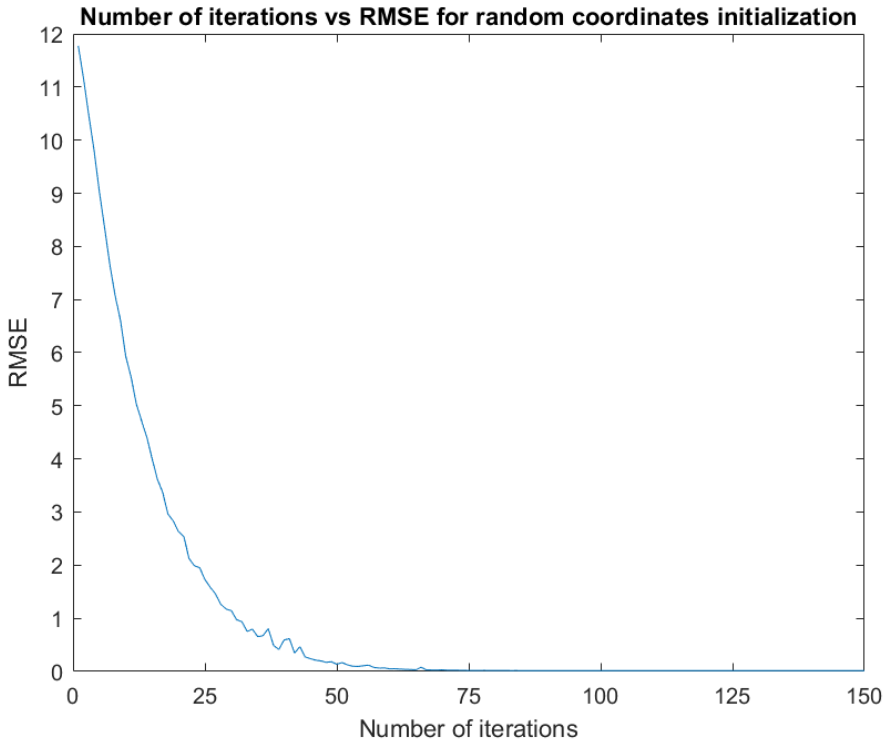

Figure 4: Number of iterations vs RMSE for P^0 as random coordinate

Figures 2-4 show the plots of RMSE vs number of iterations for different initializations for Newton's method. Figure 2 shows that when P^0 is initialized as the average of the noisy measurements, the RMSE quickly and monotonically decreases as the number of iterations increases. Additionally, the RMSE for a single iteration is fairly close to the converged RMSE. Figure 3 shows that when P^0 is initialized as a single noisy coordinate, RMSE is still monotonically decreasing, however, it requires more iterations to reach convergence. And the starting RMSE is 10 times higher. Finally, when P^0 is initialized as a random vector, RMSE converges very slowly, it is not monotonically decreasing (there are bumps in the graph), and the RMSE for a single iteration is extremely high.

The algorithm converges (the while loop terminates) in 79, 89 and 150 iterations for the 1st, 2nd and 3rd initialization schemes, respectively. This is expected as taking an average of the noisy measurements gives a better estimate of the optimal coordinate than simply taking a single sample from the noisy coordinates. Which in turn, is more effective than using a random vector without any correlation to the actual coordinates of the marker.

4. Results and Discussion

The quality of the estimated coordinates of markers found using this algorithm as well as the quality of the selected parameters (α and λ) were quantified by calculating the root-mean-square error (RMSE) between the optimized measurements and the provided ground-truth coordinates.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} ||\overline{p}_i - p_i||^2}$$

In this formula, $\overline{p}_i$ is the estimated coordinates found using the algorithm discussed.

The RMSE between the estimated coordinates and the ground truth coordinates, using $\lambda = 0.0168318035$ was $RMSE = 0.0142046174$. Note that the value of lambda used was found to minimize RMSE and is discussed in Section 3.

In comparison, the RMSE between the ground truth coordinates, and the coordinates found by simply averaging K noisy coordinates was found to be $RMSE = 0.0181290819$. As expected, using Newton's optimization to denoise the coordinates resulted in a much better RMSE.

Note that for a constant value for lambda, RMSE converges to the same value regardless of how Newton's method is initialized. Section 3 shows that for all 3 initialization schemes: averaging the noisy coordinates, picking a single coordinate, or using a randomly generated coordinate, the RMSE converges to the same value. Therefore, for $\lambda = \lambda_{op} = 0.0168318035$, the root-mean-square error converges and is minimal at $RMSE = 0.0142046174$.