Homework 3 16833: Robot Localization and Mapping

Harsh Dhruva
Linear and Nonlinear SLAM Solvers

April 8, 2021

---

1 2D Linear SLAM

1.1 Measurement Function

$$\mathbf{h_o}(\mathbf{r^t}, \mathbf{r^{t+1}}) = \mathbf{r^{t+1}} - \mathbf{r^t} = \begin{bmatrix} r_x^{t+1} - r_x^t \\ r_y^{t+1} - r_y^t \end{bmatrix} \quad \mathbf{H_o}(\mathbf{r^t}, \mathbf{r^{t+1}}) = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

$$\mathbf{h_l}(\mathbf{r^t}, \mathbf{l^k}) = \mathbf{l^k} - \mathbf{r^t} = \begin{bmatrix} l_x^k - r_x^t \\ l_y^k - r_y^t \end{bmatrix} \quad \mathbf{H_l}(\mathbf{r^t}, \mathbf{l^k}) = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

1.2 Build a linear system

1.3 Solvers

1.4 Exploit Sparsity

4.

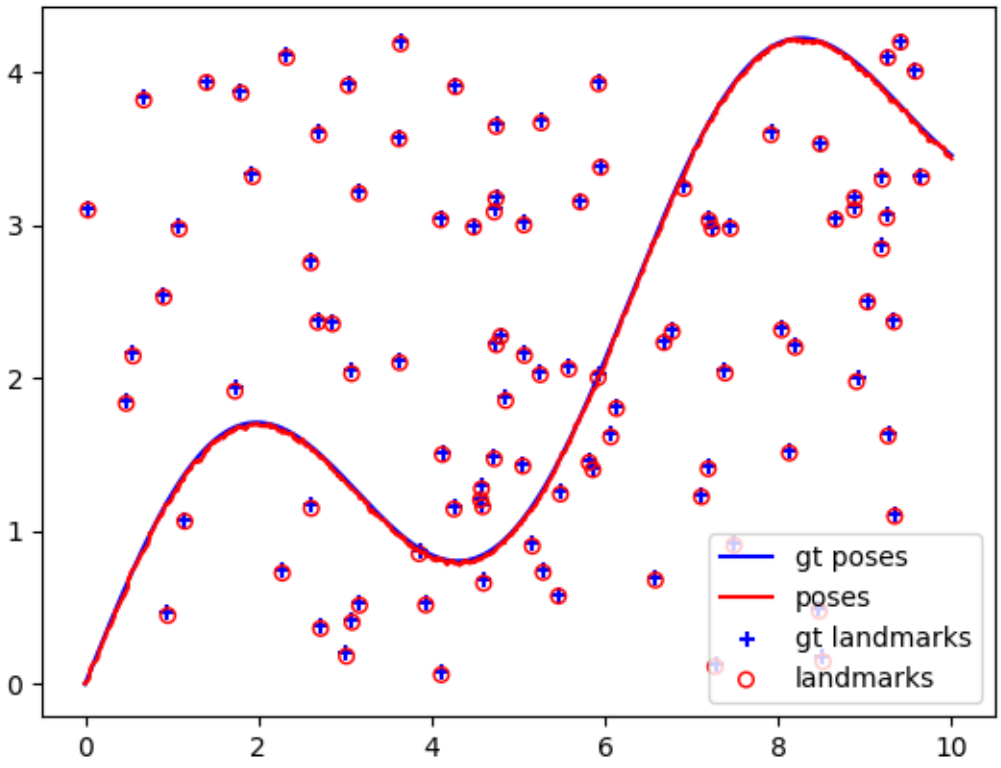
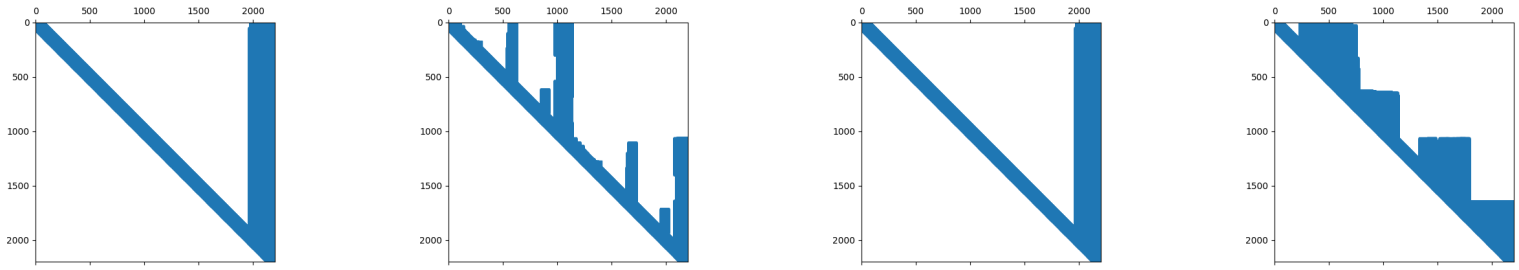


Figure 1: Trajectory and landmarks for 2d linear data

Method	Time(s)
lu	0.03643
default	0.06413
lu colamd	0.11306
qr colamd	0.69140
qr	0.71495
pinv	1.51051
lu bonus	1.51113

Table 1: Run Time using different methods from fastest to slowest

The efficiency of the methods in terms of run time are shown in Table 1, from fastest to slowest. The LU method with 'NATURAL' ordering is the fastest. In general LU is faster than QR, as LU operates on $A^T A$, which is a smaller square matrix, compared to A, the larger rectangular matrix that QR factorization operates on. Pseudo-inverse is the slowest as it does not exploit sparsity of the matrix.



(a) qr (b) qr colamd (c) lu (d) lu colamd

Figure 2: Square root information matrix for linear data

As seen in Figure 2, the square root information matrix for QR and LU with 'NATURAL' ordering are quite dense. The QR 'COLAMD' ordering results in a sparser information matrix, which is the reason the qr colamd method is faster than qr. However, the LU 'COLAMD' ordering results in a denser matrix, rendering the method to be slower than LU with 'NATURAL' ordering. The Bonus method for LU as seen in Figure 3 results in the same information matrix as the one obtained in Figure 2(d) lu colamd.

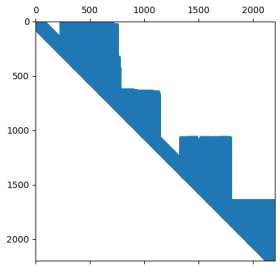


Figure 3: Square root information matrix of LU bonus for linear data

5.

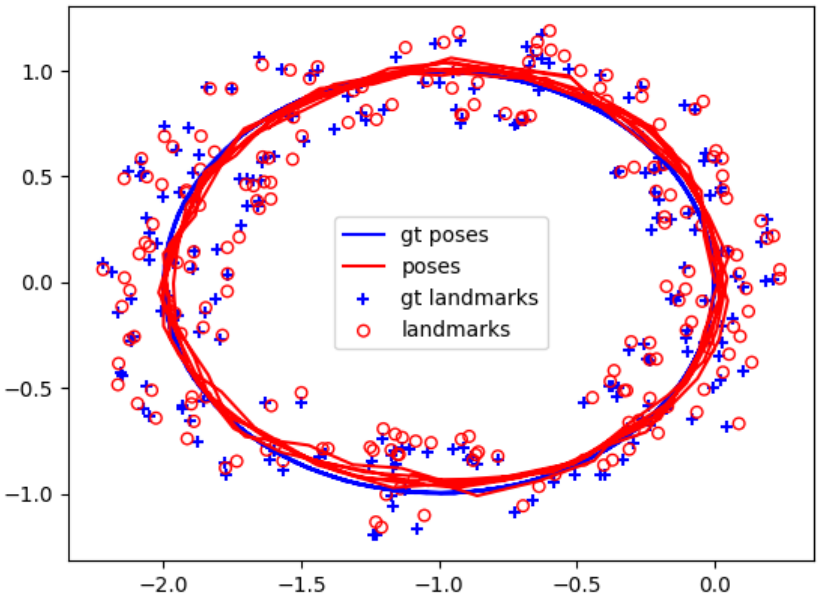
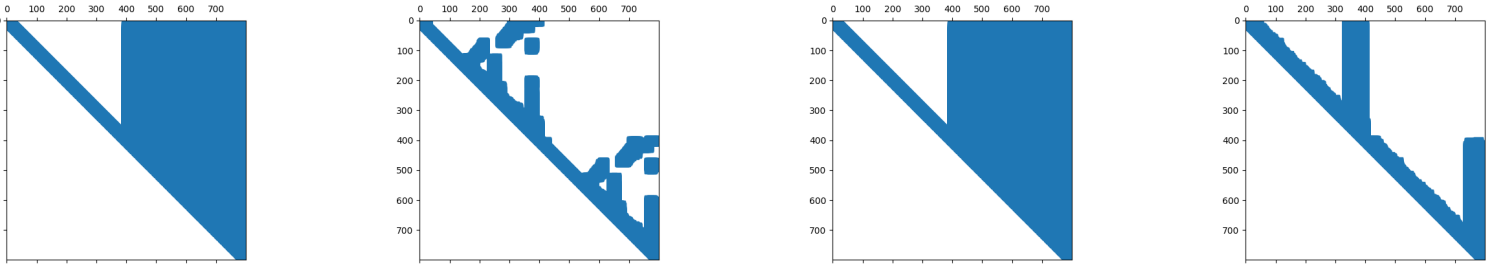


Figure 4: Trajectory and landmarks for 2d linear loop data

Method	Time(s) linear	Time(s) linear loop
lu	0.03643	0.03699
default	0.06413	0.00417
lu colamd	0.11306	0.00430
qr colamd	0.69140	0.05258
qr	0.71495	0.39594
pinv	1.51051	0.23392
lu bonus	1.51113	0.25739

Table 2: Run Time using different methods from fastest to slowest

We see from Table 2, that linear loop takes less time in general. The default method is the fastest, with lu colamd after that. In this case, the 'COLAMD' ordering results in faster speed for LU and QR. LU is still seen to be faster than QR, however the QR method is numerically more stable.



(a) qr (b) qr colamd (c) lu (d) lu colamd

Figure 5: Square root information matrix for linear loop data

We see the square root information matrices for the linear loop data in Figure 5. Here the 'NATURAL' ordering results in denser R matrices than in the linear case, and the 'COLAMD' ordering results in R matrices that are a lot sparser. This is the reason why the 'COLAMD' method results in a faster speed to compute the solution for both the LU and QR method in the linear loop case.

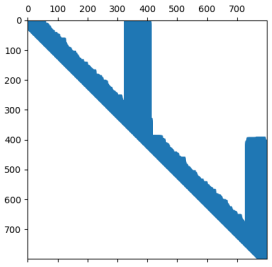


Figure 6: Square root information matrix of LU bonus for linear loop data

2 2D Nonlinear SLAM

2.1 Measurement function

2.

$$\mathbf{h_l}(\mathbf{r^t}, \mathbf{l^k}) = \mathbf{l^k} - \mathbf{r^t} = \begin{bmatrix} atan2(l_y^k - r_y^t, l_x^k - r_x^t) \\ ((l_x^k - r_x^t)^2 + (l_y^k - r_y^t)^2)^{1/2} \end{bmatrix}$$

$$\mathbf{H_l}(\mathbf{r^t}, \mathbf{l^k}) = \begin{bmatrix} \frac{l_y^k - r_y^t}{(l_x^k - r_x^t)^2 + (l_y^k - r_y^t)^2} & \frac{-(l_x^k - r_x^t)}{(l_x^k - r_x^t)^2 + (l_y^k - r_y^t)^2} & \frac{-(l_y^k - r_y^t)}{(l_x^k - r_x^t)^2 + (l_y^k - r_y^t)^2} & \frac{l_x^k - r_x^t}{(l_x^k - r_x^t)^2 + (l_y^k - r_y^t)^2} \\ \frac{-(l_x^k - r_x^t)}{\sqrt{(l_x^k - r_x^t)^2 + (l_y^k - r_y^t)^2}} & \frac{-(l_y^k - r_y^t)}{\sqrt{(l_x^k - r_x^t)^2 + (l_y^k - r_y^t)^2}} & \frac{l_x^k - r_x^t}{\sqrt{(l_x^k - r_x^t)^2 + (l_y^k - r_y^t)^2}} & \frac{l_y^k - r_y^t}{\sqrt{(l_x^k - r_x^t)^2 + (l_y^k - r_y^t)^2}} \end{bmatrix}$$

2.2 Build a linear system

2.3 Solver

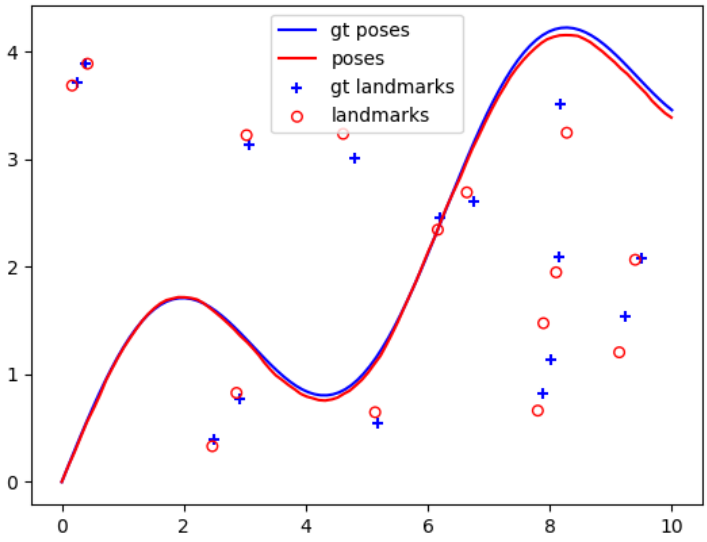


Figure 7: Trajectory and Landmarks before optimization

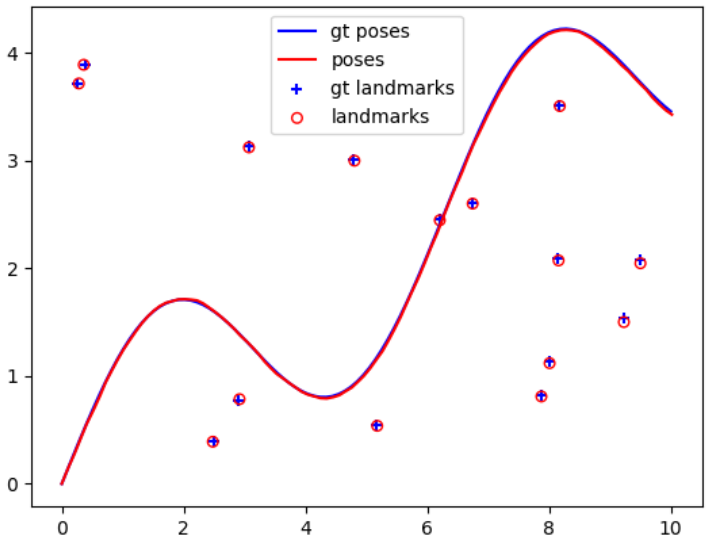


Figure 8: Trajectory and Landmarks after optimization

In the linear case, our measurement model is linear, and therefore the solution can be obtained by directly solving the normal equations in a batch fashion and exploit sparsity. However, in the non linear case a direct solution is not possible, and therefore it is solved in an iterative way, which requires an initial estimate. We linearize the measurement model with the initial estimate as the linearization point, and at an increment from the current estimate. We then solve for the increment, and add it to the current estimate to get the new estimate. We then iterate until convergence to get the final estimate.