16-833 Homework 3 Write Up

Exercise 1.1
The odometry function h_o is defined as:

$$h_o = r^{t+1} - r^t = \begin{bmatrix} r_x^{t+1} - r_x^t \\ r_y^{t+1} - r_y^t \end{bmatrix} = \begin{bmatrix} h_{o,1} \\ h_{o,2} \end{bmatrix}$$

where the states are: $X = [r_x^t, r_y^t, r_x^{t+1}, r_y^{t+1}]$. Therefore, its Jacobian matrix can be defined as:

$$H_o = \frac{\partial h_o}{\partial X} = \begin{bmatrix} \frac{\partial h_{o,1}}{\partial r_x^t} & \frac{\partial h_{o,1}}{\partial r_y^t} & \frac{\partial h_{o,1}}{\partial r_x^{t+1}} & \frac{\partial h_{o,1}}{\partial r_y^{t+1}} \\ \frac{\partial h_{o,2}}{\partial r_x^t} & \frac{\partial h_{o,2}}{\partial r_y^t} & \frac{\partial h_{o,2}}{\partial r_x^{t+1}} & \frac{\partial h_{o,2}}{\partial r_y^{t+1}} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

Similarly for measurement function h_l:

$$h_l = l^k - r^t = \begin{bmatrix} l_x^k - r_x^t \\ l_y^k - r_y^t \end{bmatrix} = \begin{bmatrix} h_{l,1} \\ h_{l,2} \end{bmatrix}$$

where the states are: $X = [r_x^t, r_y^t, l_x^k, l_y^k]$. Therefore, its Jacobian matrix can be defined as:

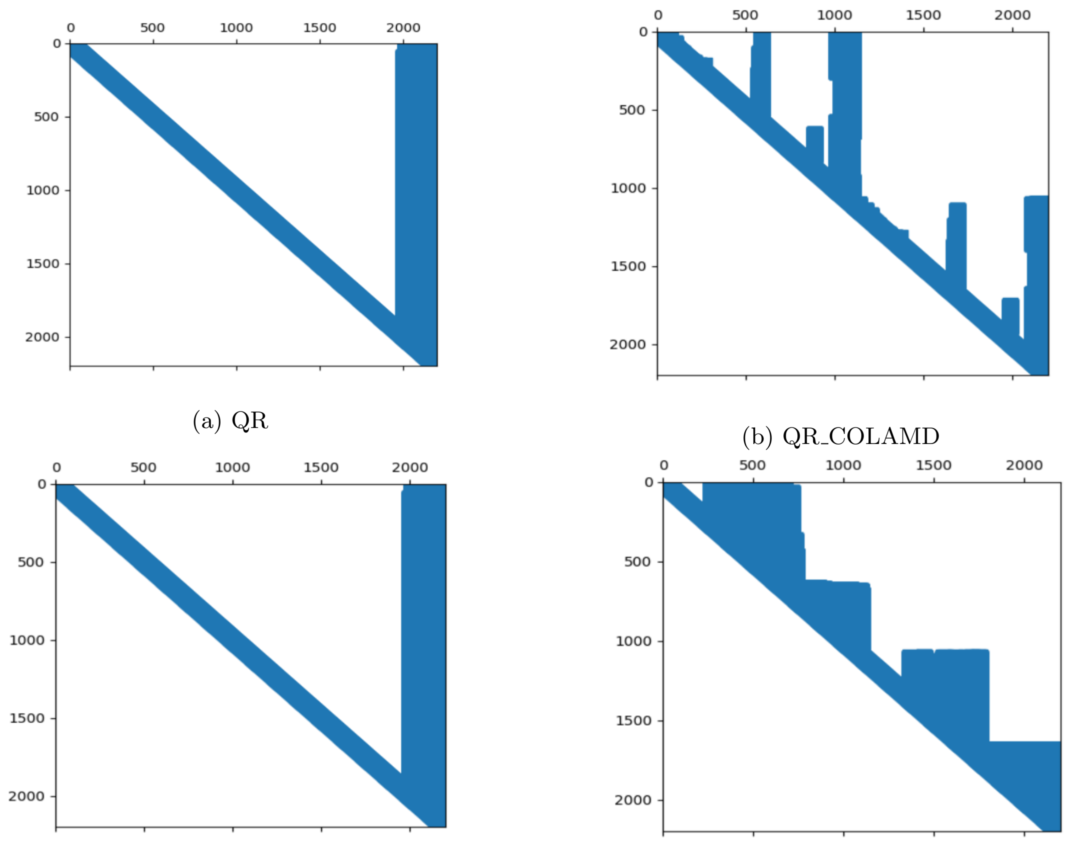
$$H_l = \frac{\partial h_l}{\partial X} = \begin{bmatrix} \frac{\partial h_{l,1}}{\partial r_x^t} & \frac{\partial h_{l,1}}{\partial r_y^t} & \frac{\partial h_{l,1}}{\partial l_x^k} & \frac{\partial h_{l,1}}{\partial l_y^k} \\ \frac{\partial h_{l,2}}{\partial r_x^t} & \frac{\partial h_{l,2}}{\partial r_y^t} & \frac{\partial h_{l,2}}{\partial l_x^k} & \frac{\partial h_{l,2}}{\partial l_y^k} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

Exercise 1.4.4

Table 1: 2d_linear.npz

Method	Average Time (s)
pinv	6.903
QR	1.088
QR_COLAMD	2.356
LU	0.1817
LU_COLAMD	0.1496

The table above shows that LU_COLAMD is the fastest method in solve the system in 2d_linear.npz. The order of solver efficiency ranks from LU_COLAMD being best, followed by LU, QR, QR_COLAMD, pinv in this order. The sparsity of the different decomposition methods can be seen in Figure 1 below. From the figures, it can be seen that there is a correlation in the decrease of solving time and sparsity of the matrix. It was, however, interesting to see that QR manages to finish faster than QR_COLAMD despite not having its columns reorganized for a faster traverse through its back-end column tracking. Going back to the comparison between QR and LU decompositions, it is a known fact that QR back-substitution makes it so that it is more numerically stable than the LU solver at the cost of solving time [3]. Hence it is expected for LU-class solvers to find a solution for the system of equations much faster QR-class solvers. The visualization of the trajectory and landmarks from the solvers can be referenced in the Appendix. In summary, all 4 methods can generate similar estimation as observed in the Appendix.


(a) QR

(b) QR_COLAMD

(c) LU

(d) LU_COLAMD

Figure 1: Visualization of all triangular matrices in 2d_linear.npz

Exercise 1.4.5

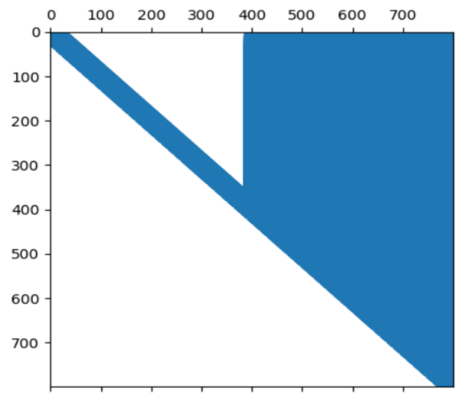
Table 2: 2d_linear_loop.npz

Method	Average Time (s)
pinv	0.512
QR	0.717
QR_COLAMD	0.098
LU	0.060
LU_COLAMD	0.009

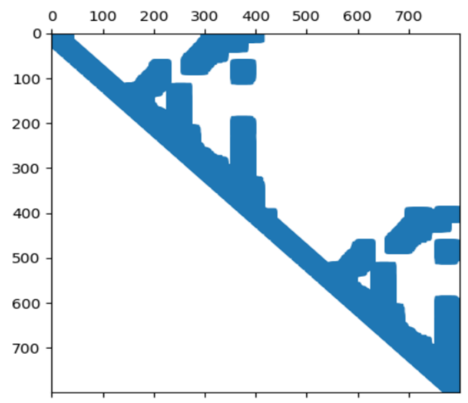
In the loop case, it can be seen that the general characteristics of LU and QR decomposition retains, meaning that LU-class solvers are significantly faster than QR-class solvers. However, pseudo-inverse was actually faster than plain QR solver in this test case. I believe it is due to the fact that both QR and LU triangular matrices are rather dense by observing Figure 2. The initial decomposition and back-substitution in QR solver for a dense matrix are both detrimental in computation time. LU solver, however, benefits from prior factorization and can still solve the system in a relatively short amount of time albeit being a denser matrix. Similar to the previous linear test case, all 4 methods can generate similar estimation.

Exercise 2.1
From the provided equation in Equation 2 in the problem statement, the Jacobian in respect to state $X = [r_x^t,\ r_y^t,\ l_x^k,\ l_y^k]^T$ is as follow as below:

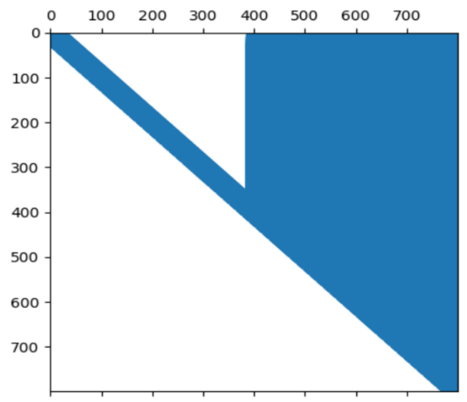
$$H_l = \frac{\partial h_l}{\partial X} = \begin{bmatrix} \frac{l_y^k - r_y^t}{(l_x^k-r_x^t)^2+\left(l_y^k-r_y^t\right)^2} & -\frac{l_x^k - r_x^t}{(l_x^k-r_x^t)^2+\left(l_y^k-r_y^t\right)^2} & -\frac{l_y^k - r_y^t}{(l_x^k-r_x^t)^2+\left(l_y^k-r_y^t\right)^2} & \frac{l_x^k - r_x^t}{(l_x^k-r_x^t)^2+\left(l_y^k-r_y^t\right)^2} \\ -\frac{l_x^k - r_x^t}{\sqrt{(l_x^k-r_x^t)^2+\left(l_y^k-r_y^t\right)^2}} & -\frac{l_y^k - r_y^t}{\sqrt{(l_x^k-r_x^t)^2+\left(l_y^k-r_y^t\right)^2}} & \frac{l_x^k - r_x^t}{\sqrt{(l_x^k-r_x^t)^2+\left(l_y^k-r_y^t\right)^2}} & \frac{l_y^k - r_y^t}{\sqrt{(l_x^k-r_x^t)^2+\left(l_y^k-r_y^t\right)^2}} \end{bmatrix}$$



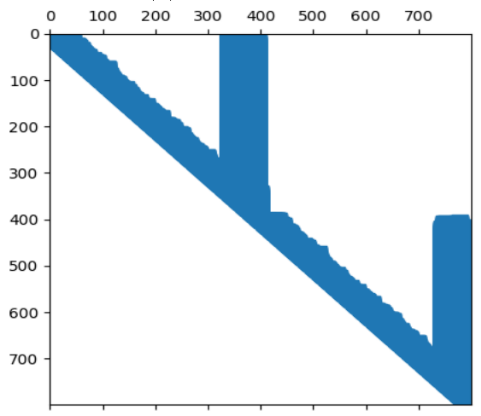
(a) QR



(b) QR_COLAMD



(c) LU



(d) LU_COLAMD

Figure 2: Visualization of all triangular matrices in 2d_linear_loop.npz

Exercise 2.3
The visualization of trajectory and landmarks before and after optimization using LU_COLAMD can be seen in Figure 14 & 15 in the Appendix. In the linear case, the least-square error solution Θ can be isolated and solved through decomposition methods in a batch method. However, it is impossible to realize the solution using the same methodology in the nonlinear case. The least-square error must be minimized incrementally through combination of Taylor series approximation and gradient descent algorithm. Once the nonlinear system is linearized to an acceptable error tolerance, only then can the solution be achieved.

Appendix

The trajectory and landmark visualization from all test cases are recorded in this section.

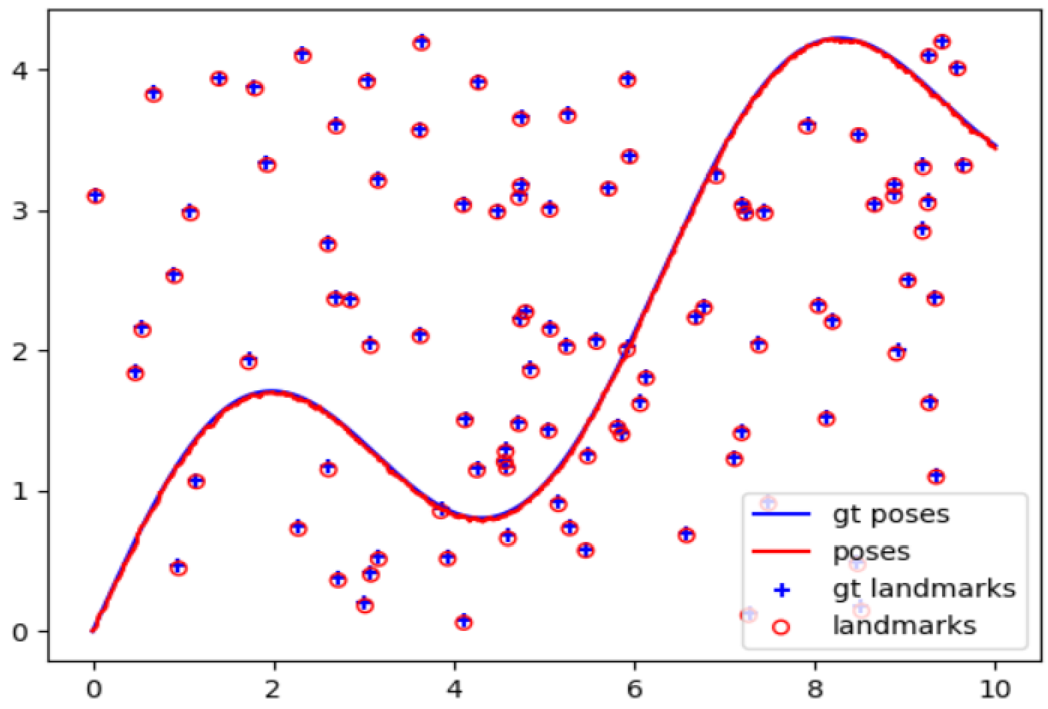


Figure 3: 2d_linear.npz estimation using QR

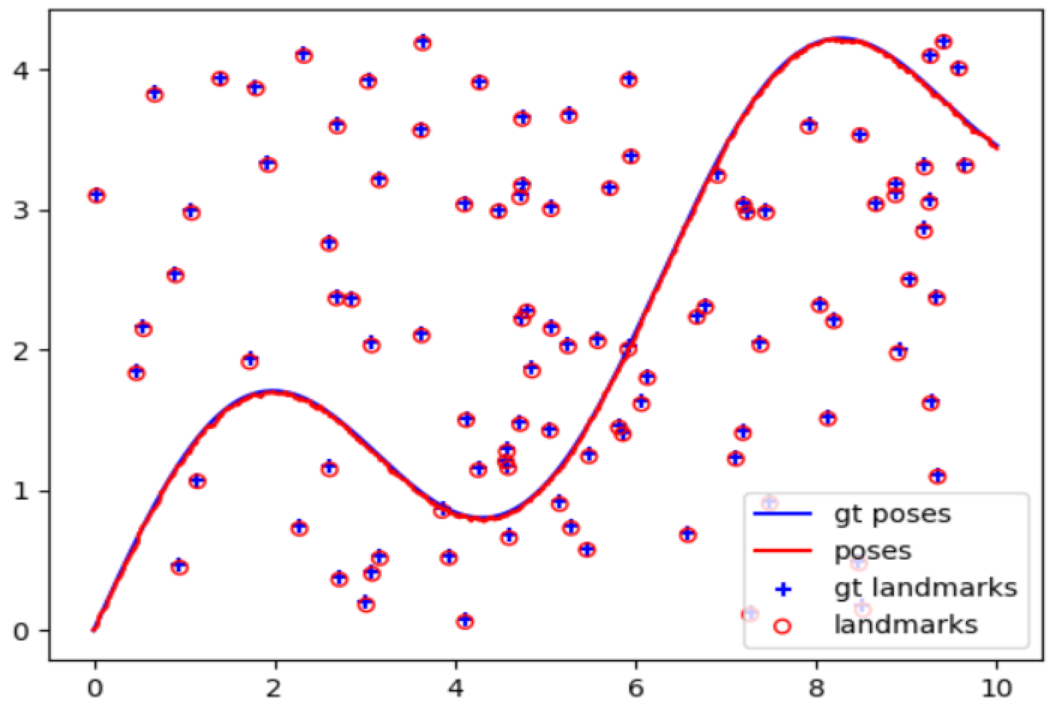


Figure 4: 2d_linear.npz estimation using QR_COLAMD

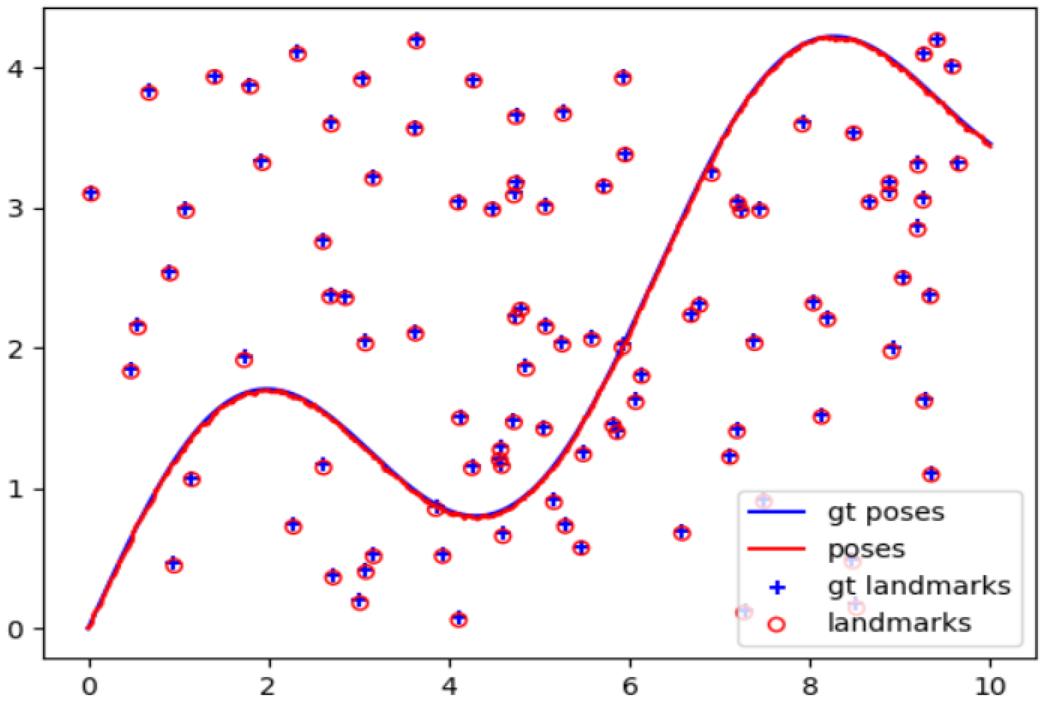


Figure 5: 2d_linear.npz estimation using LU

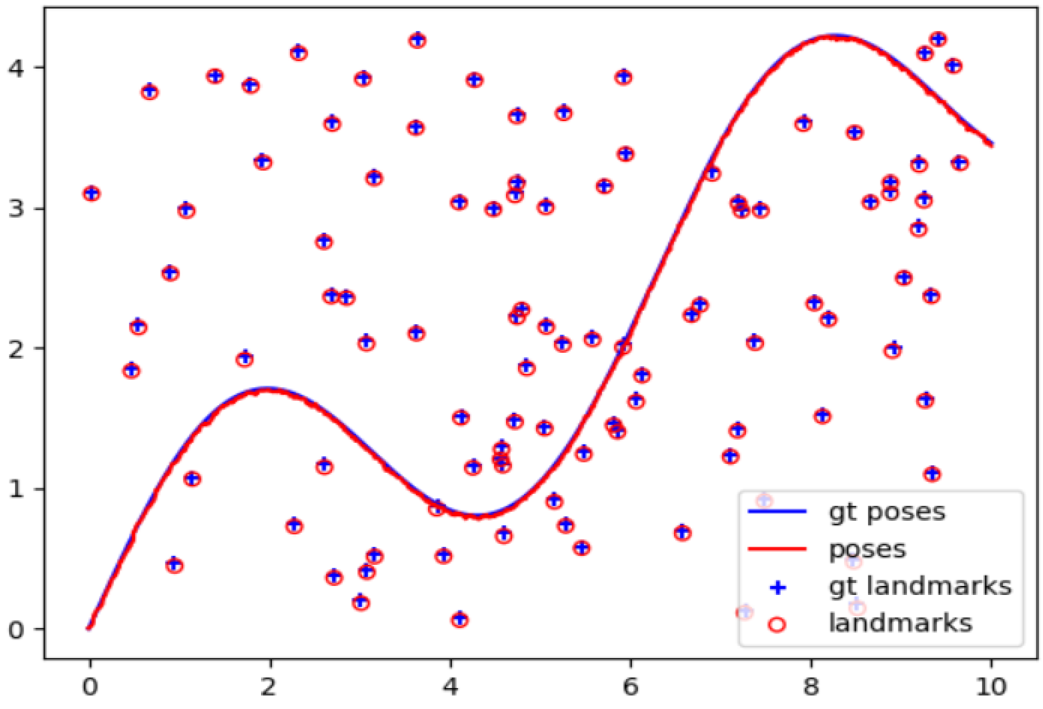


Figure 6: 2d_linear.npz estimation using LU_COLAMD

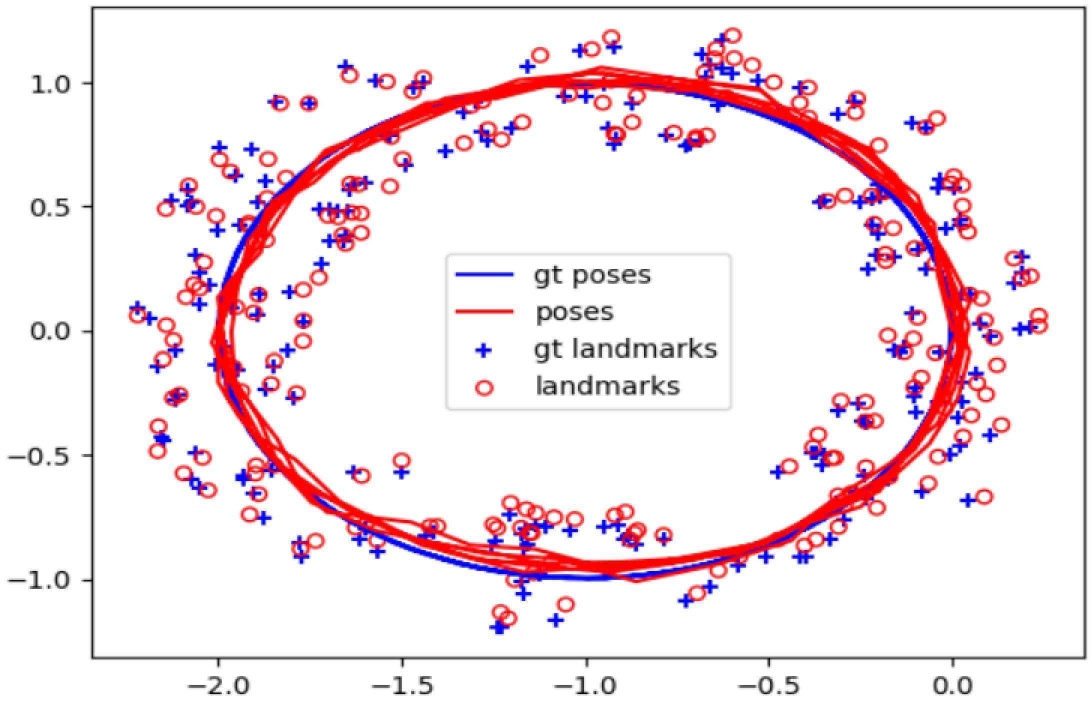


Figure 7: 2d_linear_loop.npz estimation using QR

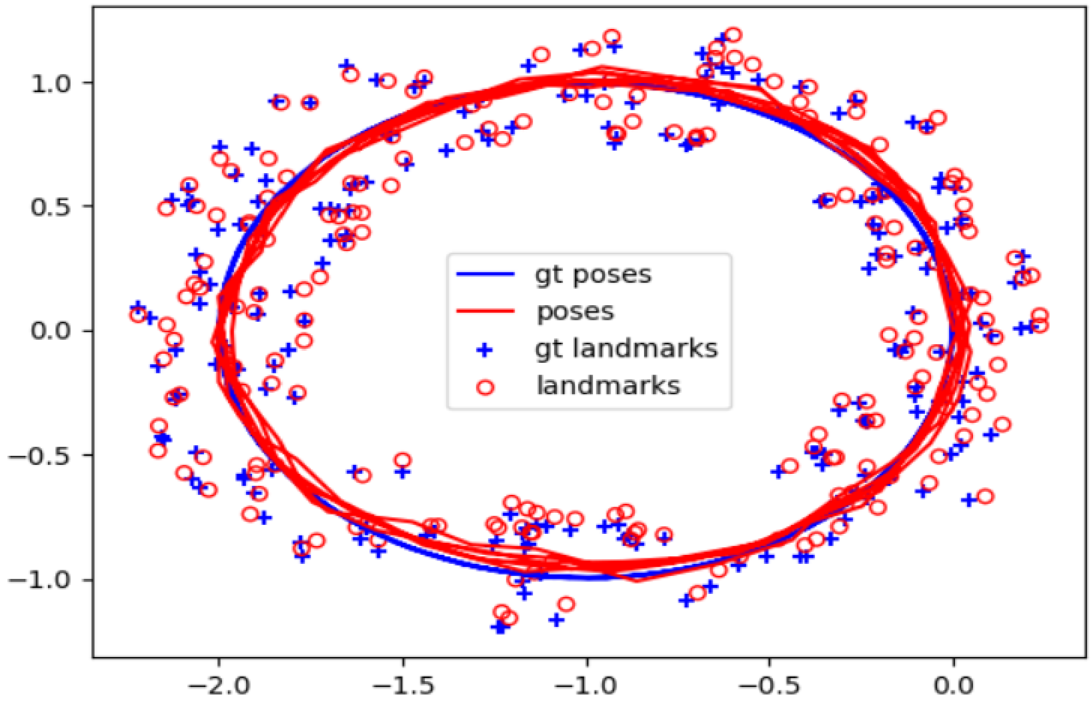


Figure 8: 2d_linear_loop.npz estimation using QR_COLAMD

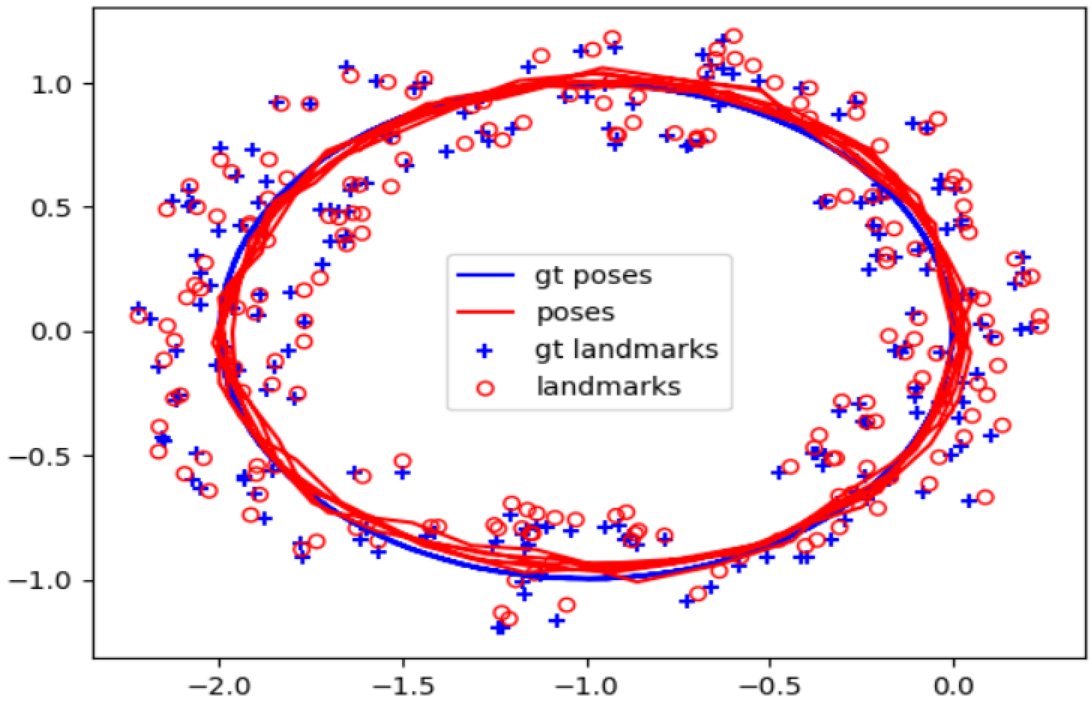


Figure 9: 2d_linear_loop.npz estimation using LU

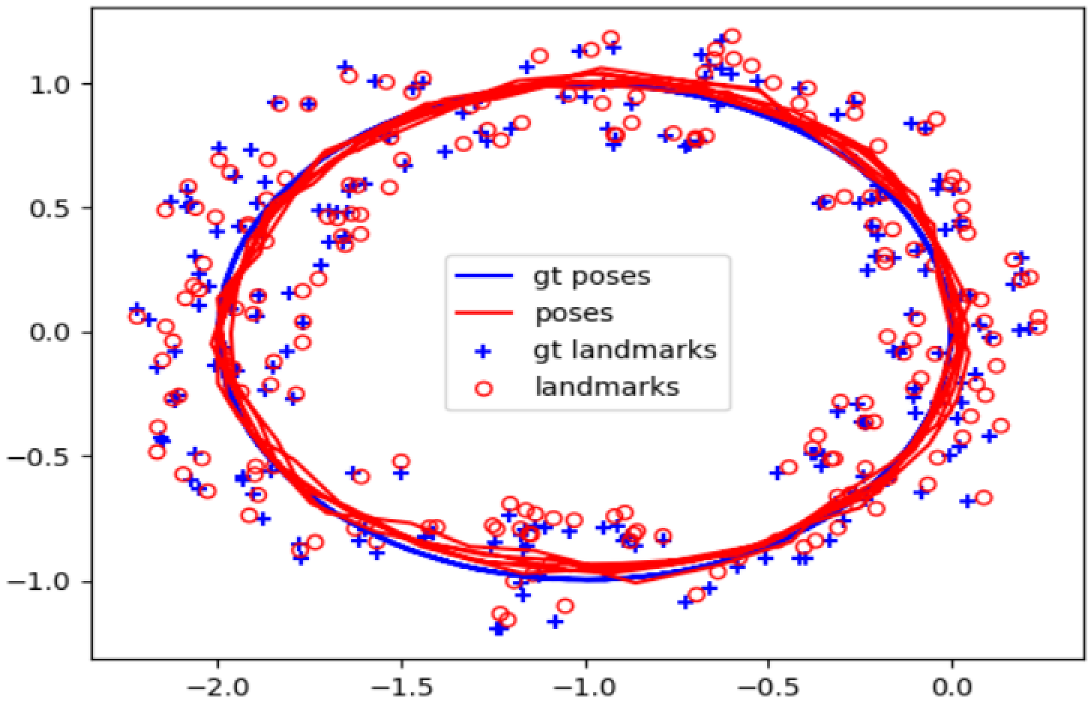


Figure 10: 2d_linear_loop.npz estimation using LU_COLAMD

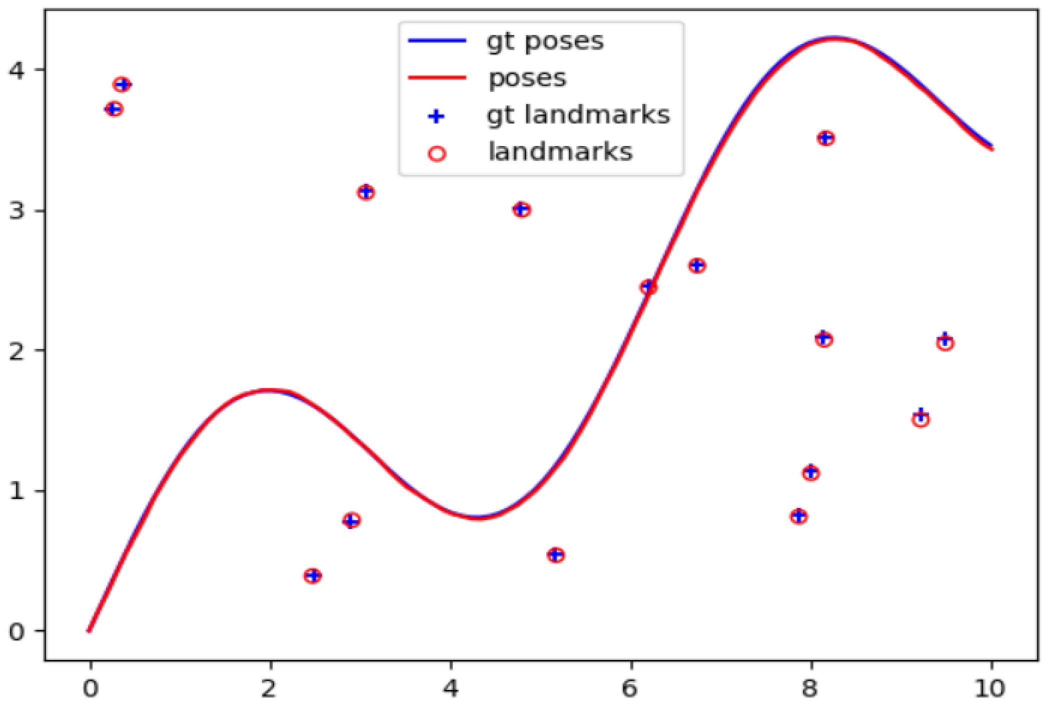


Figure 11: 2d_nonlinear.npz estimation using QR

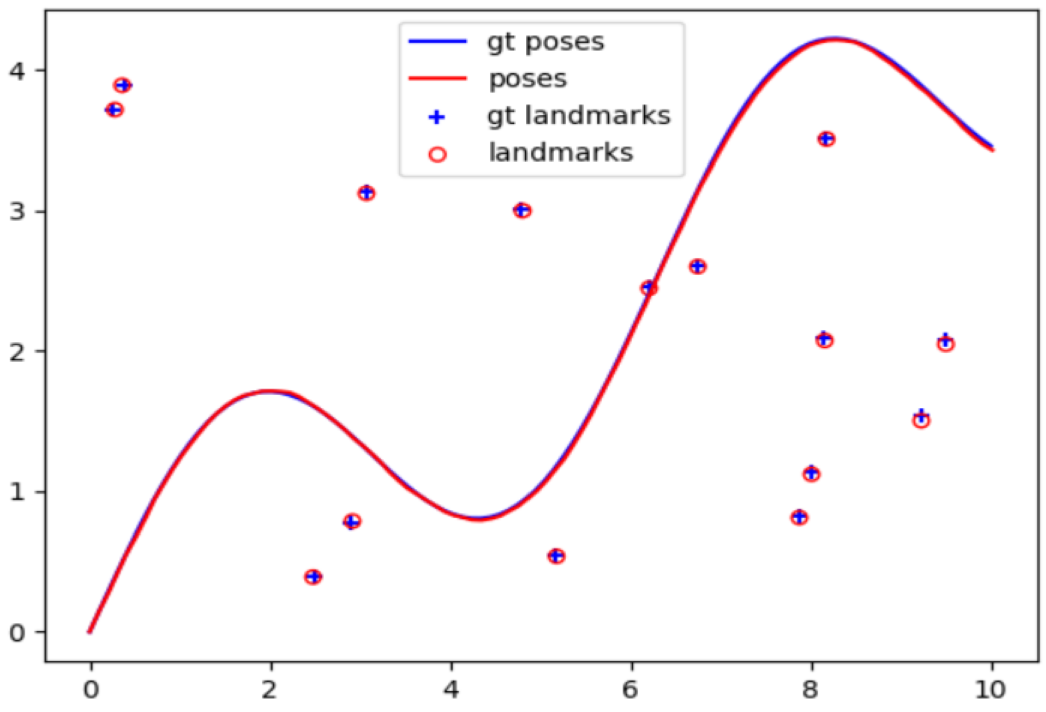


Figure 12: 2d_nonlinear.npz estimation using QR_COLAMD

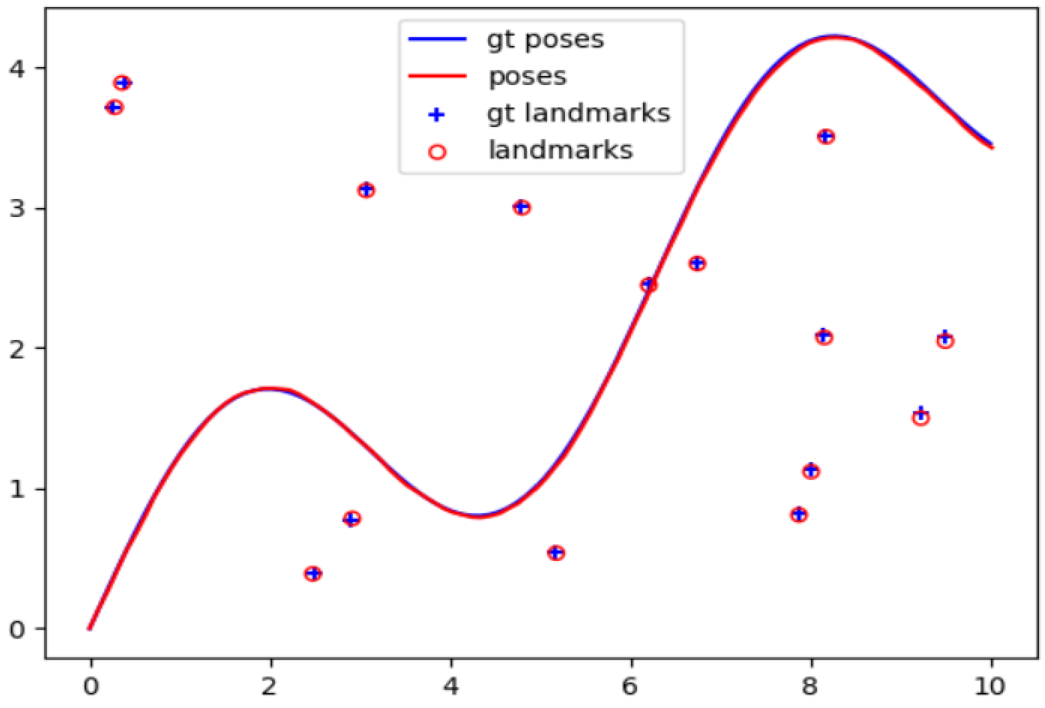


Figure 13: 2d_nonlinear.npz estimation using LU

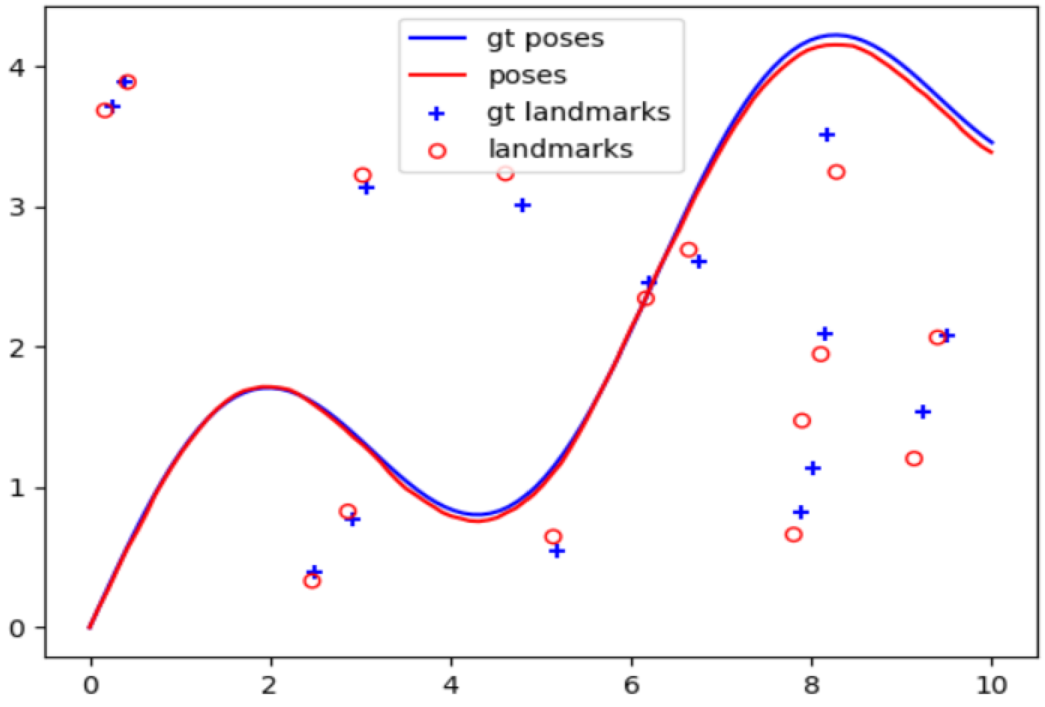


Figure 14: 2d_nonlinear.npz prior estimate optimization using LU_COLAMD

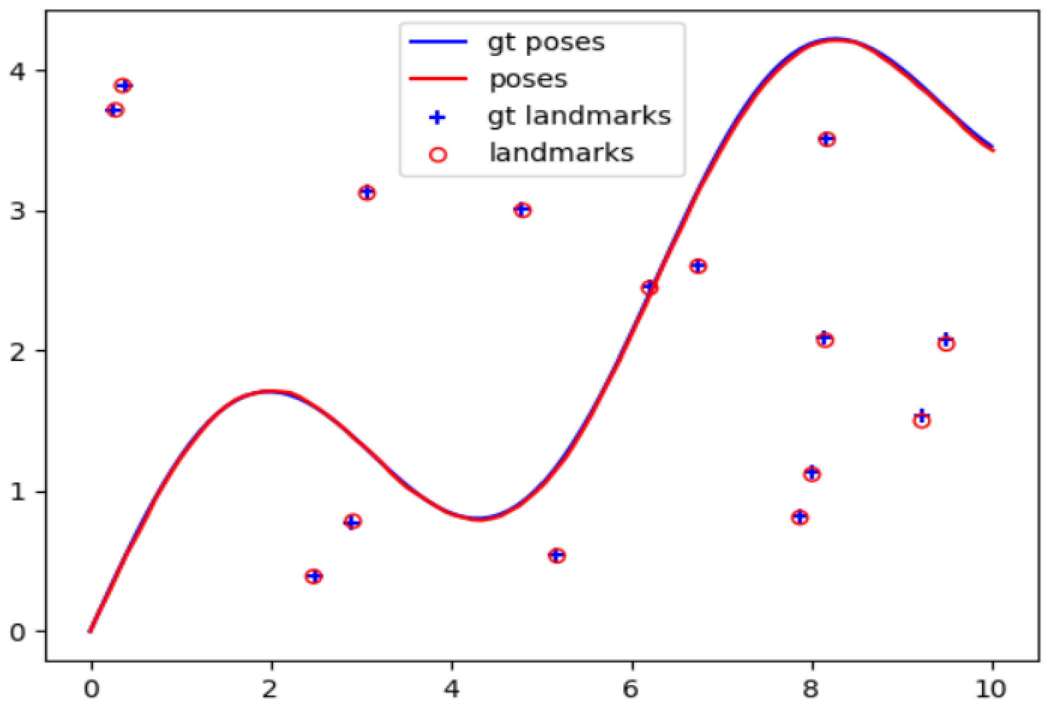


Figure 15: 2d_nonlinear.npz estimation using LU_COLAMD

References

[1] Kaess, M. *Session 12: SLAM: Exploiting Sparsity*, lecture recordings, 16-833 Robot Localization and Mapping, Carnegie Mellon University, delivered 10 March 2021.

[2] Kaess, M. *Session 13: Nonlinear Least-Squares*, lecture recordings, 16-833 Robot Localization and Mapping, Carnegie Mellon University, delivered 15 March 2021.

[3] "Matrix Decomposition" *Wikipedia*, Wikimedia Foundation, Updated 28 March 2021, https://en.wikipedia.org/wiki/Matrix_decomposition