Solved Example: 1D SLAM (weighted LS) & Illustrating Sparsity in SLAM

Solved Example: 1D SLAM — Weighted Least Squares

Question:

A robot navigates in 1D environment and closes loop by returning to starting point. Our measurements or observed values are given by:

Quantity	Ground Truth	Observed / Measured
$\mathbf{u}_0$	1.0	1.1
$\mathbf{u}_1$	1.0	1.0
$\mathbf{u}_2$	1.0	1.1
$\mathbf{u}_3$	-3.0	-2.7
$\mathbf{u}_{0,4}$	0.0	0.0

$\mathbf{u_0} \rightarrow \mathbf{u_3}$ is obtained through odometry control inputs from an odometer/IMU.
$\mathbf{u_{0,4}}$ is obtained from a loop closure method, say Bag of Visual Words.

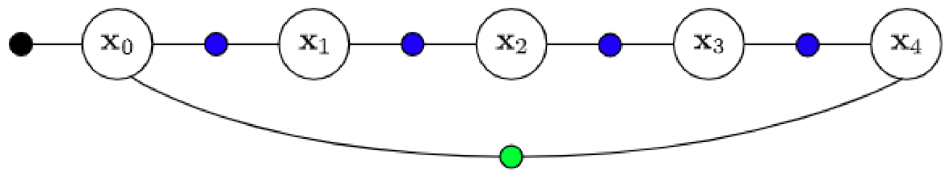


Image just for illustrative purposes and isn't representative of metric locations.

Note that in a SLAM setting, we wouldn't have the ground truth. It's been mentioned above so that we can compare and know if we are improving after we apply our algorithm.

Above, there are 4 odometry constraints of the form $\|f(\mathbf{x}_i, \mathbf{u}_i) - \mathbf{x}_{i+1}\|^2_{\mathbf{\Sigma}_i}$ and one loop closure of the form $\|f(\mathbf{x}_0, \mathbf{u}_{0,4}) - \mathbf{x}_4\|^2_{\mathbf{\Lambda}_{0,4}}$. Recollect here. Standard deviations for both are initialized as 0.1. ($\boldsymbol{\sigma}_i = 0.01 \text{ and } \boldsymbol{\lambda}_{0,4} = 0.01$)

Now the question is to get a better estimate of the robot states through optimization.

Answer:

Next robot state is given by (A simple motion model given by f) :

$$\mathbf{x}_{i+1} = f(\mathbf{x}_i, \mathbf{u}_i) = \mathbf{x}_i + \mathbf{u}_i$$

Robot states according to odometry alone:

Quantity	Ground Truth	According to Odometry
x_0	0.0	0.0
x_1	1.0	1.1
x_2	2.0	2.1
x_3	3.0	3.2
x_4	0.0	0.5

According to odometry, $\mathbf{x}_4 = 0.5$, but according to the loop closure constraint $\mathbf{x}_4 = \mathbf{x}_0 = 0.0$ (In the below residual, $\mathbf{x}_4 = 0.5$ but $\mathbf{x}_0 + \mathbf{u}_{0,4} = 0$.)

Note in the above table that according to loop closure constraint, $\mathbf{x}_4 = \mathbf{x}_0 = 0.0$. Using this "additional" measurement, we want to "correct" our robot state which has gone wrong because odometry drifted eventually (which happens in all practical scenarios).

Our objective function is

$$\mathbf{F}(\mathbf{x}) = \mathbf{f}(\mathbf{x})^\top \mathbf{\Omega} \mathbf{f}(\mathbf{x})$$

Recall:

$$\underset{X}{argmin} \underbrace{\sum_i \|f(x_i, u_i) - x_{i+1}\|^2_{\Sigma_i}}_{Odometry\ Constraints} + \underbrace{\sum_{ij} \|f(x_i, u_{ij}) - x_j\|^2_{\Lambda_{ij}}}_{Loop\ Closure\ Constraints}$$

$$\|\mathbf{x}_i - \mathbf{y}_i\|^2_{\mathbf{\Sigma}_i} = (\mathbf{x}_i - \mathbf{y}_i)^\top \mathbf{\Sigma}_i^{-1} (\mathbf{x}_i - \mathbf{y}_i)$$

So,

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f(\mathbf{x}_0, \mathbf{u}_0) - \mathbf{x}_1 \\ f(\mathbf{x}_1, \mathbf{u}_1) - \mathbf{x}_2 \\ f(\mathbf{x}_2, \mathbf{u}_2) - \mathbf{x}_3 \\ f(\mathbf{x}_3, \mathbf{u}_3) - \mathbf{x}_4 \\ f(\mathbf{x}_0, \mathbf{u}_{0,4}) - \mathbf{x}_4 \\ \mathbf{x}_0 - \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_0 + \mathbf{u}_0 - \mathbf{x}_1 \\ \mathbf{x}_1 + \mathbf{u}_1 - \mathbf{x}_2 \\ \mathbf{x}_2 + \mathbf{u}_2 - \mathbf{x}_3 \\ \mathbf{x}_3 + \mathbf{u}_3 - \mathbf{x}_4 \\ \mathbf{x}_0 + \mathbf{u}_{0,4} - \mathbf{x}_4 \\ \mathbf{x}_0 - 0 \end{pmatrix}$$

The last line is a "prior" constraint that anchors the first pose $\mathbf{x}_0$ at the origin with a very less covariance 0.001.

$$\mathbf{f}(\mathbf{x_O}) = (0 \quad 0 \quad 0 \quad 0 \quad -0.5 \quad 0)^T$$

$\mathbf{f}(\mathbf{x_O})$ is the initialization of $\mathbf{f}(\mathbf{x})$

The information matrix encodes the uncertainty of each edge. The "bigger" Ω is, the more the edge matters in the optimization or the more confident we are about the pose. As of now, we have more confidence on the pose $\mathbf{x_0}$. You can think of it as the "inverse" effect of variance values.

The information matrix is then given by:

$$\mathbf{\Omega} = \begin{pmatrix} \mathbf{\Sigma}_0 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{\Sigma}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{\Sigma}_2 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Sigma}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Lambda}_{0,4} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{\Pi} \end{pmatrix}^{-1} = \begin{pmatrix} 0.01 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.001 \end{pmatrix}^{-1}$$

$$= \begin{pmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1000 \end{pmatrix}$$

Recall Gauss Newton:

$$\mathbf{J}^\top \mathbf{\Omega} \mathbf{J} \Delta \mathbf{x} = -\mathbf{J}^\top \mathbf{\Omega}^\top \mathbf{f}(\mathbf{x})$$

$$\mathbf{f}(\mathbf{x}) = \begin{pmatrix} f(\mathbf{x}_0, \mathbf{u}_0) - \mathbf{x}_1 \\ f(\mathbf{x}_1, \mathbf{u}_1) - \mathbf{x}_2 \\ f(\mathbf{x}_2, \mathbf{u}_2) - \mathbf{x}_3 \\ f(\mathbf{x}_3, \mathbf{u}_3) - \mathbf{x}_4 \\ f(\mathbf{x}_0, \mathbf{u}_{0,4}) - \mathbf{x}_4 \\ \mathbf{x}_0 - \mathbf{0} \end{pmatrix} = \begin{pmatrix} \mathbf{x}_0 + \mathbf{u}_0 - \mathbf{x}_1 \\ \mathbf{x}_1 + \mathbf{u}_1 - \mathbf{x}_2 \\ \mathbf{x}_2 + \mathbf{u}_2 - \mathbf{x}_3 \\ \mathbf{x}_3 + \mathbf{u}_3 - \mathbf{x}_4 \\ \mathbf{x}_0 + \mathbf{u}_{0,4} - \mathbf{x}_4 \\ \mathbf{x}_0 - 0 \end{pmatrix}$$

$$\mathbf{J} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} = \left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}_0} \frac{\partial \mathbf{f}}{\partial \mathbf{x}_1} \cdots \frac{\partial \mathbf{f}}{\partial \mathbf{x}_4} \right) = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 1 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

💡 ***Do you realize now why we typically have an initial estimate in SLAM/Vision problems? And hence how that helps in optimization? (You'll realize even more after Computer Vision is taught)***

$$\mathbf{f}(\mathbf{x}_O) = (0 \quad 0 \quad 0 \quad 0 \quad -0.5 \quad 0)^T$$

$$\mathbf{H}\Delta\mathbf{x} = -\mathbf{b}$$

$$\mathbf{H} = \mathbf{J}^\top \mathbf{\Omega} \mathbf{J} = \begin{pmatrix} 1200 & -100 & 0 & 0 & -100 \\ -100 & 200 & -100 & 0 & 0 \\ 0 & -100 & 200 & -100 & 0 \\ 0 & 0 & -100 & 300 & -100 \\ -100 & 0 & 0 & -100 & 200 \end{pmatrix}$$

$$\mathbf{b} = \mathbf{J}^\top \mathbf{\Omega}^\top \mathbf{f}(\mathbf{x}_O) = \begin{pmatrix} -50 & 0 & 0 & 0 & 50 \end{pmatrix}^\top$$

$$\Delta\mathbf{x} = \begin{pmatrix} 0 & -0.1 & -0.2 & -0.3 & -0.4 \end{pmatrix}^\mathrm{T}$$

$$\mathbf{x}_I = \mathbf{x}_O + \Delta\mathbf{x} = \begin{pmatrix} 0 & 1.0 & 1.9 & 2.9 & 0.1 \end{pmatrix}^T$$

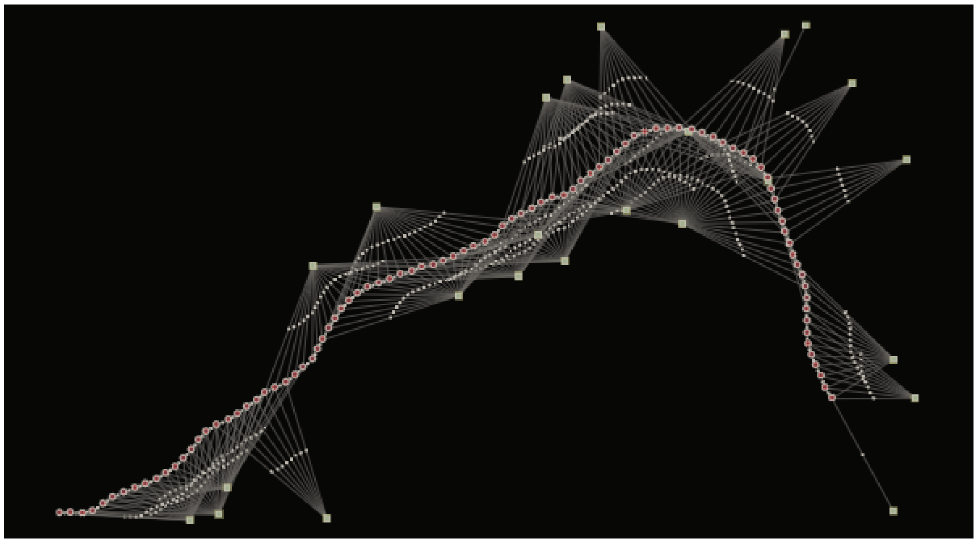
Quantity	Ground Truth	According to Odometry
x_0	0.0	0.0
x_1	1.0	1.1
x_2	2.0	2.1
x_3	3.0	3.2
x_4	0.0	0.5

Quantity	Ground Truth	Observed / Measured
$\mathbf{u}_0$	1.0	1.1
$\mathbf{u}_1$	1.0	1.0
$\mathbf{u}_2$	1.0	1.1
$\mathbf{u}_3$	-3.0	-2.7
$\mathbf{u}_{0,4}$	0.0	0.0

> **Go back to error function for 2D/3D SLAM.**

💡 `Warming up for Project 1:`
1. Code the above from scratch. Only allowed to use `numpy` library.
2. Do one more step and comment on your observations. What value did you get as $\mathbf{x}_2$? What about $\mathbf{x}_3$? What conclusion can you draw? Show your calculations clearly.
3. Can you tell the above "conclusion" directly without even solving for $\mathbf{x}_2$? Go back to basics: Does this have anything to do with *"linear/non-linear"* least squares?

Sparsity in SLAM



Source: Factor Graphs for Robot Perception, Daellart and Kaess, Figure 2.1

The sparsity can be appreciated directly from looking at the above pose graph. It is clear from the above figure that the graph is sparse, i.e., it is by no means a fully connected graph. The odometry chain linking the 100 unknown poses is a linear structure of 100 binary factors, instead of the possible 100^2 (binary) factors. In addition, with 20 landmarks we could have up to 2000 likelihood factors linking each landmark to each pose: the true number is closer to 400. And finally, there are no factors between landmarks at all. This reflects that we have not been given any information about their relative position. **This structure is typical of most SLAM problems.**

Let us understand this in detail by revisiting our solved example.

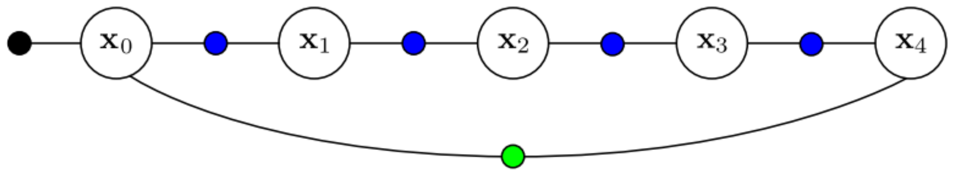
1. The normal equation can be using:

$$\mathbf{H}\Delta\mathbf{x} = -\mathbf{b}$$

$$\mathbf{H} = \mathbf{J}^{\top}\boldsymbol{\Omega}\mathbf{J} = \begin{pmatrix} 1200 & -100 & 0 & 0 & -100 \\ -100 & 200 & -100 & 0 & 0 \\ 0 & -100 & 200 & -100 & 0 \\ 0 & 0 & -100 & 300 & -100 \\ -100 & 0 & 0 & -100 & 200 \end{pmatrix}$$

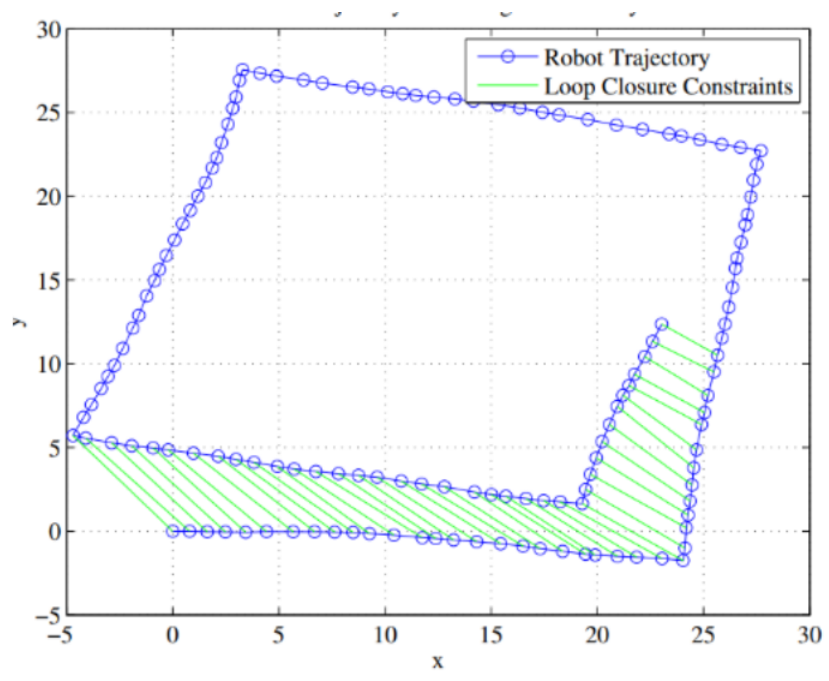
$$\mathbf{b} = \mathbf{J}^{\top}\boldsymbol{\Omega}^{\top}\mathbf{f}\left(\mathbf{x_0}\right) = \begin{pmatrix} -50 & 0 & 0 & 0 & 50 \end{pmatrix}^{\top}$$

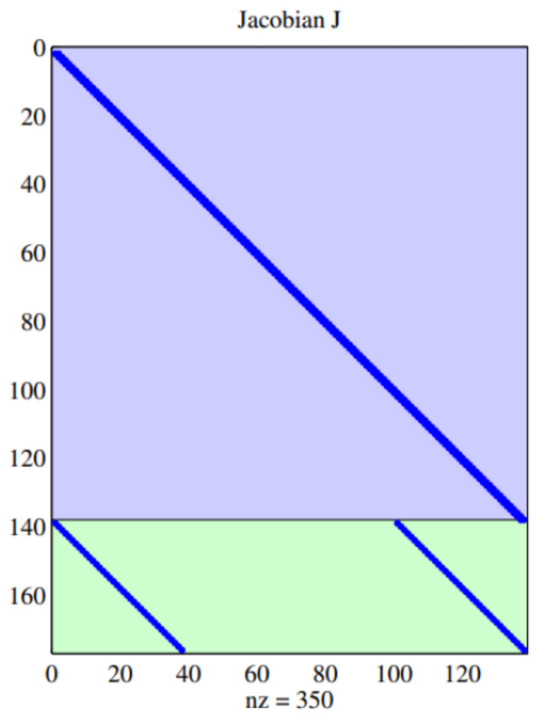
2. $\mathbf{H}$ is an adjacency matrix of the factor graph representation of the SLAM problem, and it has non-zero values only where node i and node j are connected.

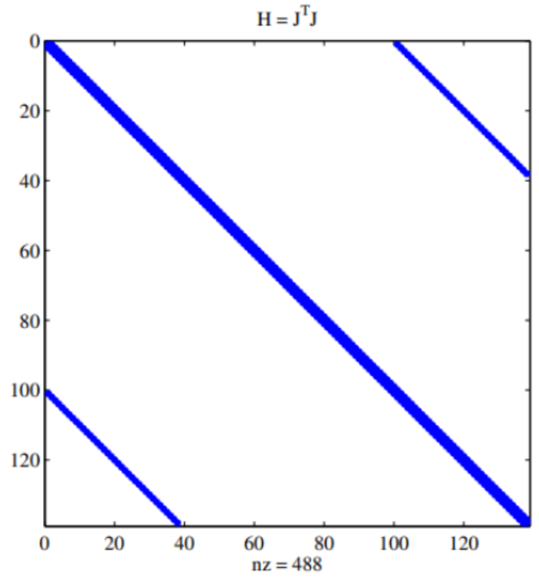


3. Most of the nodes in the factor graph representation are connected to two nodes, except the loop closure nodes.

4. The nonzero elements in the $\mathbf{J}$ and $\mathbf{H}$ are marked by dark blue. The diagonal entries correspond to odometry measurements while off-diagonal entries are from loop closure constraints.


Robot Trajectory
Loop Closure Constraints
x
y


Jacobian J
nz = 350


$H = J^T J$
nz = 488