Adam Driscoll 3/30/2018
SLAM HW3

Introduction

In this assignment we were tasked with implementing two versions of the factor graph SLAM algorithm. In the first section, the algorithm is applied to linear data. In the second version it is expanded to work with nonlinear data. Different optimization techniques are explored and evaluated as well.

Collaborators: Joe Phaneuf, David Evans

2D Linear SLAM

A. The odometry function can be determined by do the following:

$$r = \begin{bmatrix} r_x \\ r_y \end{bmatrix}, \qquad u = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}, \qquad r^{t-1} = \begin{bmatrix} r_x^{t-1} \\ r_y^{t-1} \end{bmatrix}$$

$$r^t = u + r^{t-1}$$

$$h_o = u = r^t - r^{t-1} = \begin{bmatrix} r_x^t - r_x^{t-1} \\ r_y^t - r_y^{t-1} \end{bmatrix}$$

To derive the Jacobian we take the partial derivative with respect to each of the terms:

$$H_o = \begin{bmatrix} \frac{\delta h_{ox}}{r_x^{t-1}} & \frac{\delta h_{ox}}{r_y^{t-1}} & \frac{\delta h_{ox}}{r_x^t} & \frac{\delta h_{ox}}{r_y^t} \\ \frac{\delta h_{oy}}{r_x^{t-1}} & \frac{\delta h_{oy}}{r_y^{t-1}} & \frac{\delta h_{oy}}{r_x^t} & \frac{\delta h_{oy}}{r_y^t} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

For the landmark measurement function:

$$m = \begin{bmatrix} m_x \\ m_y \end{bmatrix}, \qquad l = \begin{bmatrix} l_x \\ l_y \end{bmatrix}$$

$$l = m + r$$

$$h_m = m = l - r = \begin{bmatrix} l_x - r_x \\ l_y - r_y \end{bmatrix}$$

And for the measurement Jacobian:

$$H_m = \begin{bmatrix} \frac{\delta h_{mx}}{r_x} & \frac{\delta h_{mx}}{r_y} & \frac{\delta h_{mx}}{l_x} & \frac{\delta h_{mx}}{l_y} \\ \frac{\delta h_{my}}{r_x} & \frac{\delta h_{my}}{r_y} & \frac{\delta h_{my}}{l_x} & \frac{\delta h_{my}}{l_y} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

C. iii.) It is possible to utilize the “economy size” version of the qr() function as the full A matrix is over determined, meaning it has row > cols.

D. i.)

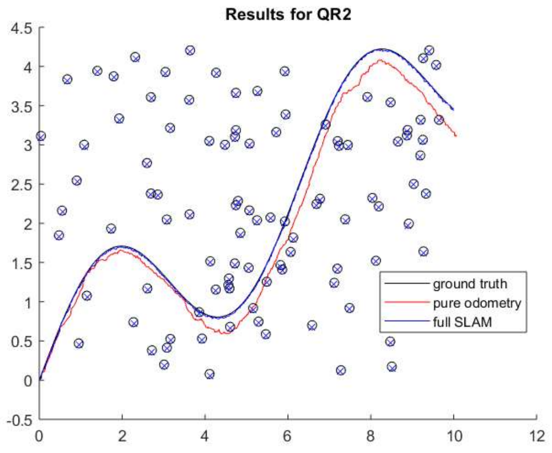

Figure 1: Map Result for Linear 2D SLAM

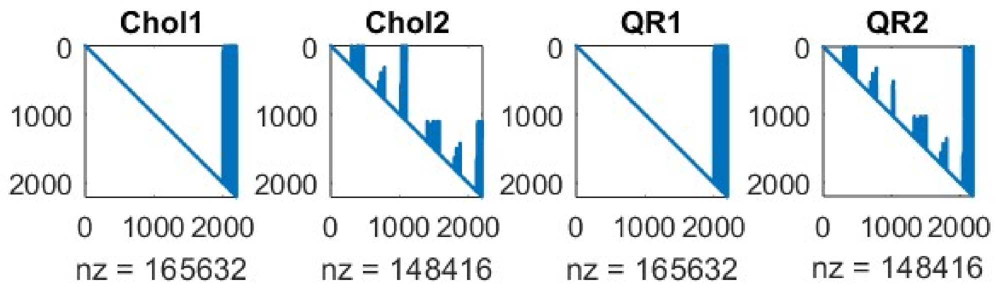

Figure 2: Matrix Results for Linear 2D SLAM

Timings for Linear 2D SLAM	
QR2	1.319814e-01 sec
Chol2	1.390131e-01 sec
QR2	2.416307e-01 sec
Chol1	2.482342e-01 sec
Pinv	2.224639e+00 sec

Both versions of QR and Cholesky operate in with the complexity of O(mn^2) which is why QR1 is comparable to Chol1 and QR2 is comparable to Chol2. The 2nd version of each is faster than the original version because the reduced fill-in techniques used increase the sparsity of the respective matrixes allowing for higher efficiency in factorization. Pinv is the slowest because it has complexity of O(n^3).

ii.)

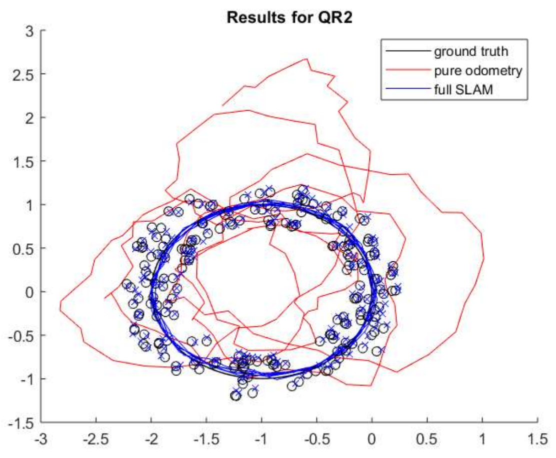

Figure 3: Map Result for Linear 2D SLAM with Loop

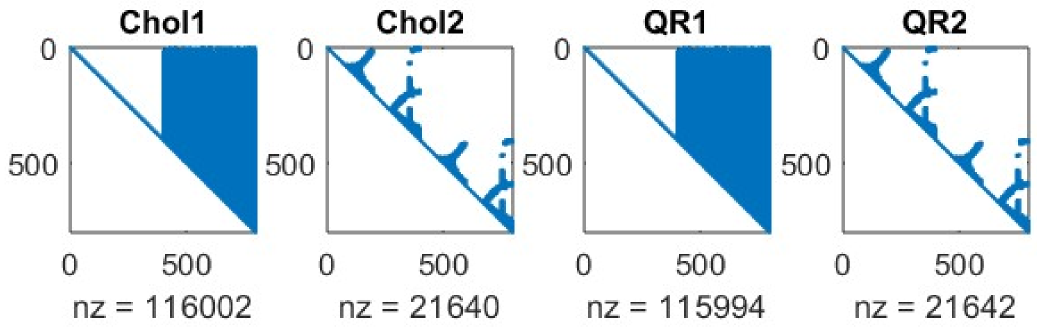

Figure 4: Matrix Results for Linear 2D SLAM with Loop

Timing Results	
QR2	7.566219e-03 sec
Chol2	2.228305e-02 sec
Pinv	8.502041e-02 sec
Chol1	1.113201e-01 sec
QR1	1.398858e-01 sec

It is not in the same order. QR2 and Chol2 remain the fastest because they are very sparse. Pinv overtook QR1 and Chol1 because those two matrixes are very dense resulting in much lower efficiency of computation.

Adam Driscoll 3/30/2018
SLAM HW3

2D Nonlinear SLAM

B. To get the Jacobian for the nonlinear measurement function we perform the same operations as in part 1:

$$H_m = \begin{bmatrix} \frac{\delta h_{mx}}{r_x} & \frac{\delta h_{mx}}{r_y} & \frac{\delta h_{mx}}{l_x} & \frac{\delta h_{mx}}{l_y} \\ \frac{\delta h_{my}}{r_x} & \frac{\delta h_{my}}{r_y} & \frac{\delta h_{my}}{l_x} & \frac{\delta h_{my}}{l_y} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{l_y - r_y}{d} & -\frac{l_x - r_x}{d} & -\frac{l_y - r_y}{d} & \frac{l_x - r_x}{d} \\ -\frac{l_x - r_x}{\sqrt{d}} & -\frac{l_y - r_y}{\sqrt{d}} & \frac{l_x - r_x}{\sqrt{d}} & \frac{(l_y - r_y)}{\sqrt{d}} \end{bmatrix}$$

$$d = (l_x - r_x)^2 + \left(l_y - r_y\right)^2$$

D.

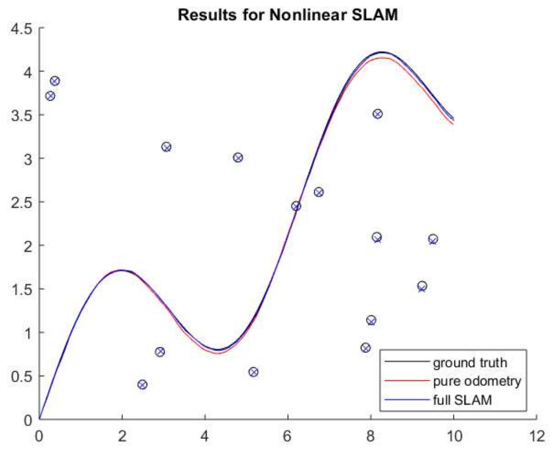

Figure 5: Map Result for Nonlinear 2D SLAM

Error Values	
RMSE Odom	0.0579
RMSE SLAM	0.0144

E. The nonlinear algorithm needs to be solved iteratively because we are only making a guess at the start and we don't know how good that it will be. We have to linearize around that guess to determine whether it was correct and then make a small adjustment based on how much error there was in our guess.