Exploiting Sparsity

Representation:
Linked List? -> very compact, easy to add entries, but slow for important ops.
Compressed Sparse Row (CSR)

\[ [0, 0, 3, 4, 0, 0, 1] \]

\[ \Rightarrow \text{indices} = \{2, 3, 6\} \text{ sorted so indices is from left to right,} \]
\[ \text{values} = \{3, 4, 1\} \]

Consider dot product of vector above with:
\[ \text{indices} = \{1, 3, 4, 6\} \]
\[ \text{values} = \{2, 2, 2, 2\} \]

Answer? 10
Write pseudo-code,

Gaussian Elimination
Basic operation:
\[ s = a + \beta b \] (why?)

Can this be accelerated using CSR?

Remember: we wouldn't actually use G, E - but it's illustrative.
We've ignored pivots, for example!
Posterior Covariance

We've mostly focused on ML solutions, but we know the posterior of a least-squares problem is itself a Gaussian.

\[ M = \text{ML estimate.} \]

\[ \Sigma_x \]

To solve this system, we formulated a quadratic loss that looked like:

\[ (z(x) - z)^T \Sigma_z^{-1} (z(x) - z) \]

We approximated

\[ z(x) \approx \frac{z}{J_x^2} \Delta x + z(x_0) \]

so:

\[ (J_x \Delta x - r)^T \Sigma_z^{-1} (J_x \Delta x - r) \]

where \( r = z_{x_0} - z(x_0) \)

This system can be viewed as one enormous multi-variate Gaussian in the space of the \( z \)'s (not the \( x \)'s)

Let's manipulate this eqn so that it looks like a quadratic loss in terms of \( x \):

\[ \Delta x^T J^T \Sigma_z^{-1} J \Delta x - 2 \Delta x^T J^T \Sigma_z^{-1} r + \frac{\Delta M}{\text{some constant}} \]

\[ \Rightarrow (\Delta x - \frac{r}{J^T \Sigma_z^{-1} J})^T J^T \Sigma_z^{-1} J (\Delta x - \frac{r}{J^T \Sigma_z^{-1} J}) \]

some mess to complete the square.

\[ \Sigma_x^{-1} \]
\[ \Sigma_x^{\dagger} \text{ was staring at us the whole time!} \]

\[ (J^T \Sigma_x^{-1} J) \Delta x = J^T \Sigma_x^{-1} r \]

not just the "A" in our \( Ax = b \) problem,
but also \( \Sigma_x^{-1} \).

\[ \Sigma_x^{-1} \text{ is sparse, as we've discussed,} \]

what about \( \Sigma_x \)? \(< \text{Not sparse in general,} \>

Implication \( \Sigma_x \) takes \( O(n^2) \) storage, quickly becomes problematic.

When do we want \( \Sigma_x \)?
- data association, mostly. But an approximation is usually good enough.

Which landmarks should I test for data association? (Which features could be within sensor range: \( r^2 \)?)
How do we compare two SLAM algorithms? Let's ignore runtime & focus on quality... for now.

There is a correct solution. If both algorithms find the correct solution, they're both equally good... "optimal".

\[
x^* = \arg \min_x (Jx - r)^T \Sigma^{-1}_\epsilon (Jx - r)
\]

Our cost function, \( \mathcal{K}^2(x) \).

What if the methods aren't optimal?

Is \( x_a \) better than \( x_b \)? if \( \mathcal{K}^2(x_a) < \mathcal{K}^2(x_b) \)?

\[
\begin{array}{c}
\mathcal{K}^2(x) \\
\hline
x_a \quad x^* \quad x_b
\end{array}
\]

Is \( x_a \) better than \( x_b \)? Arguably not...

An imperfect SLAM solution is characterized in two ways:
- Its error in the \( \mathcal{K}^2 \) direction (\( \mathcal{K}^2 \))
- Its error in the \( x \) direction. (MSE)

Which error do you value more? Probably \( x \)!

Should MSE beeval'd w/ ground truth or numerical optimum? -can tip example.

(show CSW mops in which \( \mathcal{K}^2 \) is a bad predictor of quality)
Problems with MSE:

- it presupposes knowing the answer!
- useful for off-line (controlled evaluations), not online.

\( \chi^2 \)

- Do we know what \( \min_x \chi^2(x) \) is?
- also presupposes the answer.

Why would we want to know?

\( \chi^2(x_{row}) - \chi^2(x_{min}) \) could help us decide if we need more iterations.

Good News!

\( \chi^2 \) follows a distribution.
- parameterized by "degrees of freedom."

- Intuition: the more observations, the more \( \chi^2 \) error we expect.

Degrees of freedom:

\[ \text{# of observation egns - # state variables} \]

\[ \text{DOF} \]

- Useful Fact: \( E[\frac{\chi^2}{\text{DOF}}] = 1 \).

- Can use as a guide: when to iterate
- Warning: what happens if noise models are wrong?