COMPENG 3SK3 Project 2 Report

Raeed Hassan
hassam41

February 26, 2023

MATLAB Implementation

The documented code for the MATLAB implementation of Newton's method is implemented in `project2.m`. The function implementing Newton's method is shown below in Listing 1.

Listing 1: MATLAB Implementation of Newton's Method

```
%% Implement the Newton's method based denoising algorithm
function p_est = newtons(init_scheme, alpha, N, K, lambda,
   max_iterations)
    p_est = zeros(N,3);

    %% Load data
    % load distances, d (dist)
    load('data for student/dist_R5_L40_N100_K21.mat','dist');
    % load LiDAR observation positions, q (pts_o)
    load('data for student/observation_R5_L40_N100_K21.mat','
       pts_o');
    % load LiDAR measurements, p (pts_markers)
    load('data for student/pts_R5_L40_N100_K21.mat','
       pts_markers');

    for i = 1:N
        %% extract coordinates, distances for marker and set
           appropriate p, d, q values
        p_hat = squeeze(pts_markers(:,i,:));
        d = dist(i,:);
        q = pts_o;

        %% initalize value of p(0)
        switch init_scheme
            case 'mean'
                p0 = p0_init_mean(p_hat);
            case 'first'
                p0 = p0_init_first(p_hat);
            case 'random'
                p0 = p0_init_random;
        end

        p = p0.';

        for j = 1:max_iterations
        r = zeros(K,1);
        J = zeros(K,3);

        %% Calculate and update iteration
```

```
        for k = 1:K
            % Calculate r and J
            r(k) = norm(p-q(k,:).') - d(k);

            J(k,:) = [...
            (p(1) - q(k,1).') / (r(k) + d(k)),...
            (p(2) - q(k,2).') / (r(k) + d(k)), ...
            (p(3) - q(k,3).') / (r(k) + d(k))];

        end
        % Calculate g and H
        g = (J.' * r) + (lambda * sum(2*(p.'-p_hat)).');
        H = (J.' * J) + (2*lambda*K*eye(3));

        % Update iteration
        p = p - alpha*inv(H)*g;
        end

        p_est(i,:) = p;
    end
end
```

Pseudo Code

The pseudo code describing all the necessary details of the Newton's method implementation is shown in Algorithm 1.

Algorithm 1: Newton's method based denoising algorithm

Initialize variables;
Load data;
input : $\{\hat{\mathbf{p}}_{\mathrm{ik}}\}_{i=1,k=1}^{N,K}$, $\{d_{\mathrm{ik}}\}_{i=1,k=1}^{N,K}$, $\{\mathbf{q}_{\mathrm{k}}\}_{k=1}^{K}$
output: $\{\tilde{\mathbf{p}}_{\mathrm{i}}\}_{i=1}^{N}$
for $i \leftarrow 1$ **to** N **do**
    Extract measured coordinates and distances corresponding to the i-th marker;
    Initialize starting position $\mathrm{p}^{(0)}$;
    for $j \leftarrow 1$ **to** max_iterations **do**
        Calculate vector of residuals, $\mathbf{r}$, and the Jacobian matrix, $\mathbf{J}$;
        Calculate gradient of the loss function, $\mathbf{g}$, and Hessian matrix of loss function, $\mathbf{H}$;
        Calculate descent direction for Newton's method;
        Update the coordinate of the i-th marker according to the Newton's method;
    end
    Record $\tilde{\mathbf{p}}_{\mathrm{i}}$;
end

Root Mean Square Error (RSME)

The quality of the estimated coordinates of markers was evaluated by computing the RSME between the estimated coordinates and the ground-truth markers with the formula below:

$$\text{RSME} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|\tilde{\mathbf{p}}_i - \mathbf{p}_i\|_2^2} \tag{1}$$

The RSME was calculated in MATLAB with the function shown in Listing 2.

Listing 2: MATLAB Implementation of RSME

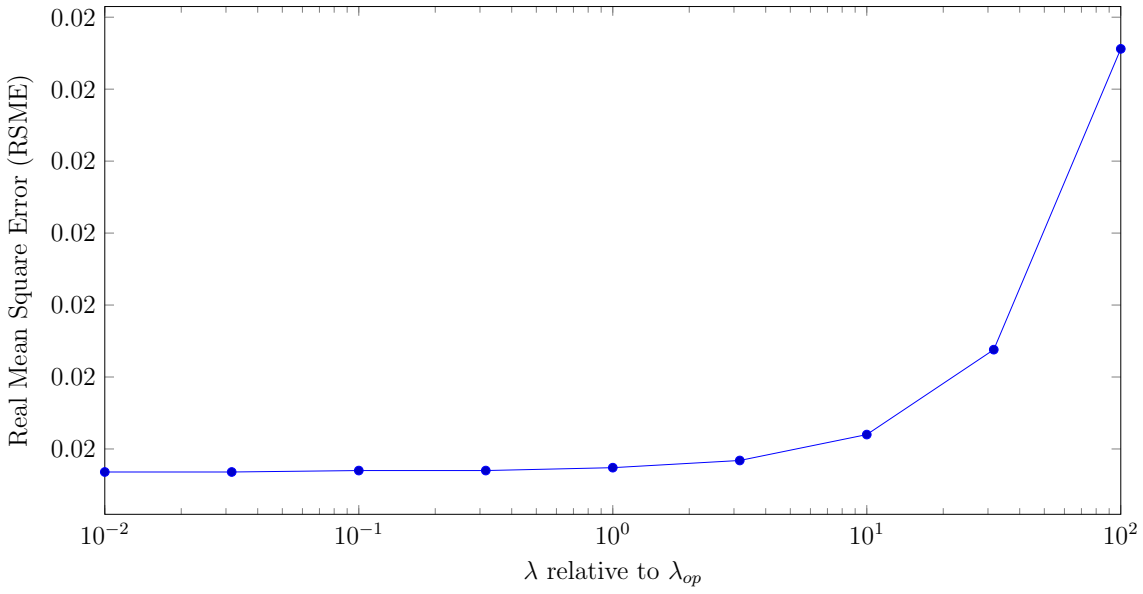
```
function val = RSME(p_est, N)
    % load ground-truth oordinates, (pts_marks_gt)
    load('data for student/gt_R5_L40_N100_K21.mat','
       pts_marks_gt');

    val = sqrt((1/N) * norm(p_est - pts_marks_gt)^2);
end
```

Fine-tune λ

To determine the optimal λ, λ_{op}, we change the value of λ when $\mathbf{p}^{(0)}$ is the average of the K measured coordinates to reduce the RMSE after 50 iterations of Newton's method. We observe that lower value of λ is, the lower the converged RSME is. We select $\lambda = 10^{-4}$ as λ_{op} as continuing to decrease λ any further results in a negligible reduction in the RSME.

Figure 1: RSME against λ



Fine-tune the initialization $\mathbf{p}^{(0)}$

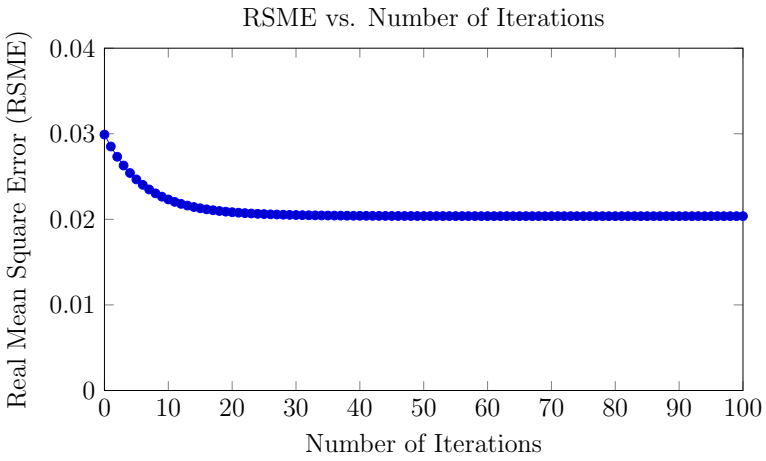
The curves of RSME against iterations given three different initialization schemes at $\lambda = \lambda_{op} = 10^{-4}$ are shown below. Figure 2a shows the RSME against iterations given $\mathbf{p}^{(0)}$ is the average of the K measured coordinates. Figure 2b shows the RSME against iterations given $\mathbf{p}^{(0)}$ is the first measured coordinate. Figure 2c shows the RSME against iterations given $\mathbf{p}^{(0)}$ is a normally distributed random vector.

The number of iterations needed varies depending on the chosen initialization scheme. When $\mathbf{p}^{(0)}$ is the average of the K measured coordinates, we observe the initial error (RSME) to be relatively close to the converged value, and quickly converges within 20 iterations. When selecting the first measured coordinate instead, we see more iterations are needed to converge and initial error is larger (nearly an order of magnitude larger), but the final converged error is similar. Similarly, when switching to a normally distributed random vector for the initialization scheme we observe the number of iterations needed for convergence and the initial error both increase significantly (again increasing by an order of magnitude from the previous scheme), while the final converged error is similar.

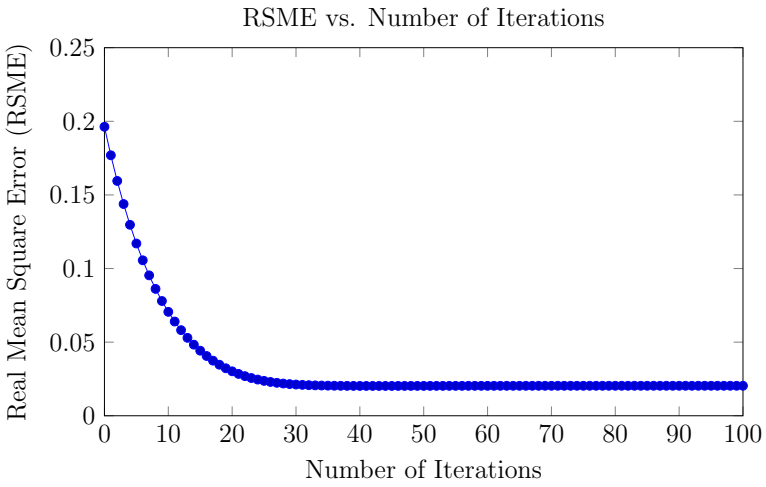
We can conclude that the converged RSME will be similar regardless of the initialization scheme used, however the number of iterations required for convergence and the initial RSME will vary depending on the scheme.

Figure 2: RSME against Iterations for Different Initialization Schemes

(a) RSME for Average of Measured Coordinates



(b) RSME for First of Measured Coordinates



(c) RSME for Normally Distributed Random Vector