SLAM HW3 Writeup Qishun Yu

1.

1.1 $r^t = \begin{bmatrix} r_x^t \\ r_y^t \end{bmatrix}$ $l^k = \begin{bmatrix} l_x^k \\ l_y^k \end{bmatrix}$

$r^{t+1} = \begin{bmatrix} r_x^{t+1} \\ r_y^{t+1} \end{bmatrix}$

$$h_0 = \begin{bmatrix} r_x^{t+1} - r_x^t \\ r_y^{t+1} - r_y^t \end{bmatrix}$$

$$H_0 = \begin{bmatrix} \frac{\partial h_{01}}{\partial r_x^t} & \frac{\partial h_{01}}{\partial r_y^t} & \frac{\partial h_{01}}{\partial r_x^{t+1}} & \frac{\partial h_{01}}{\partial r_y^{t+1}} \\ \frac{\partial h_{02}}{\partial r_x^t} & \frac{\partial h_{02}}{\partial r_y^t} & \frac{\partial h_{02}}{\partial r_x^{t+1}} & \frac{\partial h_{02}}{\partial r_y^{t+1}} \end{bmatrix}$$

$$= \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

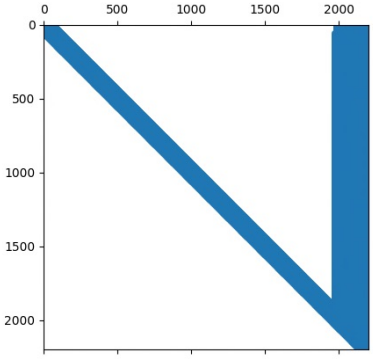
$$h_l = \begin{bmatrix} l_x^k - r_x^t \\ l_y^k - r_y^t \end{bmatrix}$$

$$H_l = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

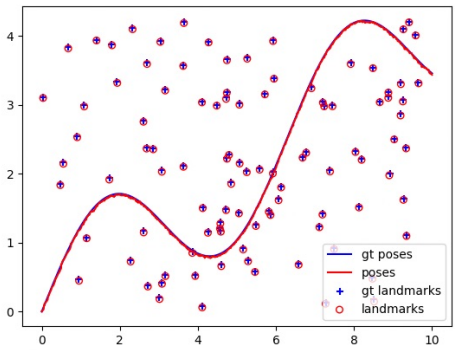
1.4

data: 2d_linear.npz

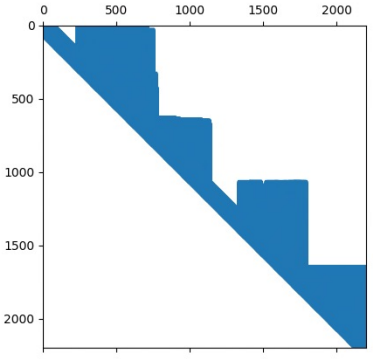
Method	Time on average
LU	0.058713 s
LU colamd	0.126654 s
QR	0.804530 s
QR Colamd	0.779370 s



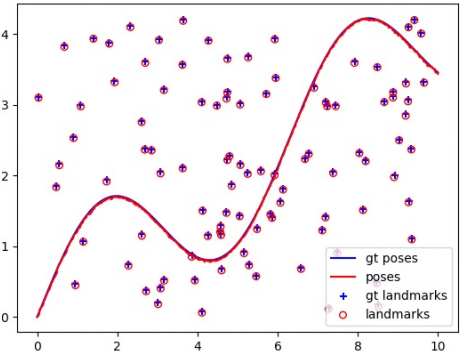
LU matrix



LU Result

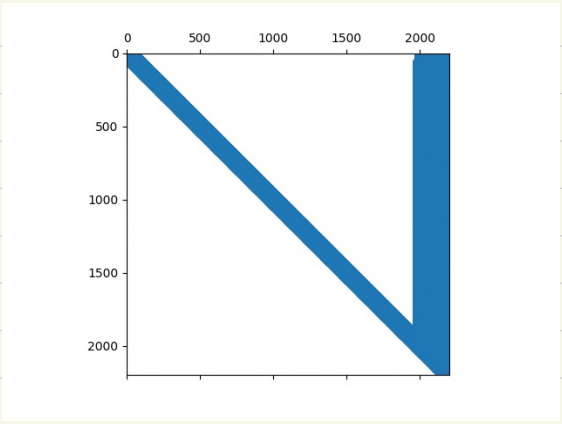


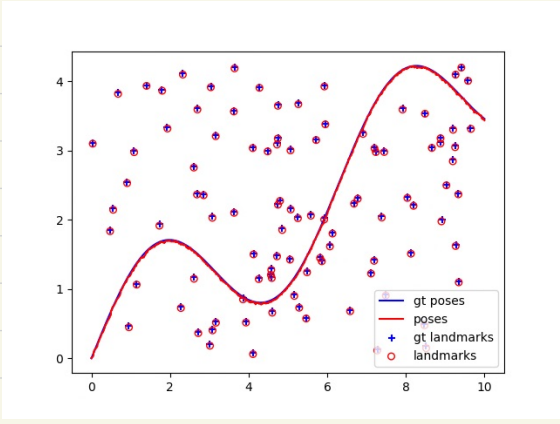
LU colamd matrix



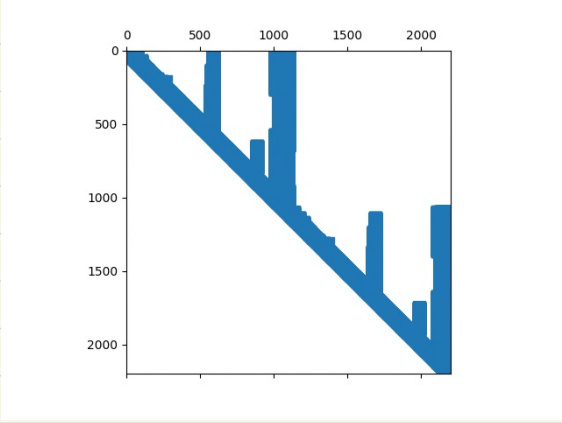
LU colamd Result

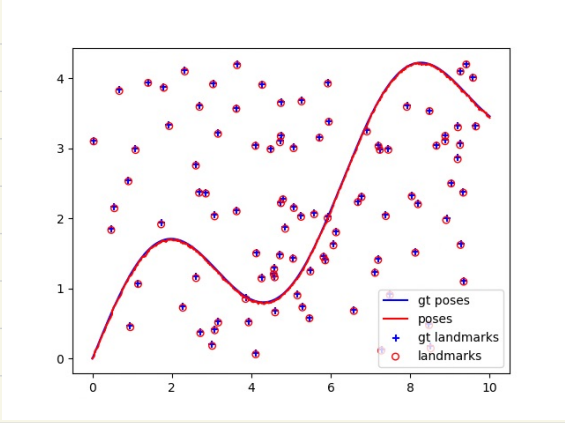
QR:





QR colamd





The resulting poses / landmarks estimation is similar. Generally, LU factorization is faster than QR factorization. The efficiency is:

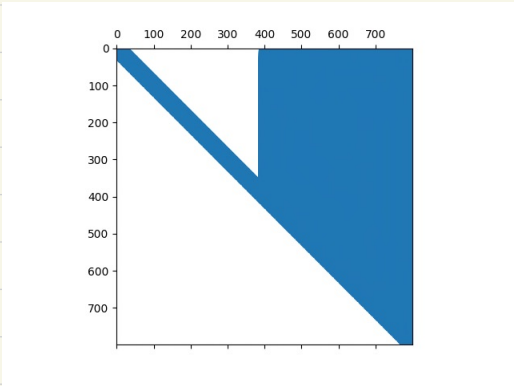
LU > LU colamd > QR colamd > QR

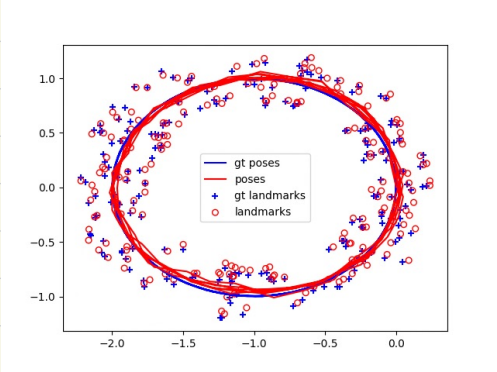
As mentioned in the lecture, QR factorization is more numerically stable and slower ($\frac{2}{3}m^3$ to $\frac{4}{3}m^3$ time complexity) The reason why LU outperform LU colamd could be the LU matrix is more dense than the LU colamd matrix

data: 2d_linear_loop.npz

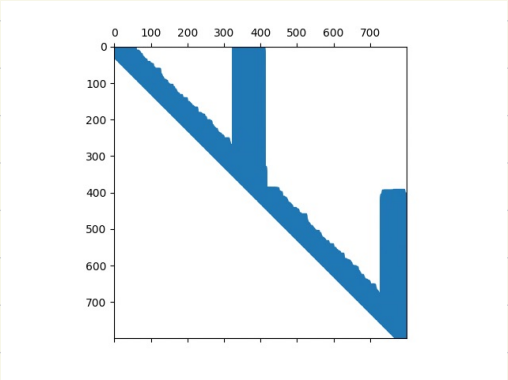
Method	Time on average
LU	0.075467 s
LU Colamd	0.008115 s
QR	0.425134 s
QR Colamd	0.049823 s

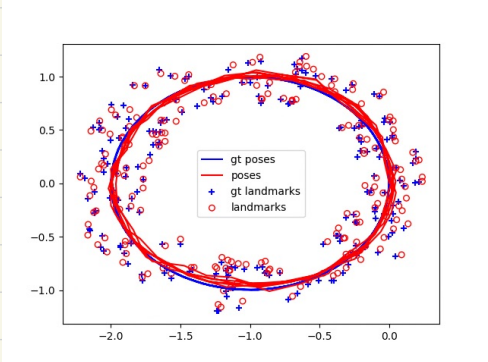
LU:



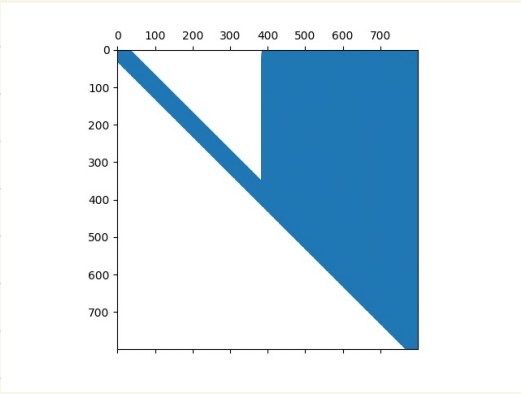


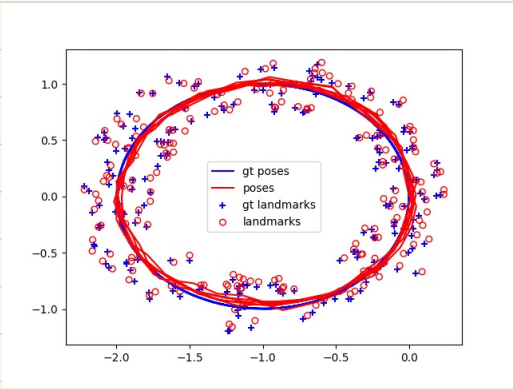
LU Colamd



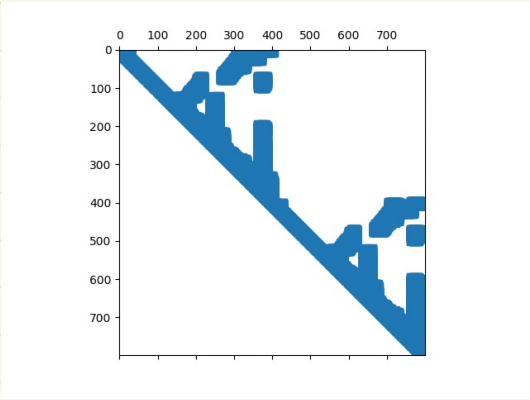


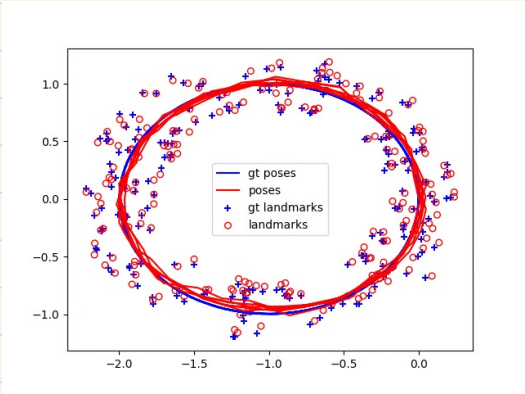
QR





QR Colamd





The results are similar to the previous case. LU factorization is faster than QR factorization. However, both LU Colamd and QR colamd are faster than the original factorization. The reason could be the A matrix is more dense in loop than non-loop environment. Therefore the minimum degree algorithm is able to reduce more arithmetic operations.

LU colmad > LU > QR colamd > QR

2.

2.1.2

$$h_l = \begin{bmatrix} \arctan2\,(l_y - r_y,\ l_x - r_x) \\ \sqrt{(l_y - r_y)^2 + (l_x - r_x)^2} \end{bmatrix}$$

$$H_l = \begin{bmatrix} \frac{\partial h_{l1}}{\partial r_x} & \frac{\partial h_{l1}}{\partial r_y} & \frac{\partial h_{l1}}{\partial l_x} & \frac{\partial h_{l1}}{\partial l_y} \\ \frac{\partial h_{l2}}{\partial r_x} & \frac{\partial h_{l2}}{\partial r_y} & \frac{\partial h_{l2}}{\partial l_x} & \frac{\partial h_{l2}}{\partial l_y} \end{bmatrix}$$

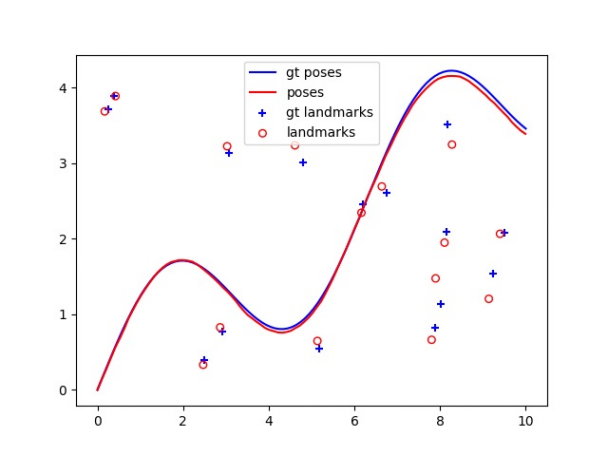
Set $l_y - r_y = dy$

$l_x - r_x = dx$

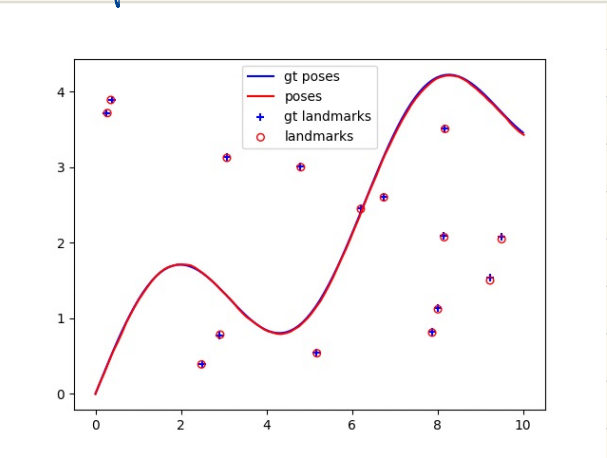
$$H_l = \begin{bmatrix} \frac{dy}{dy^2 + dx^2} & \frac{-dx}{dy^2 + dx^2} & \frac{-dy}{dy^2 + dx^2} & \frac{dx}{dy^2 + dx^2} \\ \frac{-dx}{(dy^2 + dx^2)^{\frac{1}{2}}} & \frac{-dy}{(dy^2 + dx^2)^{\frac{1}{2}}} & \frac{dx}{(dy^2 + dx^2)^{\frac{1}{2}}} & \frac{dy}{(dy^2 + dx^2)^{\frac{1}{2}}} \end{bmatrix}$$

2.3

before optimization



after optimization.



In linear least-square case. Solutions are directly solved by the normal equation. On the other hand, in non-linear case, we have to initialize a estimate, linearize the model and solve the states iteratively.