

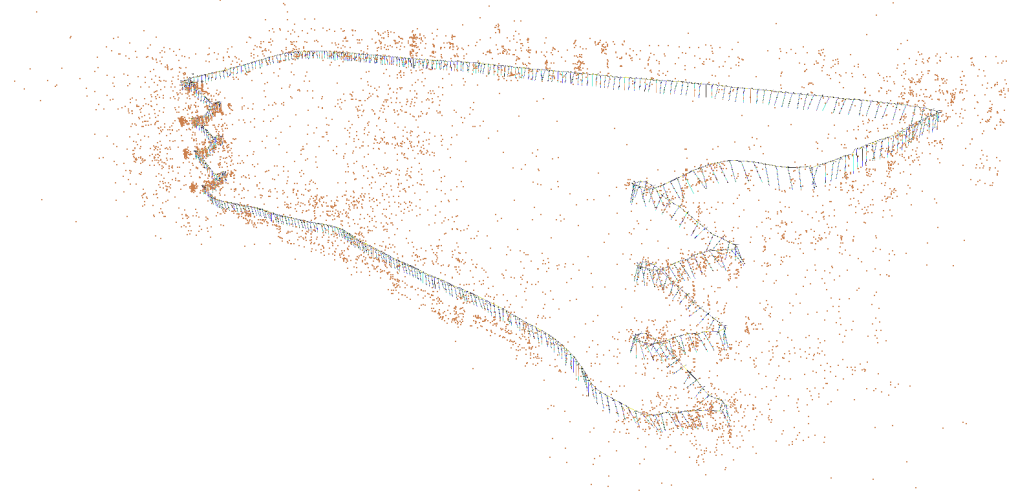
# Probabilistic Robotics Course

## Sparse Optimization

**Giorgio Grisetti**

`grisetti@diag.uniroma1.it`

Department of Computer, Control, and Management Engineering  
Sapienza University of Rome



# Large Problems

Many relevant estimation problems require estimating a large number of variables given a very large number of measurements.

Examples include

- Pose-graphs
- Pose-Landmark
- Calibration
- **<add your own>**
- **<combine the above problems>**

**The larger the problem, the slower the solution**

# Large and Sparse Problems

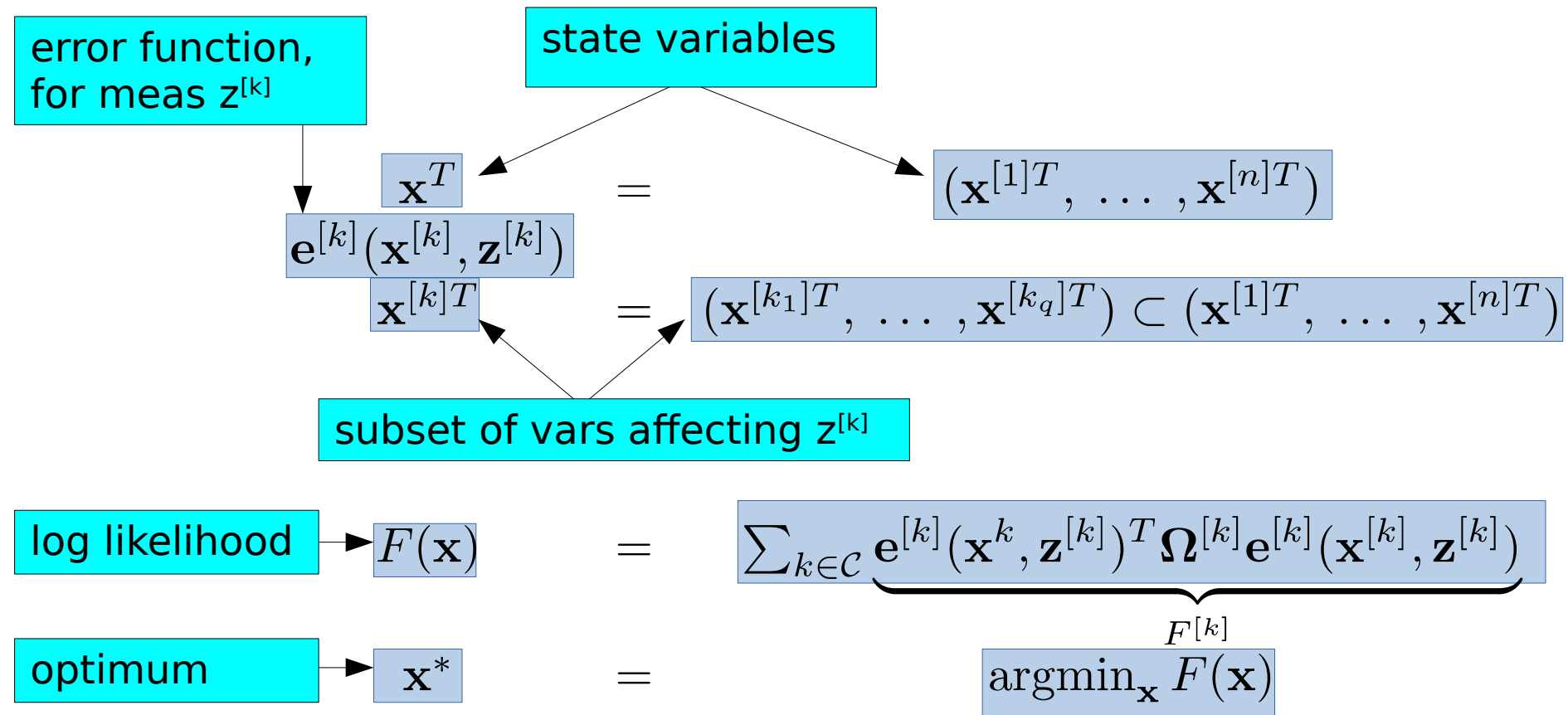
Despite the high number of variables, a single measurement is typically only affected by a subset of state variables

Examples:

- *Landmark measurement* determined by :
  - the observer position  $X_r^{[n]}$
  - the position of the landmark  $X_l^{[m]}$
- Pose-Pose measurement determined by
  - the observer position  $X_r^{[i]}$
  - the observed position  $X_r^{[j]}$

# Graphical Representation

Formally we can highlight the dependency of a measurement from a subset of state variables by the following notation

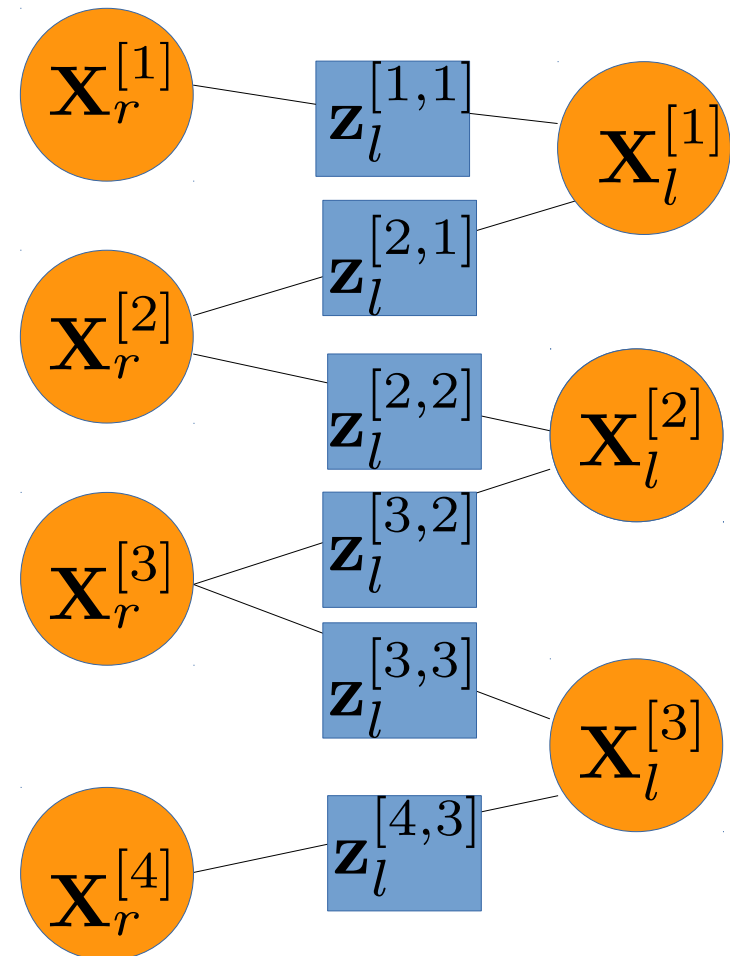


# Graphical Representation

We can represent the problem with a graph

- A node for each state variable
- A node for each measurement
- An edge between a variable and a measurement if they are dependant

Example: pose-landmark SLAM

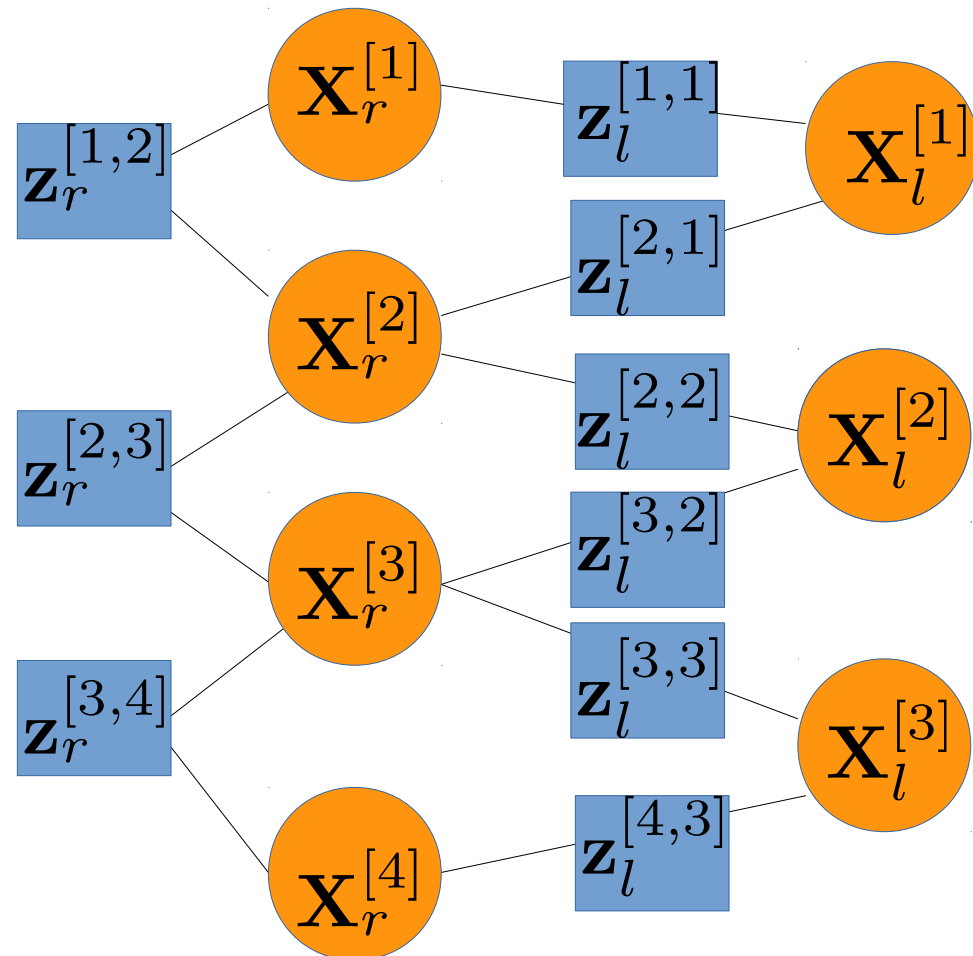


# Graphical Representation

We can represent the problem with a graph

- A node for each state variable
- A node for each measurement
- An edge between a variable and a measurement if they are dependant

Example: landmark+odometry



# Graphical Representation

This representation is known as a factor graph.

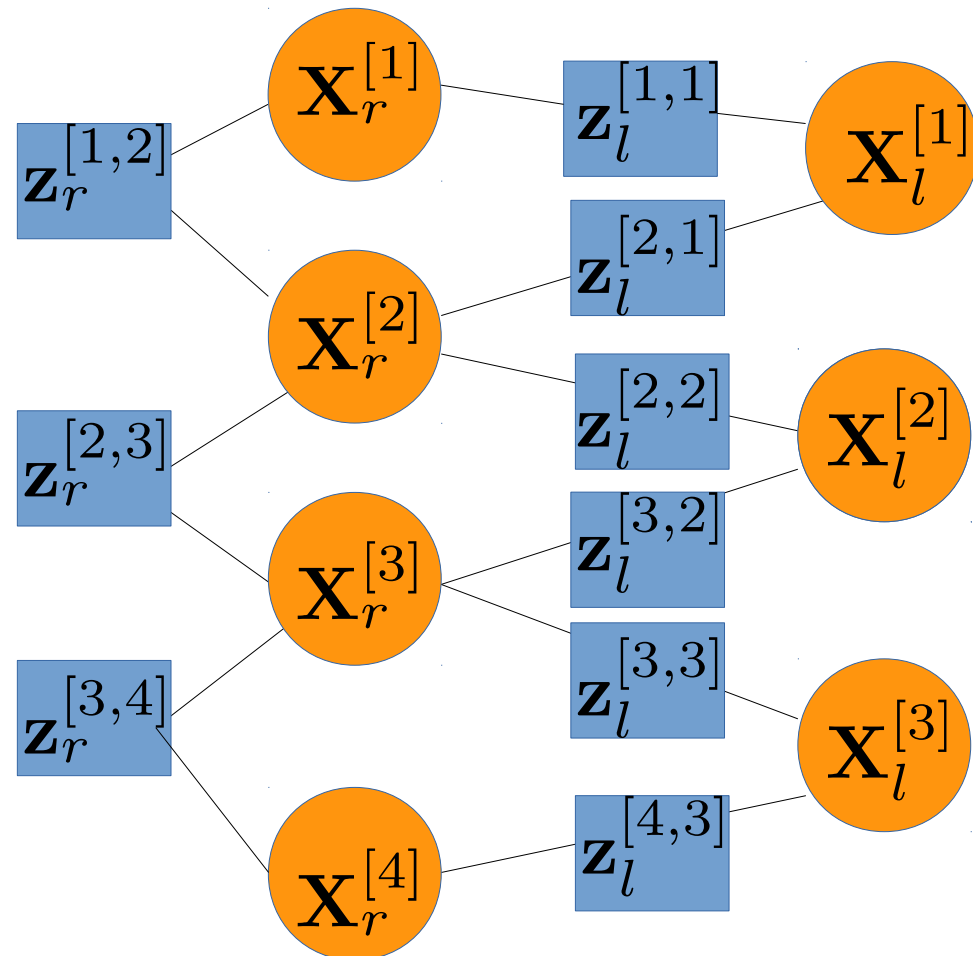
Can represent arbitrary PDF

In our case we restrict to Gaussians

It is bipartite:

- two variable nodes are connected only through a measurement
- factors can be seen as hyper-edges connecting multiple variables

Example: landmark+odometry



# Structure

The contribution to the K matrix for a measurement  $z^{[k]}$  measurement will affect only the state components in  $x^{[k]}$

$$\mathbf{J}^{[k]} = (\mathbf{0} \dots \mathbf{0} \mathbf{J}^{[k_1]} \dots \mathbf{J}^{[k_i]} \dots \mathbf{0} \dots \mathbf{J}^{[k_q]} \mathbf{0} \dots \mathbf{0})$$

$$\mathbf{H}^{[k]} = \begin{pmatrix} \ddots & & & & & & \\ & \mathbf{J}^{[k_1]T} \boldsymbol{\Omega}^{[k]} \mathbf{J}^{[k_1]} & \dots & \mathbf{J}^{[k_1]T} \boldsymbol{\Omega}^{[k]} \mathbf{J}^{[k_i]} & \dots & \mathbf{J}^{[k_1]T} \boldsymbol{\Omega}^{[k]} \mathbf{J}^{[k_q]} & \\ & \vdots & & \vdots & & \vdots & \\ & \mathbf{J}^{[k_i]T} \boldsymbol{\Omega}^{[k]} \mathbf{J}^{[k_1]} & \dots & \mathbf{J}^{[k_i]T} \boldsymbol{\Omega}^{[k]} \mathbf{J}^{[k_i]} & \dots & \mathbf{J}^{[k_i]T} \boldsymbol{\Omega}^{[k]} \mathbf{J}^{[k_q]} & \\ & \vdots & & \vdots & & \vdots & \\ & \mathbf{J}^{[k_q]T} \boldsymbol{\Omega}^{[k]} \mathbf{J}^{[k_1]} & \dots & \mathbf{J}^{[k_q]T} \boldsymbol{\Omega}^{[k]} \mathbf{J}^{[k_i]} & \dots & \mathbf{J}^{[k_q]T} \boldsymbol{\Omega}^{[k]} \mathbf{J}^{[k_q]} & \\ & & & & & & \ddots \end{pmatrix}$$

$$\mathbf{b}^{[k]T} = \begin{pmatrix} \vdots \\ \mathbf{J}^{[k_1]T} \boldsymbol{\Omega}^{[k]} \mathbf{e}^{[k]} \\ \vdots \\ \mathbf{J}^{[k_i]T} \boldsymbol{\Omega}^{[k]} \mathbf{e}^{[k]} \\ \vdots \\ \mathbf{J}^{[k_q]T} \boldsymbol{\Omega}^{[k]} \mathbf{e}^{[k]} \\ \vdots \end{pmatrix}$$

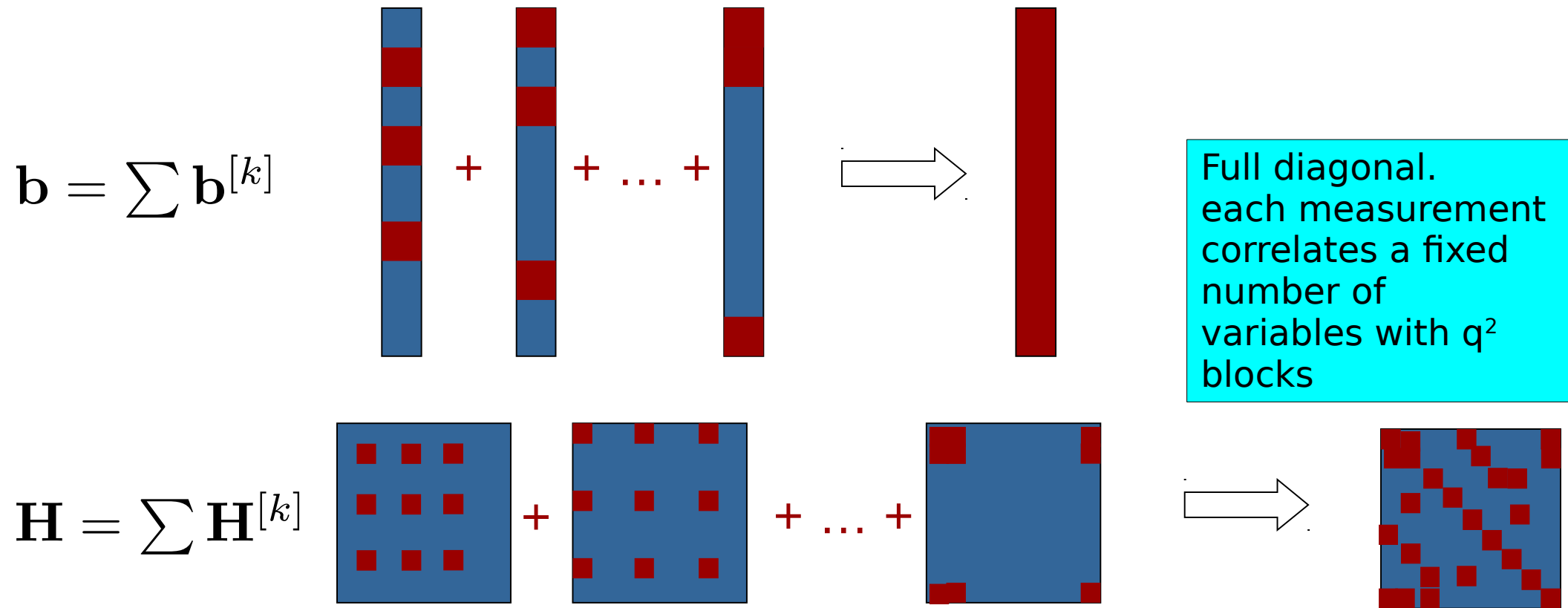


# Structure

Structure of **H**: location of the non zero elements

The structure of H depends only on the structure of the measurements

- preallocate before the iterations
- do not allocate memory for zero blocks



# Structure and Efficiency

The number of non-zero blocks in  $H$  depends on the number of measurements

Bounded by:

- number of poses
- perception range
- landmark density

In typical SLAM-related problems the number of measurements grows linearly with the length of the trajectory

$H$  has a *linear* number of non-zero blocks (it is mostly empty)

Use special techniques to solve sparse linear systems

# Solving Sparse Systems

A common solution to solve a symmetric positive definite system of equation is through Cholesky factorization

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

System we want to solve

$$\mathbf{H} = \mathbf{L}\mathbf{L}^T$$

Cholesky (L lower triangular)

$$\mathbf{L}\underbrace{\mathbf{L}^T\Delta\mathbf{x}}_y = -\mathbf{b}$$

$$\mathbf{L}y = -\mathbf{b}$$

Solve for y by backward substitution

$$\mathbf{L}^T\Delta\mathbf{x} = y$$

Solve for x by forward substitution

**Big Issue:** H sparse does not mean L sparse!!  
We lose the benefits of sparsity

# Permutations and Cholesky

## Permutation matrix

- Encodes a reordering of the variables
- The elements are either 0 or 1
- Exactly 1 non zero for each row
- Exactly 1 non zero for each column
- The inverse of a permutation is its transpose

If a matrix is sparse, there is a reordering of the variables that renders the Cholesky factor maximally sparse

Computing such an ordering is **NP complete**

Efficient heuristics solve the problem

# Permutations and Cholesky

Solve the linear system through a permutation

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

System we want to solve

$$\mathbf{P}\mathbf{H}\underbrace{\mathbf{P}^T\mathbf{P}}_{\mathbf{I}}\Delta\mathbf{x} = -\mathbf{P}\mathbf{b}$$

Apply a permutation

$$\underbrace{\mathbf{P}\mathbf{H}\mathbf{P}^T}_{\mathbf{H}'}\underbrace{\mathbf{P}\Delta\mathbf{x}}_{\Delta\mathbf{x}'} = -\underbrace{\mathbf{P}\mathbf{b}}_{\mathbf{b}'}$$

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

Solve the system under permutation

$$\mathbf{H}' = \mathbf{L}'\mathbf{L}'^T$$

Cholesky decomposition of  $\mathbf{H}'$  (sparse)

$$\mathbf{L}'\underbrace{\mathbf{L}'^T\Delta\mathbf{x}'}_y = -\mathbf{b}'$$

$$\mathbf{L}'y = -\mathbf{b}'$$

solve through forward/backward subst.

$$\mathbf{L}'^T\Delta\mathbf{x}' = y$$

$$\Delta\mathbf{x} = \mathbf{P}^T\Delta\mathbf{x}'$$

Recover  $\Delta\mathbf{x}$  applying inverse permutation

# Conclusions

We approached a complex multi-robot multi-landmark problem

- The measurement independence leads to a sparse structure of  $H$
- The structure of  $H$  does not change during the iterations
- It can be efficiently solved by using sparse methods
- Cholesky is not the only possible way (also other approaches such as QR factorization will do)
- Sparse methods rely on finding an ordering that keeps the triangular system sparse

# Tomorrow: Total Least Squares

Sparse Optimization with

- Pose-Pose (3D)
- Pose-Landmark (3D)
- Pose-Landmark (Projection: 2D)

Using a factor graph

# Pose-Landmark Constraint

- Here we represent the state as the pose of the robot in the world, i.e.  $\mathbf{X} : {}^W\mathbf{T}_R$
- Consequently, prediction and error functions become

$$\mathbf{h}^{[i,j]}(\mathbf{X}) = \mathbf{X}_r^{[i]-1} \mathbf{X}_l^{[j]} = \mathbf{R}_\theta^T \mathbf{X}_l^{[j]} + \mathbf{R}_\theta^T \mathbf{t}$$

$$\mathbf{e}^{[i,j]}(\mathbf{X}_r \boxplus \Delta \mathbf{x}_r, \mathbf{X}_l + \Delta \mathbf{x}_l) = [v2t(\Delta \mathbf{x}_r) \mathbf{X}_r]^{-1} (\mathbf{X}_l + \Delta \mathbf{x}_l) - \mathbf{z}_j$$

$$\mathbf{J}_R = \left. \frac{\partial [v2t(\Delta \mathbf{x}_r) \mathbf{X}_r]^{-1} (\mathbf{X}_l + \Delta \mathbf{x}_l) - \mathbf{z}_j}{\partial \Delta \mathbf{x}_r} \right|_{\Delta \mathbf{x}_r=0, \Delta \mathbf{x}_l=0}$$

$$\mathbf{J}_L = \left. \frac{\partial [v2t(\Delta \mathbf{x}_r) \mathbf{X}_r]^{-1} (\mathbf{X}_l + \Delta \mathbf{x}_l) - \mathbf{z}_j}{\partial \Delta \mathbf{x}_l} \right|_{\Delta \mathbf{x}_r=0, \Delta \mathbf{x}_l=0}$$



# Pose-Pose Constraint

- Here we represent the state as the pose of the robot in the world, i.e.  $\mathbf{X} : {}^W\mathbf{T}_R$
- Consequently, prediction and error functions become

$$h^{[i,j]}(\mathbf{X}) = \hat{\mathbf{Z}}^{[i,j]} = \mathbf{X}_r^{[i]-1} \mathbf{X}_r^{[j]}$$

$$e^{[i,j]}(\mathbf{X}) = t2v \left[ \mathbf{Z}^{[i,j]-1} \mathbf{X}_r^{[i]-1} \mathbf{X}_r^{[j]} \right]$$

$$e^{[i,j]}(\mathbf{X}_r \boxplus \Delta \mathbf{x}_r) = t2v \left[ \mathbf{Z}^{[i,j]-1} v2t(\Delta \mathbf{x}_r^i)^{-1} \hat{\mathbf{Z}}^{[i,j]} v2t(\Delta \mathbf{x}_r^j) \right]$$