Assignment 3

Andrew ID: sajaved

Name: Syed Ashar Javed

Q1

Part a

Assuming the state vector to be in the order $[r_x^{t-1}, r_y^{t-1}, r_x^t, r_y^t]$ for odometry and $[r_x^t, r_y^t, l_x^t, l_y^t]$ for landmark measurements, the measurement model and their Jacobians for the linear case is as follows:

$$
\begin{aligned}
h_0 &= [r_x^t - r_x^{t-1}, r_y^t - r_y^{t-1}] \\
h_l &= [l_x^t - r_x^t, l_y^t - r_y^t] \\
H_0 &= \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \\
H_1 &= \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}
\end{aligned}
$$

The above expression is given for any robot pose r and landmark location l at a given time t.

Part c

The economy version can be used as it does not return the full square matrix and instead returns only a rectangular matrix of size $m \times n$. For cases where the $m >> n$, it results in a lot of savings in space and time. This is possible because when solving a set of equations in x, the remaining columns of the Q matrix is independent of x and does not affect the result. If we're trying to minimize $||Ax - b||^2$, the solution is given by:

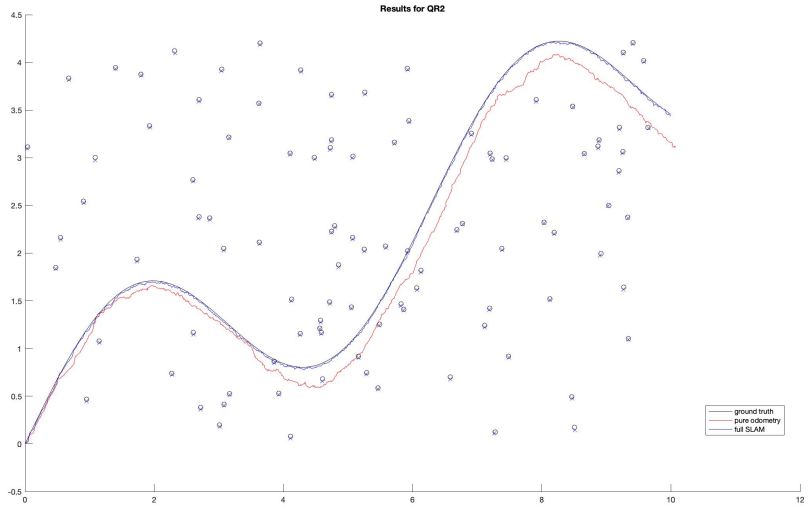
$$x^* = R_1^{-1}(Q_1^T b)$$

Thus, only the rectangular matrix Q_1 is needed.

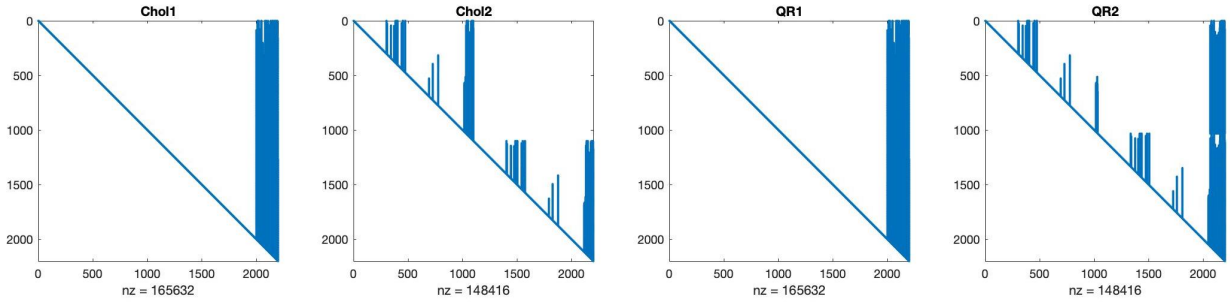
Part d

(i)

The results from the *2D_linear dataset is given below:*



Linear SLAM output trajectory and map



Sparse matrix comparison

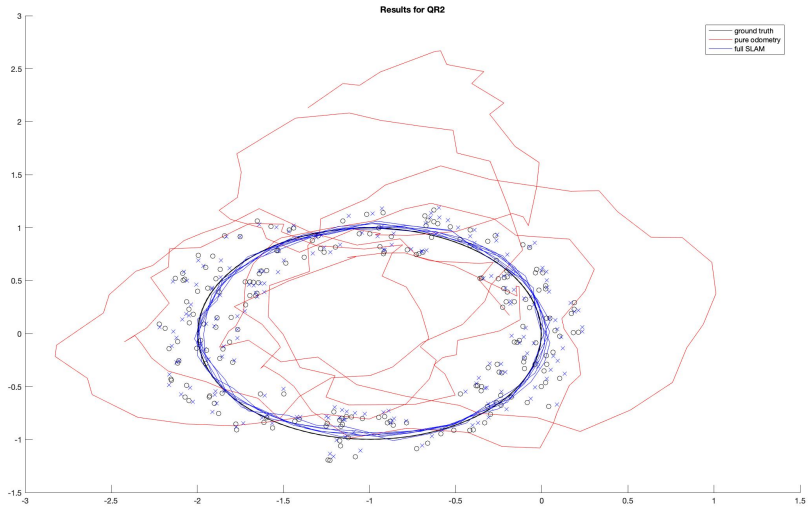
```
Command Window
  Timing Results
  Pinv:  3.280940e+00 sec
  Chol1: 2.657898e-01 sec
  Chol2: 1.569792e-01 sec
  QR1:   3.101233e-01 sec
  QR2:   2.215298e-01 sec

  Pinv solution is correct!
  Chol1 solution is correct!
  Chol2 solution is correct!
  QR1 solution is correct!
  QR2 solution is correct!

  rmse_odom =

      0.1887


  rmse_slam =

      0.0191

fx >>
```

Timing comparison

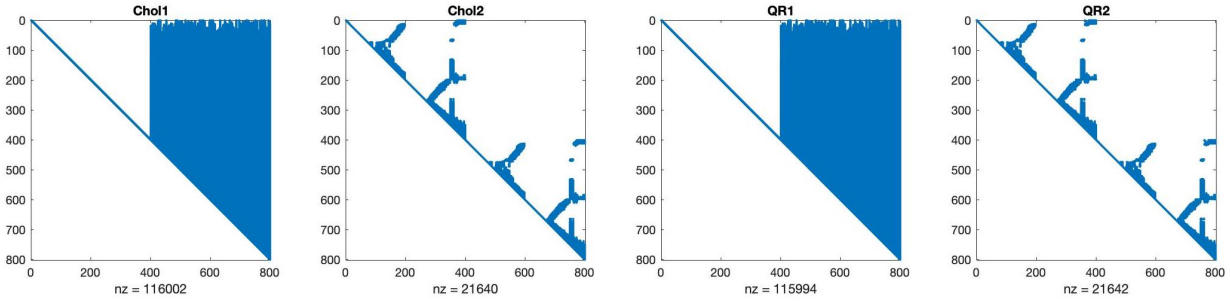
The order in which efficiency decreases is Chol2 > QR2 > Chol1 > QR1 > Pinv. This is expected as Cholesky is faster than QR in general. Also each of them is faster with the fill-in re-ordering. Finally, taking naive inverse is the least efficient with such a large matrix.

(ii)

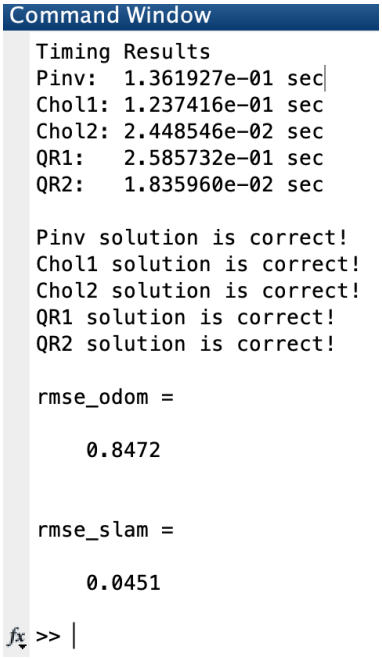
The results from the 2D_linear_loop dataset is given below:



Linear loop SLAM output trajectory and map



Sparse matrix comparison



Timing comparison

The order in which efficiency decreases is QR2 > Chol2 > Chol1 > Pinv > QR1. This is different from the previous dataset. Cholesky with fill-in does worse than QR with fill-in, however it's the same as before if no fill-in is used. This can be attributed to either the difference in the matrix size or the computation difference arising from doing $A^T A$ in Cholesky. Similar to the last dataset, the re-ordering helps, but due to the small size of the matrix, inverse does better this time.

Q2

Part b

The non-linear measurement model is given as:

$$\theta = atan2(l_y - r_y, l_x - r_x)$$
$$d = ((l_y - r_y)^2 + (l_x - r_x)^2)^{\frac{1}{2}}$$

The measurement Jacobian will be the follows:

$$\frac{\partial h}{\partial X} = \begin{bmatrix} \frac{\partial \theta}{\partial r_x} & \frac{\partial \theta}{\partial r_y} & \frac{\partial \theta}{\partial l_x} & \frac{\partial \theta}{\partial l_y} \\ \frac{\partial d}{\partial r_x} & \frac{\partial d}{\partial r_y} & \frac{\partial d}{\partial l_x} & \frac{\partial d}{\partial l_y} \end{bmatrix}$$

where X is the state vector and h is the measurement function. The jacobian fater taking the deriavtive will be the following:

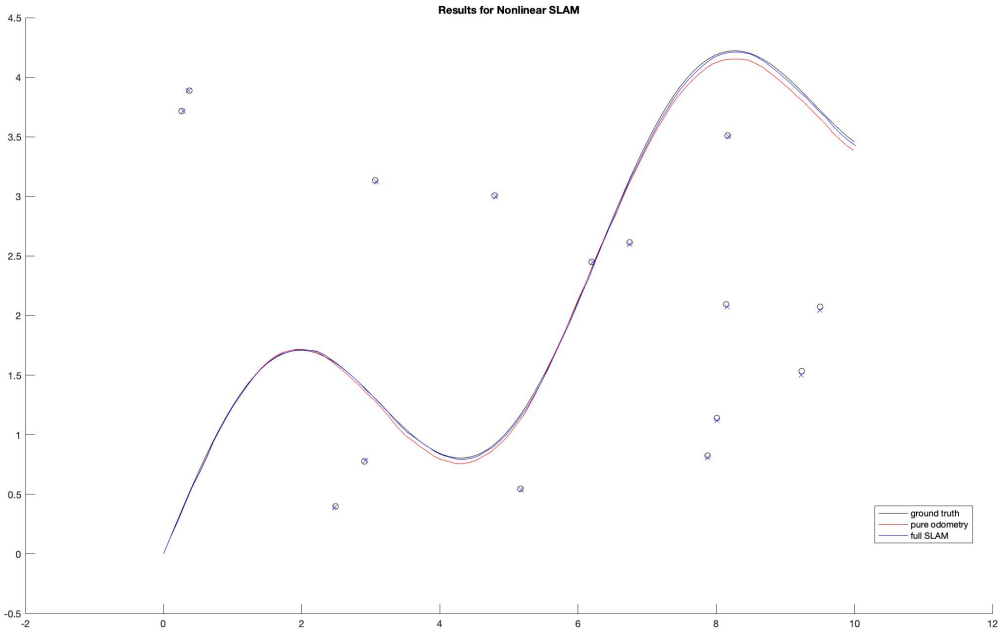
$$\frac{\partial h}{\partial X} = \begin{bmatrix} \frac{y}{d^2} & \frac{-x}{d^2} & \frac{-y}{d^2} & \frac{x}{d^2} \\ \frac{-x}{d} & \frac{-y}{d} & \frac{x}{d} & \frac{y}{d} \end{bmatrix}$$

where the variables used are:

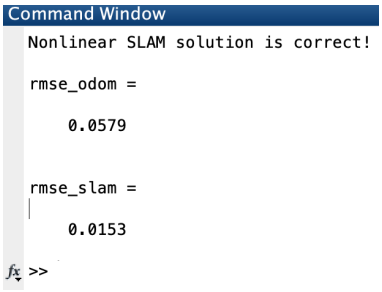
$$d = ((l_y - r_y)^2 + (l_x - r_x)^2)^{\frac{1}{2}}$$
$$x = l_x - r_x$$
$$y = l_y - r_y$$

Part d

The results from the nonlinear SLAM are as follows:



Nonlinear SLAM output trajectory and map



Nonlinear SLAM RMSE

Part e

The solution implemented in the nonlinear model is iteratively built instead of the full batch optimization with the full set of robot locations and landmark locations at once. We use Gauss-Newton method which works well with a good initialization. Also, the robot might be collecting data in an online way. Hence, it makes sense to refine the estimates as the data comes along so that the full batch optimization has initial estimates which are closer to the ground truth.