16-833: Robot Localization and Mapping, Spring 2021

Homework 3 - SLAM Solvers

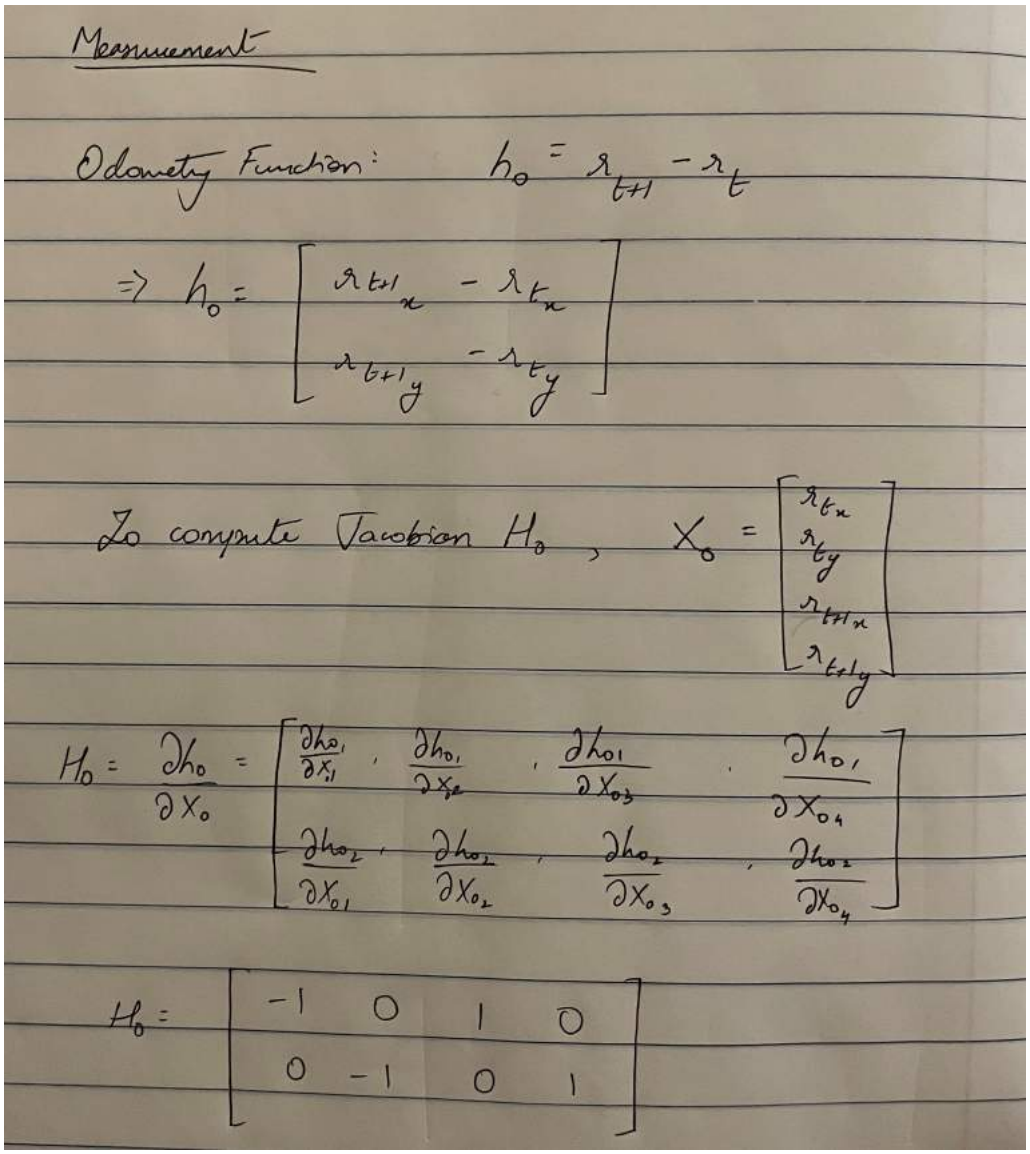
Aditya Ramakrishnan

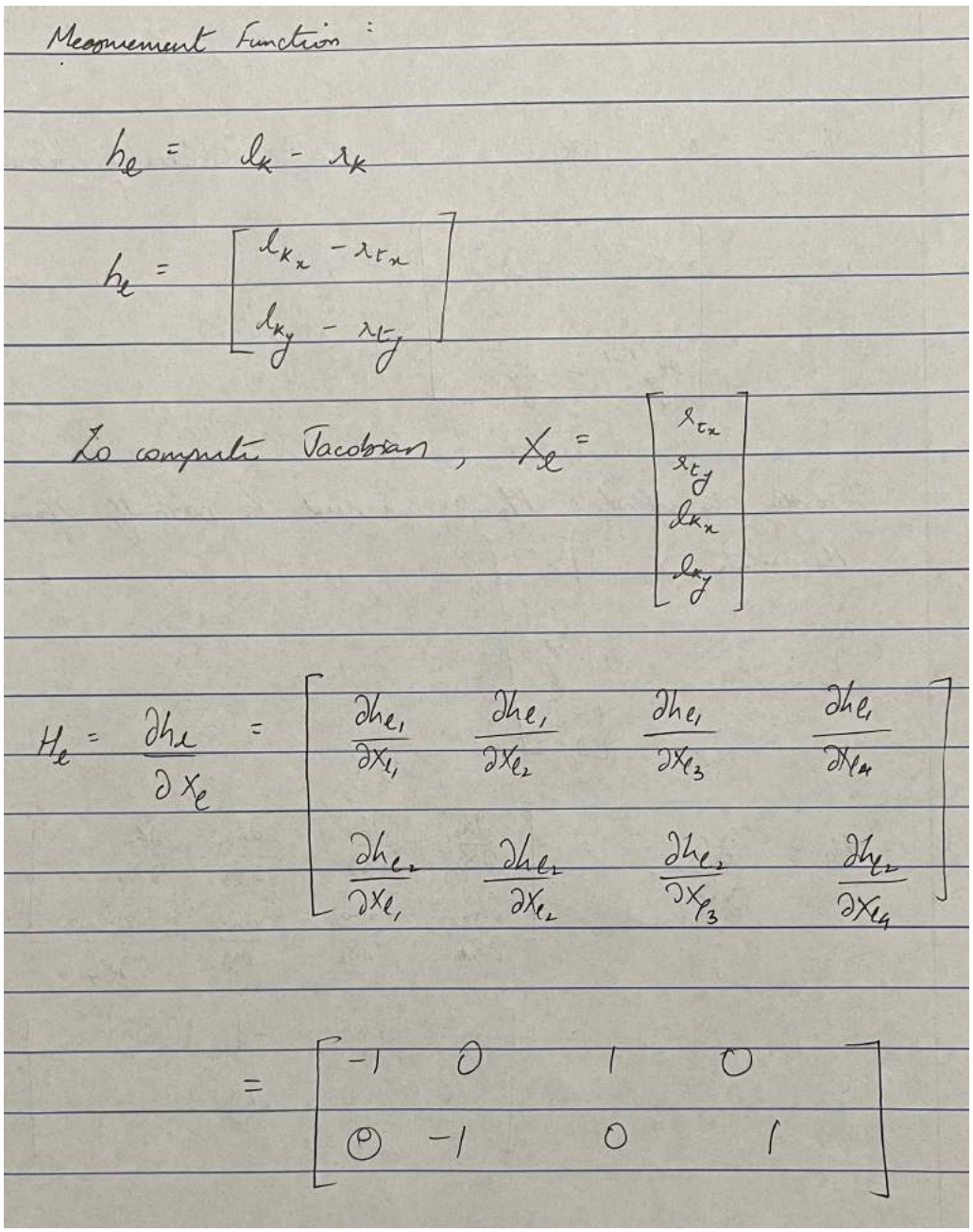
April 1, 2022

1 2D Linear SLAM

Q1.1 Measurement function

Given robot poses at time t and $t+1$, write out the measurement function and its Jacobian. Similarly, given the robot pose at time t and the k-th landmark $\mathbf{l}^k = [l_x^k,\ l_y^k]^\top$, find the Jacobian and measurement function.





Q1.4.4 Comparison of Linear Solvers

Running the 2*dlinear.npz* program shows that LU is the fastest solver in the system followed by LU_{COLAMD}, QR_{COLAMD}, QR, and P_{INV}. The sparsity of each method appears to present a relation between the decrements in solving time and sparsity of the matrix. The LU solver is clearly much faster than the QR solver, which is interesting as the LU solver is specifically used for square matrices and it is optimized to find the fastest solution, however, system impact includes issues in mathematical stability. Additionally, it is also observed that the QR solver is marginally faster than QR_{COLAMD} solver due to the implementation column tracking optimization in the latter. The results of these runs are shown in the following pages and the times for each solver are as below:

```
adi@adi-XPS-15-9570:~/SLAM/3/code$ python3 linear.py
Applying pinv
pinv takes 3.3977463245391846s on average
Applying qr
/home/adi/.local/lib/python3.8/site-packages/scipy/sp
  warn('CSR matrix format is required. Converting to
qr takes 0.8322124481201172s on average
Applying lu
lu takes 0.043293714523315 43s on average
Applying qr_colamd
/home/adi/.local/lib/python3.8/site-packages/scipy/sp
  warn('CSR matrix format is required. Converting to
qr_colamd takes 0.7072768211364746s on average
Applying lu_colamd
lu_colamd takes 0.14575886726379395s on average
```

LU solver :	0.0433 s
LU_COLAMD solver :	0.145 s
QR_COLAMD solver :	0.707 s
QR solver :	0.832 s
PINV solver :	3.397 s

Figure 1: Timestamps for Solvers

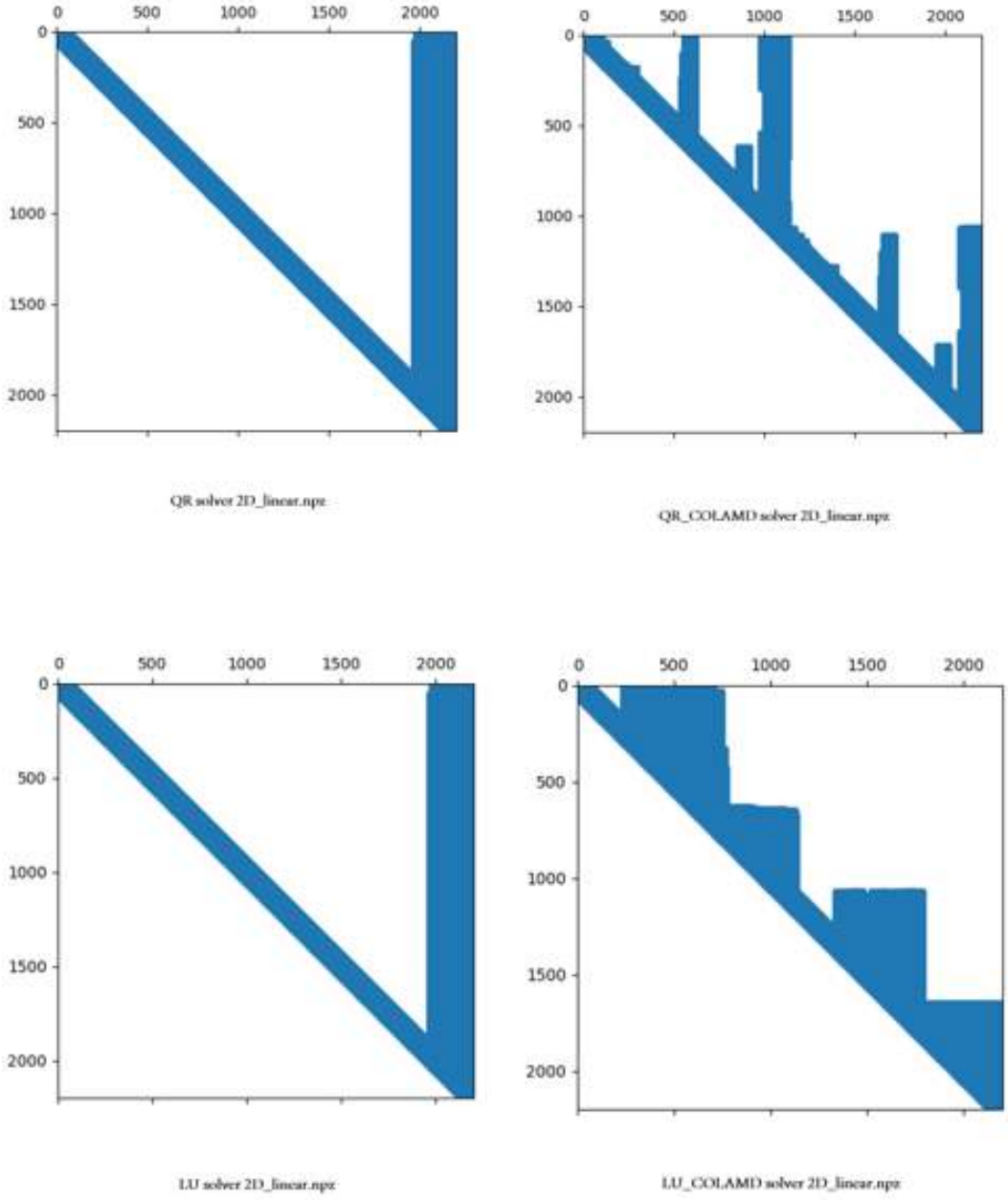
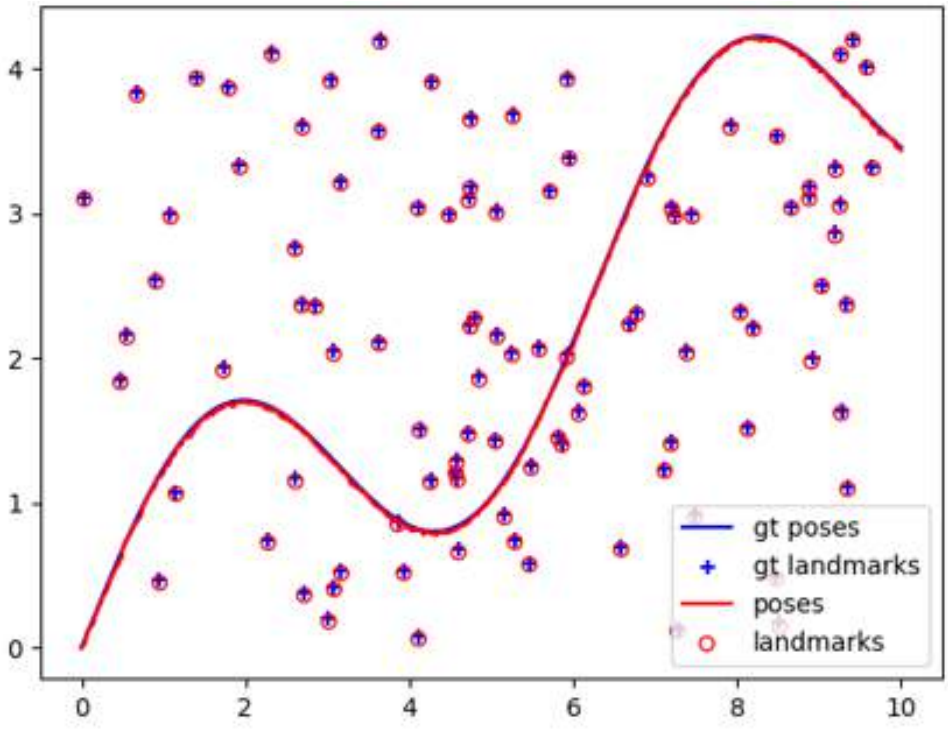
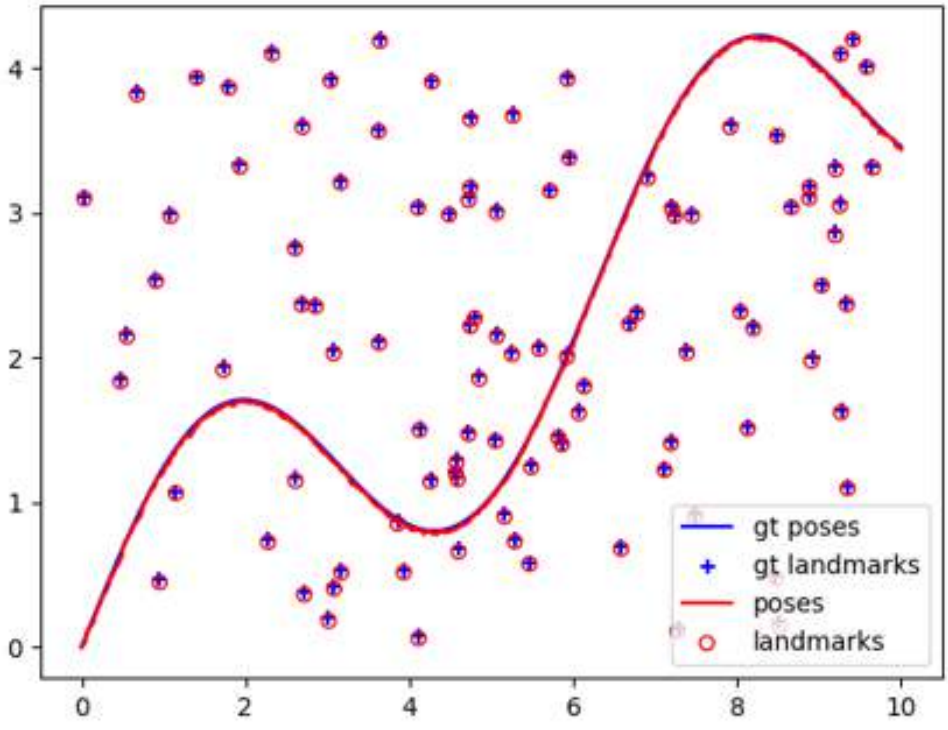


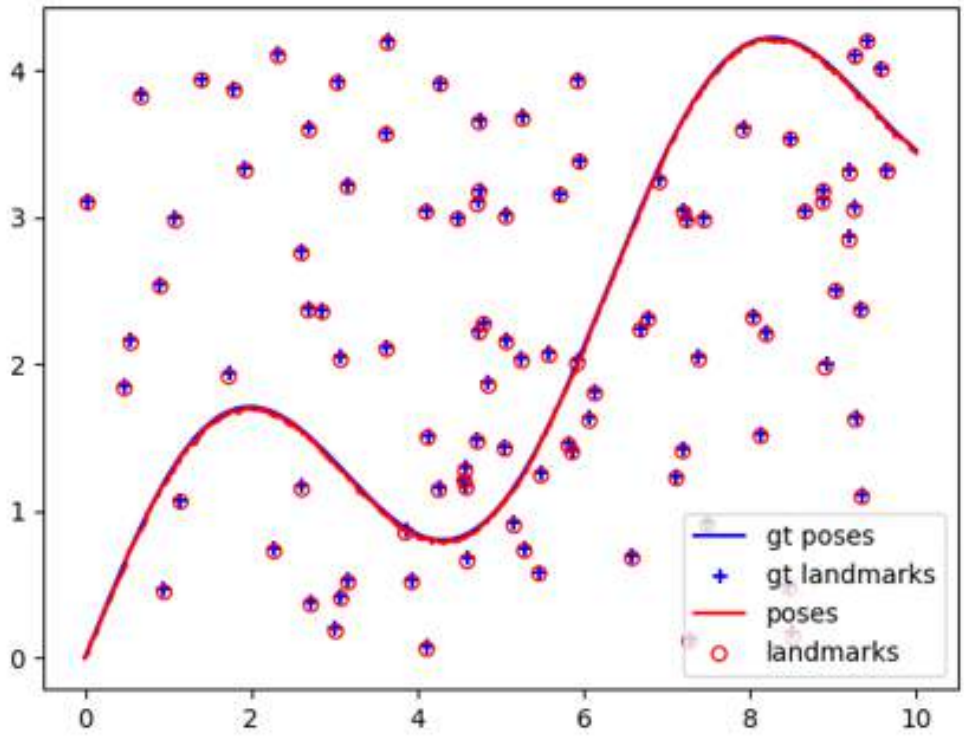
Figure 2: Sparsity Map of Solvers



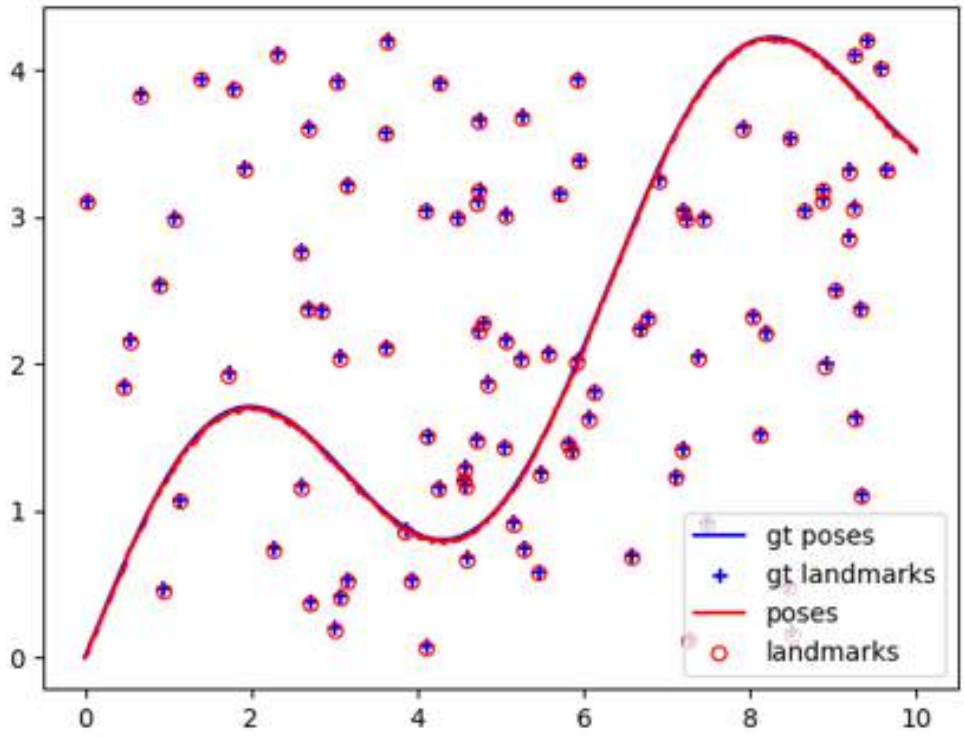
LU solver 2D_linear.npz



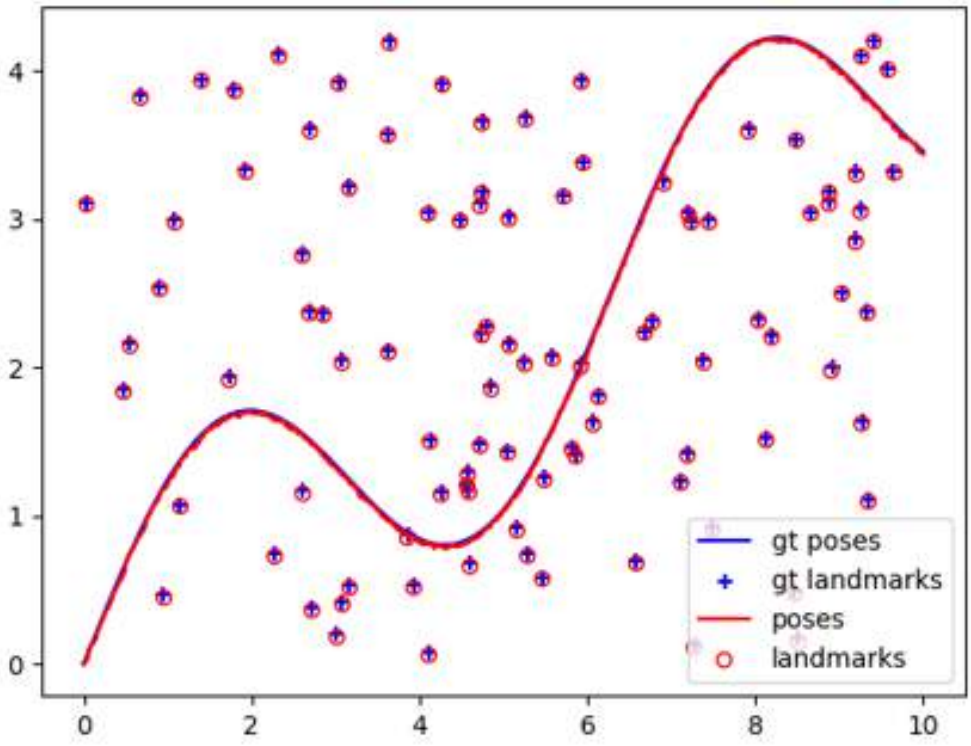
LU_COLAMD solver 2D_linear.npz



QR solver 2D_linear.npz



QR_COLAMD solver 2D_linear.npz

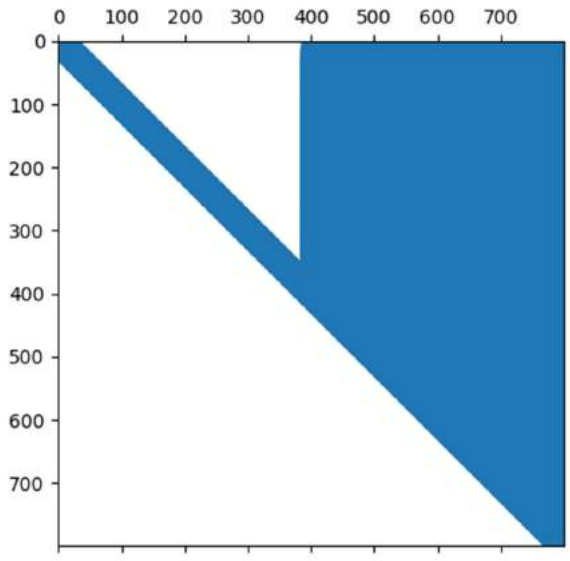


PINV solver 2D_linear.npz

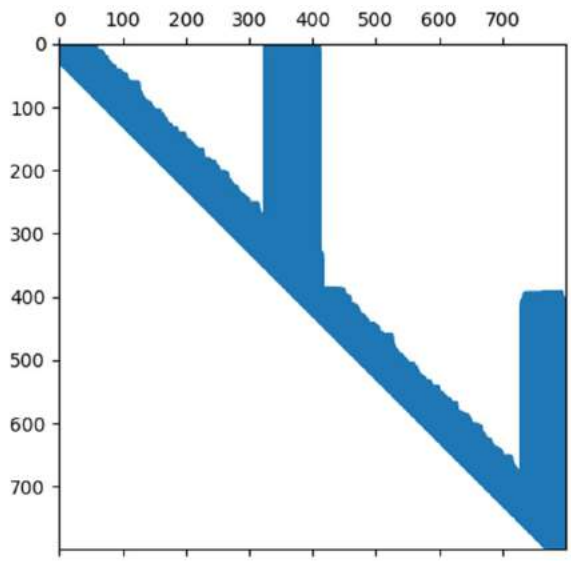
Q1.4.5 Comparison of Linear Loop Solvers

Running the 2*dlinearloop.npz* program shows a more or less similar performance to the 2*dlinear* case, i.e. the LU solver is faster compared to the QR solver. Interestingly, the pseudo-inverse P_{INV} solver appeared to be computationally faster than the QR solver even on repeated trials. This can be attributed to the matrix density of QR which is computationally expensive. The LU solver is the optimal choice when selecting a faster solver due to its prior factorisation. The estimation results for all 4 solver models can be seen below.

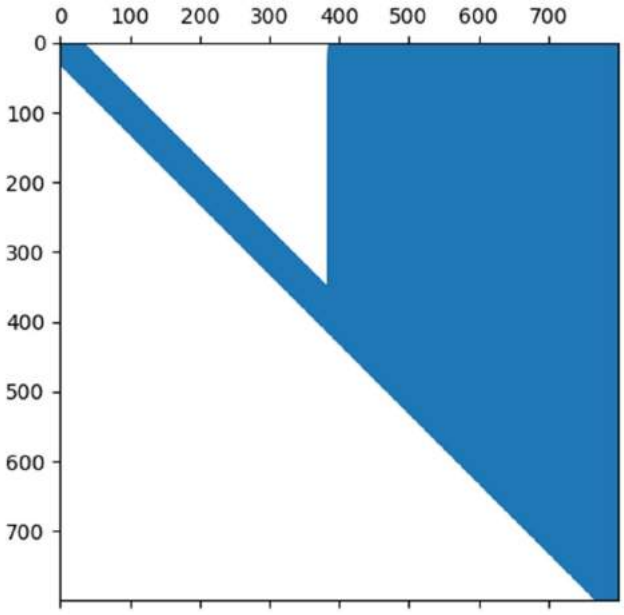
LU_COLAMD solver:	0.00575 s
LU solver:	0.0389 s
QR_COLAMD solver:	0.05 s
PINV solver:	0.3266 s
QR solver:	0.429 s



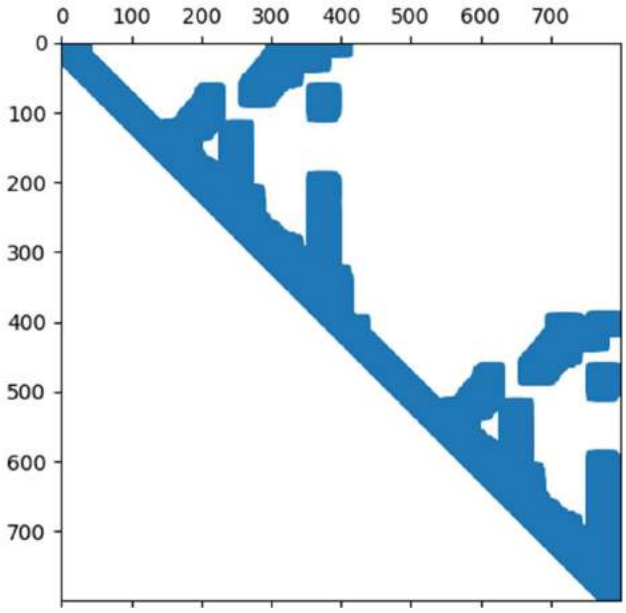
LU solver 2D_linear_loop.npz



LU_COLAMD solver 2D_linear_loop.npz

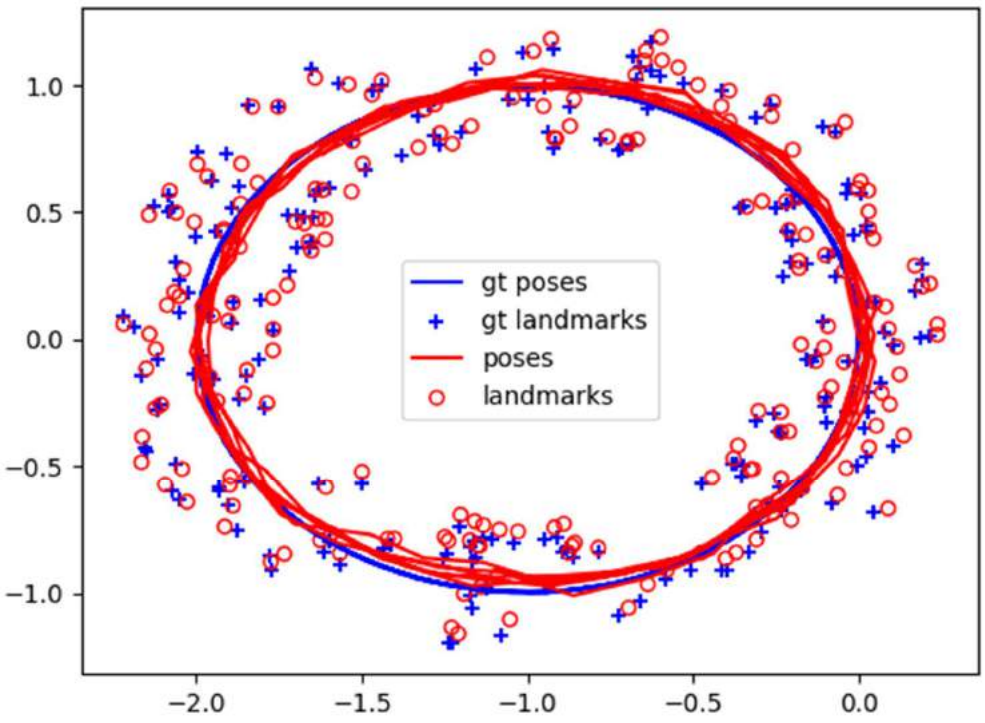


QR solver 2D_linear_loop.npz

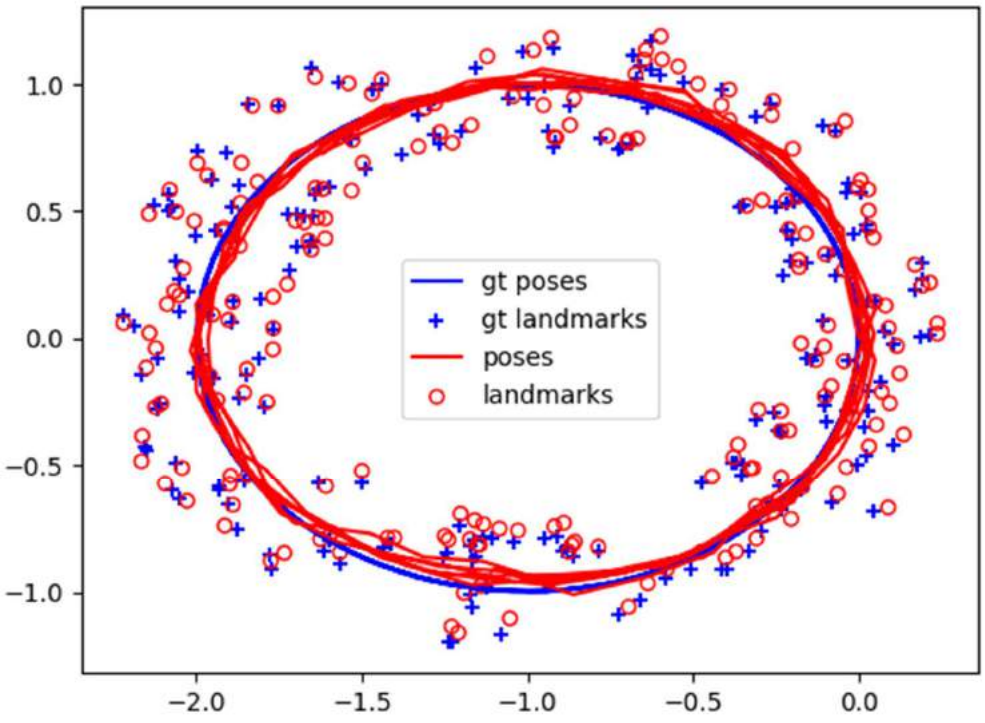


QR_COLAMD solver 2D_linear_loop.npz

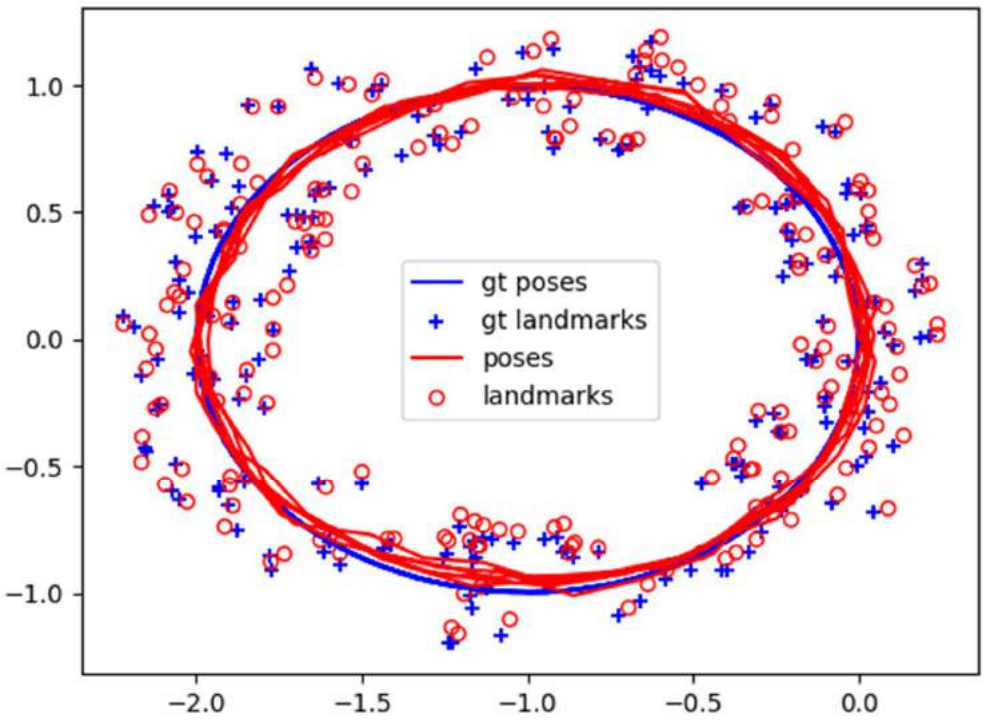
```
adi@adi-XPS-15-9570:~/SLAM/3/code$ python3 linear.py /home/adi/SLAM/3/
Applying pinv
pinv takes 0.32667112350463867s on average
Applying qr
/home/adi/.local/lib/python3.8/site-packages/scipy/sparse/linalg/_dsol
  warn('CSR matrix format is required. Converting to CSR matrix.',
qr takes 0.429970502853393555s on average
Applying lu
lu takes 0.038914442062377935s on average
Applying qr_colamd
/home/adi/.local/lib/python3.8/site-packages/scipy/sparse/linalg/_dsol
  warn('CSR matrix format is required. Converting to CSR matrix.',
qr_colamd takes 0.0500640869140625s on average
Applying lu_colamd
lu_colamd takes 0.005754232406616211s on average
```



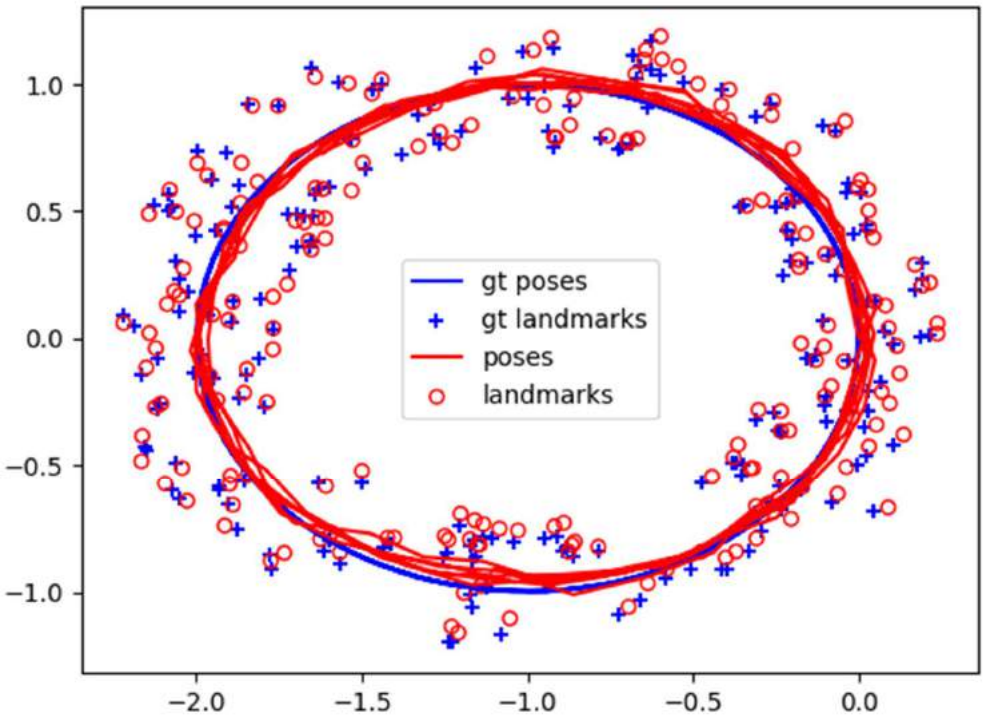
LU solver 2D_linear_loop.npz



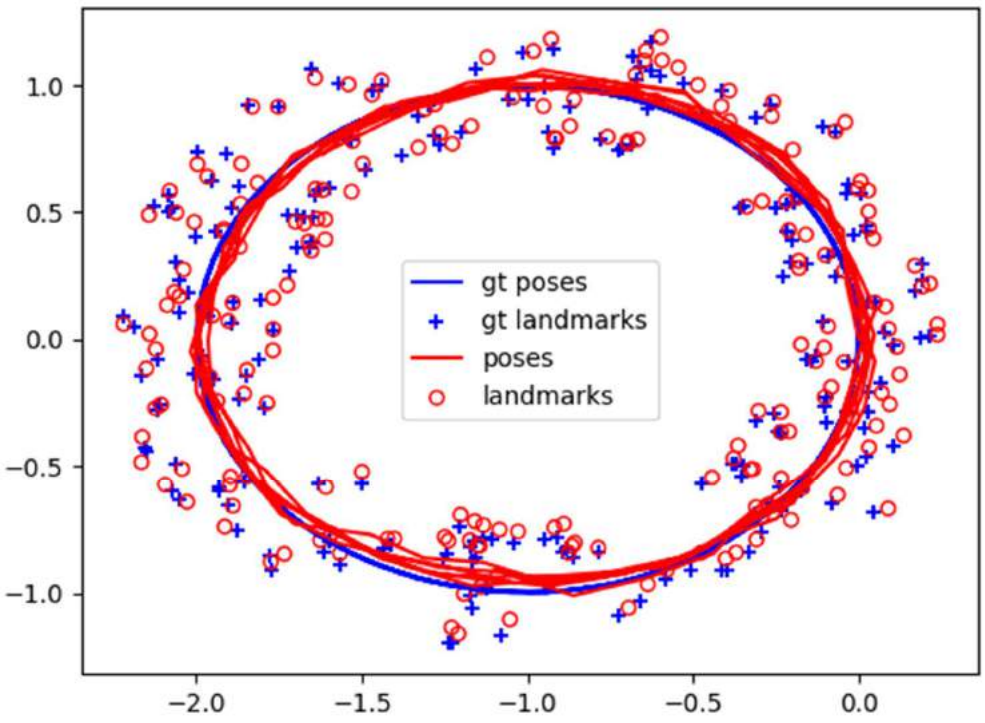
LU_COLAMD solver 2D_linear_loop.npz



QR solver 2D_linear_loop.npz

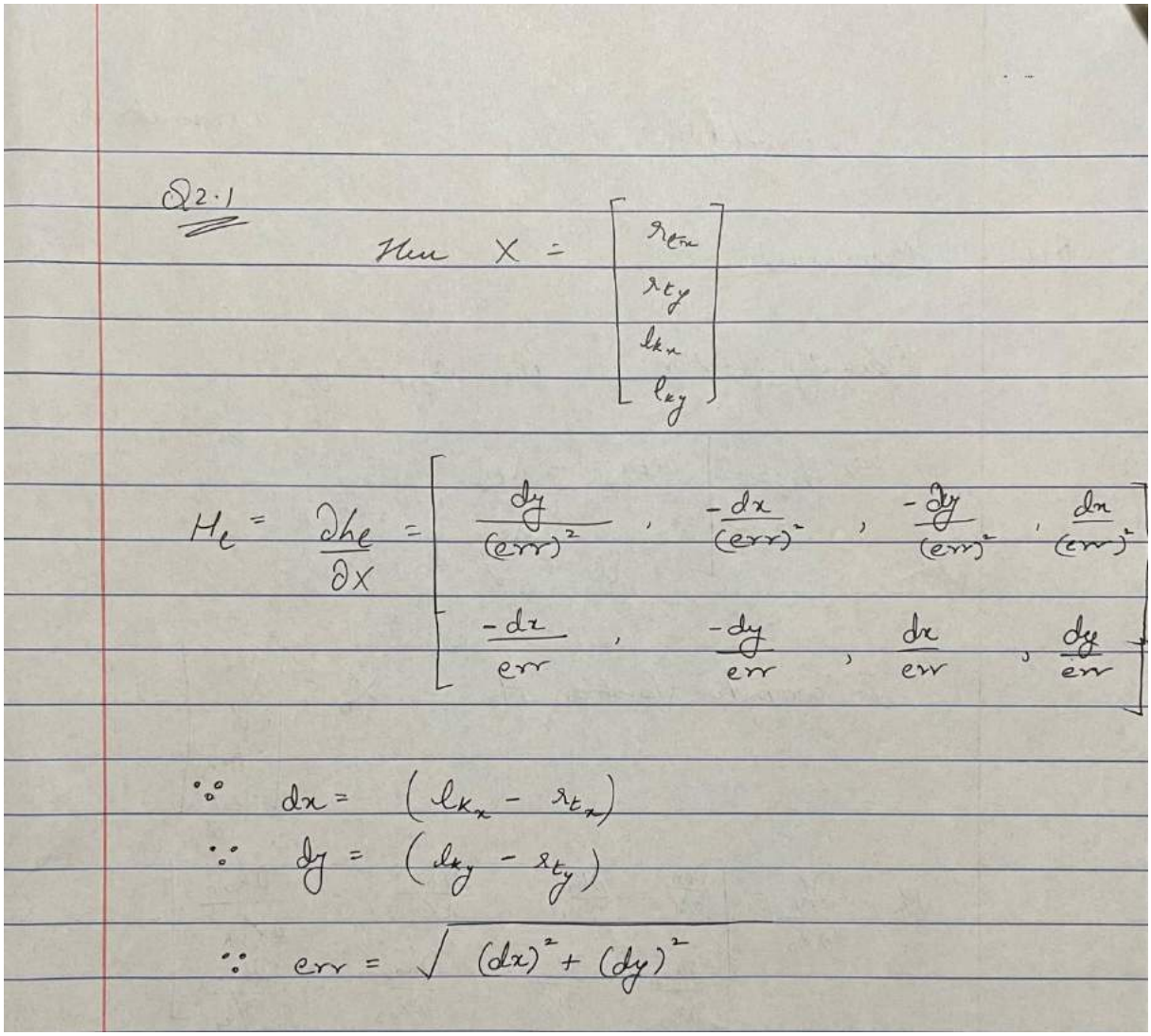


QR_COLAMD solver 2D_linear_loop.npz



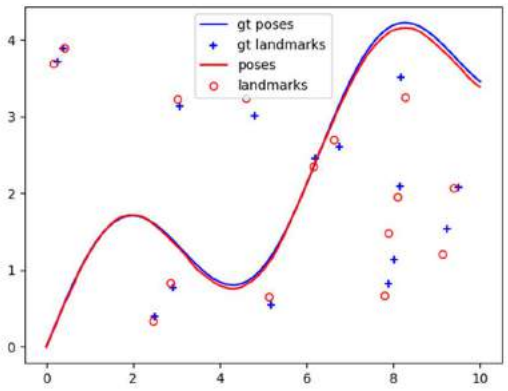
PINV solver 2D_linear_loop.npz

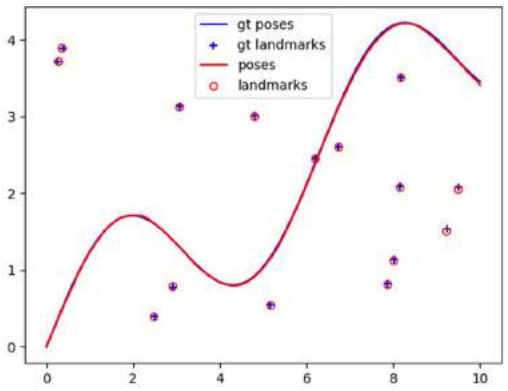
Q2.1 2D Non-Linear SLAM system Jacobian (measurement function)



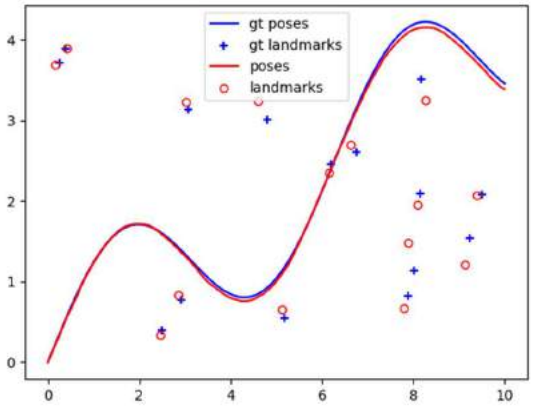
Q 2.3 2D-Non-Linear Solvers

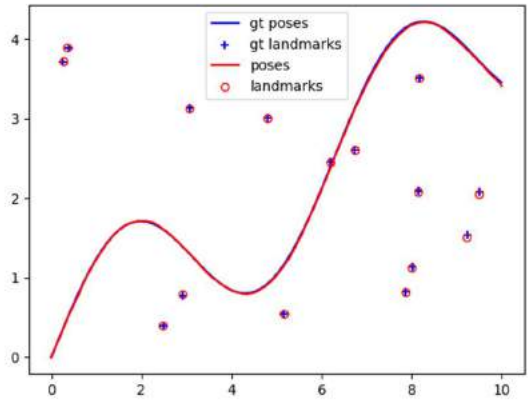
Running the *2d_nonlinear.npz* program outputs estimation plots for each of our solvers as shown in the following pages. The approach considered in this assignment involves computing the least-square error. This error must be minimized iteratively. The nonlinear system is first linearized as it is not possible to implement the decomposition otherwise. Below all 4 non-linear landmark and trajectories can be seen for the solvers are shown, both prior to, and after optimization.



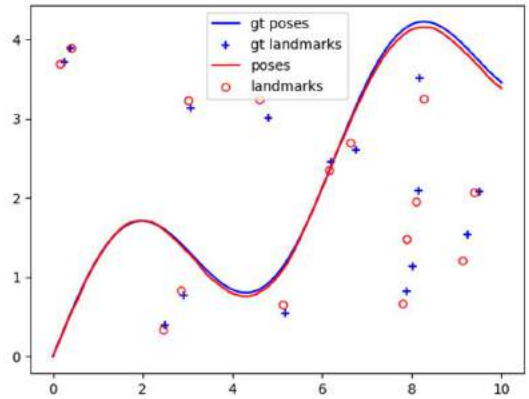


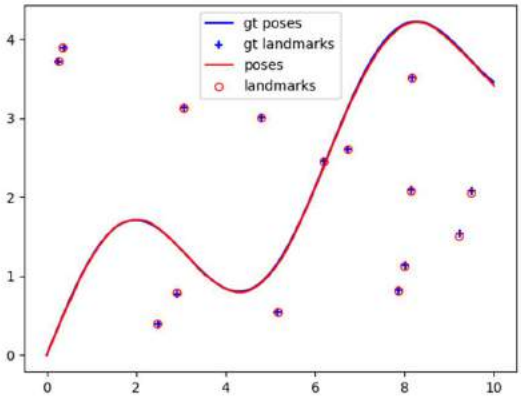
PINV 2D non-linear estimation before and after optimization



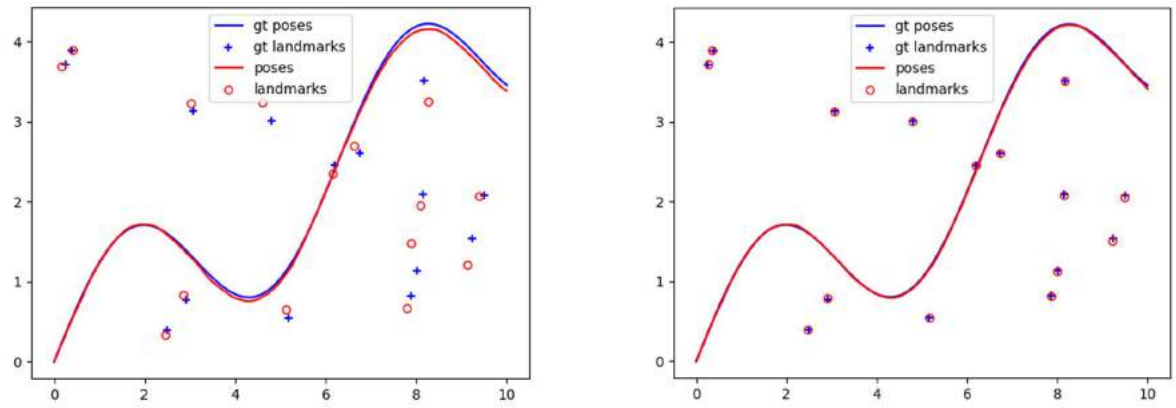


LU 2D non-linear estimation before and after optimization

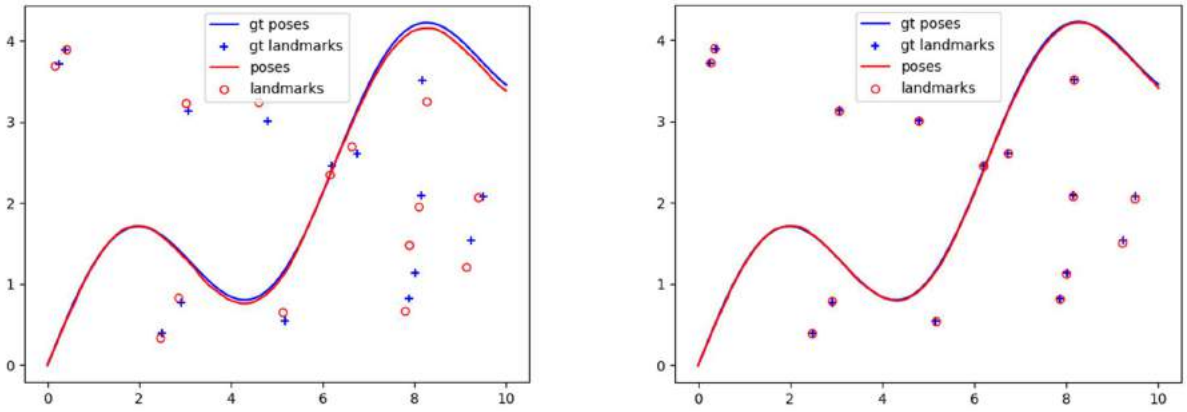




LU_COLAMD 2D non-linear estimation before and after optimization



QR 2D non-linear estimation before and after optimization



QR_COLAMD 2D non-linear estimation before and after optimization