Siddharth Choudhary

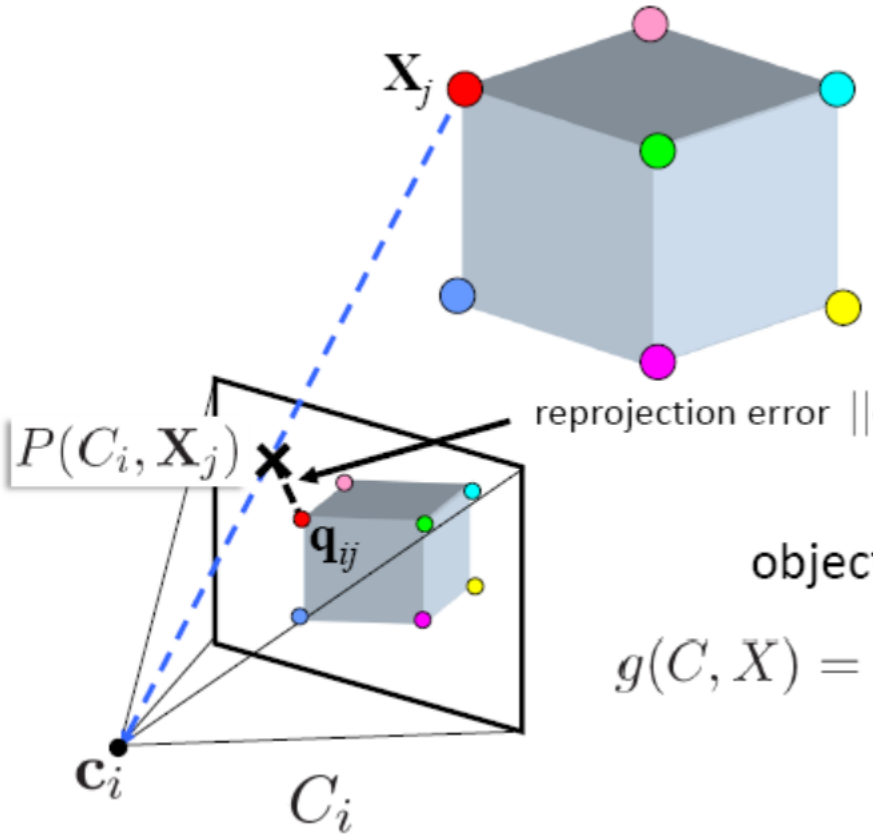
Bundle Adjustment : A Tutorial

What is Bundle Adjustment ?

- Refines a visual reconstruction to produce jointly optimal 3D structure and viewing parameters

- *'bundle'* refers to the bundle of light rays leaving each 3D feature and converging on each camera center.

Re Projection Error



reprojection error $||\mathbf{q}_{ij} - P(C_i, \mathbf{X}_j)||$

objective function:

$$g(C, X) = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} ||\mathbf{q}_{ij} - P(C_i, \mathbf{X}_j)||^2$$

indicator variable:
1 if point *j* is visible in camera *i*
0 otherwise

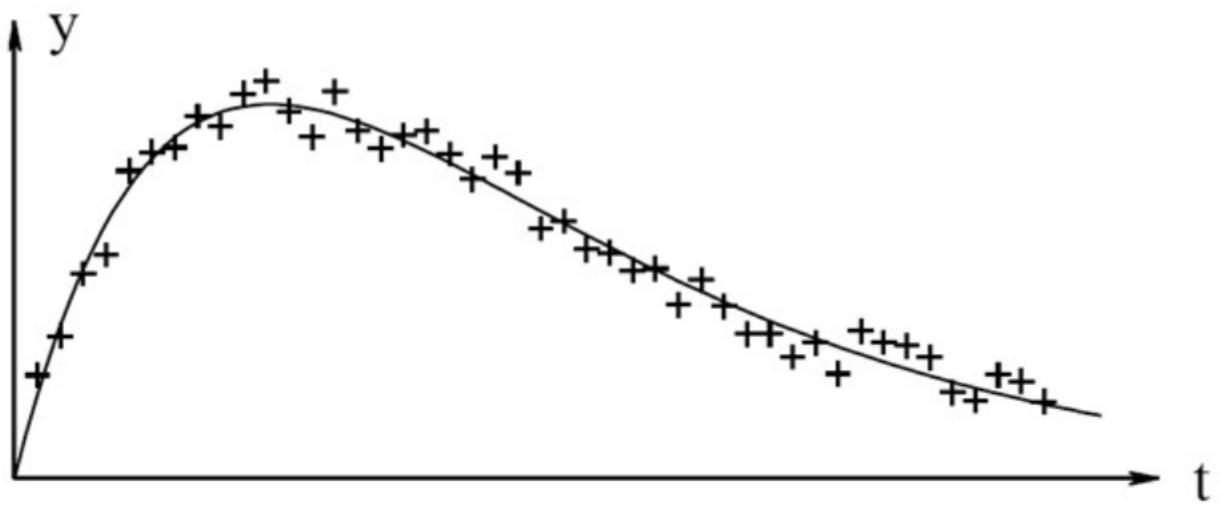
Some Notations

- Structure and Cameras being parameterized by a single large vector '**x**'
- Small displacement in **x** represented by $\partial \boldsymbol{x}$
- Observations denoted by '$\underline{z}$'
- Predicted values at parameter value **x**, denoted by z = z(x)
- Residual prediction error, $\Delta z(\boldsymbol{x}) = \underline{z} - z(\boldsymbol{x})$
- Cost Function = $f(x) = f(\,predz\,(x))$

Objective Function

- Minimization of weighted sum of squared error (SSE) cost function:

$$f(x) \equiv \frac{1}{2} \sum_i \Delta z_i(x)^\top W_i \, \Delta z_i(x) \,, \qquad \Delta z_i(x) \equiv \underline{z}_i - z_i(x)$$



Some Facts about Non linear least squares

- Least-squares fitting is a maximum likelihood estimation of the fitted parameters if the measurement errors are independent and normally distributed with constant standard deviation

- The probability distribution of the sum of a very large number of very small random deviations almost always converges to a normal distribution.

Disadvantage of Non Linear Least Squares

- It is highly sensitive to outliers, because the Gaussian has extremely small tails compared to most real measurement error distribution.

(It is the reason of using Hierarchical SFM)

Gaussian Tail problem and its effects is addressed in the paper ‘ Pushing the envelope of modern bundle adjustment techniques, CVPR 2010’

Optimization Techniques

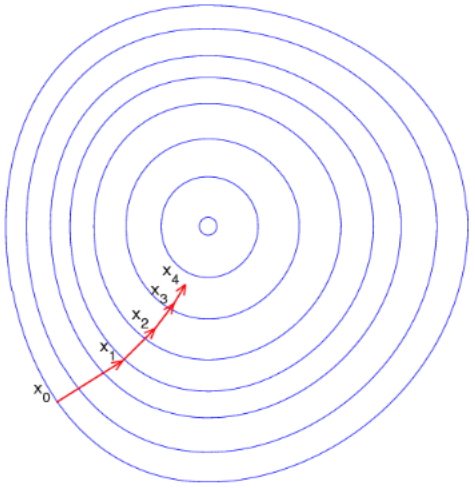
- Gradient Descent Method
- Newton-Rhapson Method
- Gauss – Newton Method
- Levenberg – Marquardt Method

Gradient Descent Method

- A first-order optimization algorithm.
- To find a local minimum of a function using gradient descent, one takes steps proportional to the *negative* of the gradient of the function at the current point.

While k<kmax

$$x_k = x_{k-1} + \lambda \nabla f(x_{k-1})$$



Gradient Descent Method

- It is robust when x is far from optimum but has poor final convergence

(this fact is used in designing the LM iteration)

Newton – Rhapson Method

- It is a second order optimization method
- Newton's method can often converge remarkably quickly, especially if the

$$f(x+\delta x) \approx f(x) + g^{\top}\delta x + \frac{1}{2}\delta x^{\top} H\,\delta x$$

quadratic local model

$$g \equiv \frac{df}{dx}(x)$$

gradient vector

$$H \equiv \frac{d^2 f}{dx^2}(x)$$

Hessian matrix

the desired root.

$$\frac{df}{dx}(x+\delta x) \approx H\,\delta x + g$$

$$\delta x = -H^{-1} g$$

Newton – Rhapson Method

- For quadratic function it converges in one iteration
- For other general function, its asymptotic convergence is quadratic

- The disadvantage of this method is the high computation complexity of H^{-1}

Gauss – Newton Method

- The Gauss–Newton algorithm is a method used to solve non-linear least squares problems

$$f(x) \equiv \tfrac{1}{2} \triangle z(x)^{\top} W \triangle z(x)$$

$$g \equiv \frac{df}{dx} = \triangle z^{\top} W J \qquad H \equiv \frac{d^2 f}{dx^2} = J^{\top} W J + \sum_i (\triangle z^{\top} W)_i \frac{d^2 z_i}{dx^2}$$

$$H \approx J^{\top} W J. \qquad \frac{d^2 z_i}{dx^2} \approx 0$$

$$(J^{\top} W J)\, \delta x = -J^{\top} W \triangle z$$

Gauss – Newton Method

- For well-parametrized bundle problems under an outlier-free least squares cost model evaluated near the cost minimum, the Gauss-Newton approximation is usually very accurate

Levenberg – Marquardt Algorithm

- The LMA interpolates between the Gauss–Newton algorithm (GNA) and the method of gradient descent.

- When far from the minimum it acts as a steepest descent and it performs gauss newton iteration when $(\mathrm{H} + \lambda \mathrm{W})\,\delta \mathrm{x} = -\mathrm{g}$

Levenberg – Marquardt Algorithm

- It takes in to account the best of both gradient descent and gauss newton method

$$\lambda \gg 1 \Rightarrow Gradient\ Descent\ Method$$
$$\lambda < 1 \Rightarrow Gauss - Newton\ Method$$

General Facts about optimization methods

- Second order optimization methods like Gauss – Newton and LM requires a few but heavy iterations

- First order optimization methods like Gradient descent requires a lot of light iterations.

General Implementation Issues

- Exploit the problem structure
- Use factorization effectively
- Use stable local parametrizations
- Scaling and preconditioning

Computational Bottleneck in LM Iteration

- $$\delta x = -(H + \lambda W)^{-1} g$$

$H^{-1} \approx (J^T W J)^{-1}$ computation Is the main bottleneck

Network Graph representation of Jacobian and Hessian

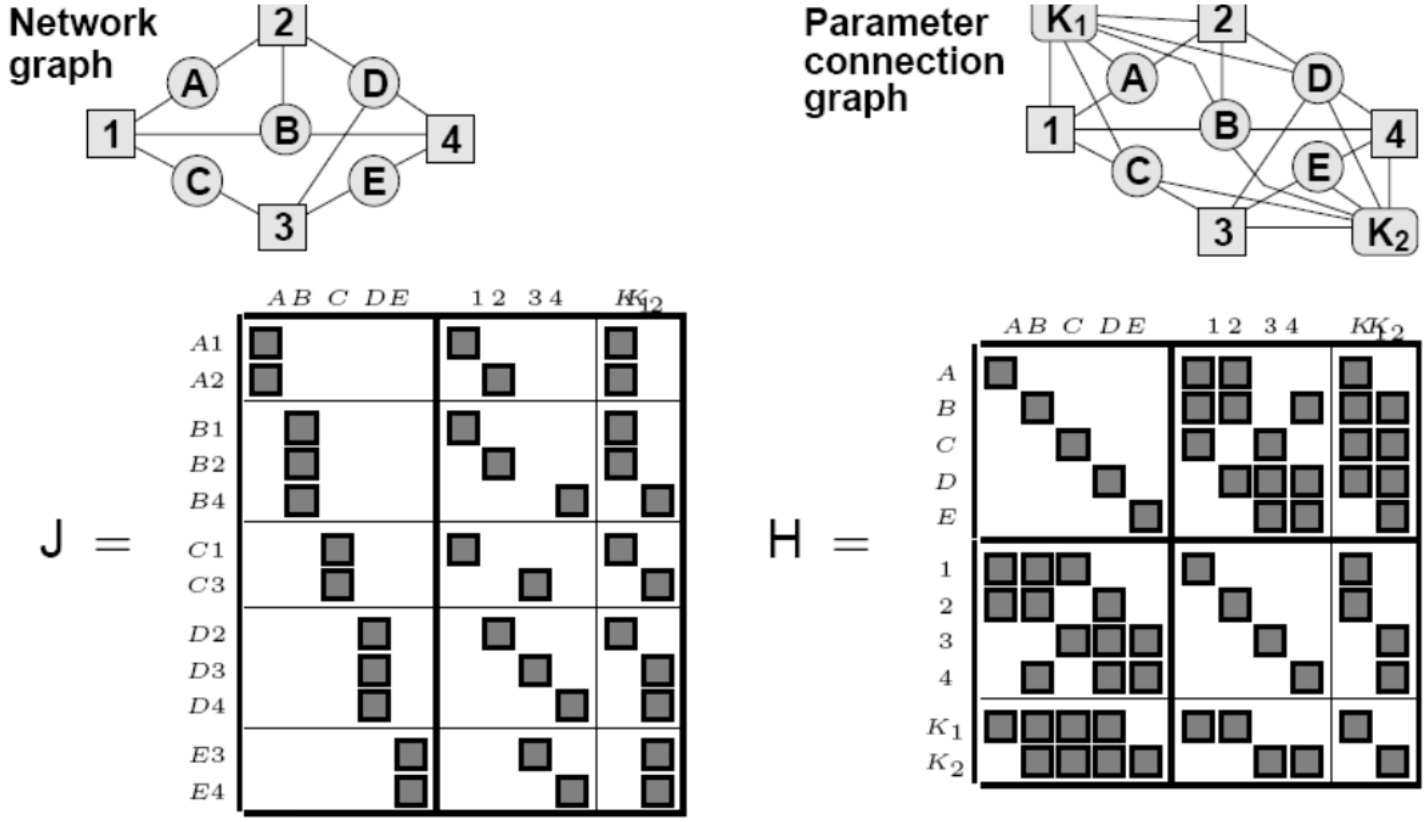


Fig. 3. The network graph, parameter connection graph, Jacobian structure and Hessian structure for a toy bundle problem with five 3D features A–E, four images 1–4 and two camera calibrations K_1 (shared by images 1,2) and K_2 (shared by images 3,4). Feature A is seen in images 1,2; B in 1,2,4; C in 1,3; D in 2–4; and E in 3,4.

H^{-1} Calculation Strategies

- The Schur Complement and the reduced camera system
- Cholesky Decomposition
- Sparse Factorization
  - Variable Ordering
    - Top down ordering
    - Bottom up ordering
  - Preconditioning
  - Conjugate Gradient method
  - Multigrid Methods

Schur Complement

$$H\delta x = \begin{pmatrix} U & W \\ W^T & V \end{pmatrix} \begin{pmatrix} \delta_a \\ \delta_b \end{pmatrix} = \begin{pmatrix} \varepsilon_a \\ \varepsilon_b \end{pmatrix}$$

Left Multiply $\begin{pmatrix} I & -WV^{*-1} \\ 0 & I \end{pmatrix}$ to both sides

$$\begin{pmatrix} U - WV^{-1}W^T & 0 \\ W^T & V \end{pmatrix} \begin{pmatrix} \delta_a \\ \delta_b \end{pmatrix} = \begin{pmatrix} \varepsilon_a - WV^{-1}\varepsilon_b \\ \varepsilon_b \end{pmatrix}$$

$$(U - WV^{-1}W^T)(\delta_a) = (\varepsilon_a - WV^{-1}\varepsilon_b)$$

Reduced Camera System

Cholesky Decomposition

Decompose the matrix A into $A = LL^T$, where L is a lower triangular matrix

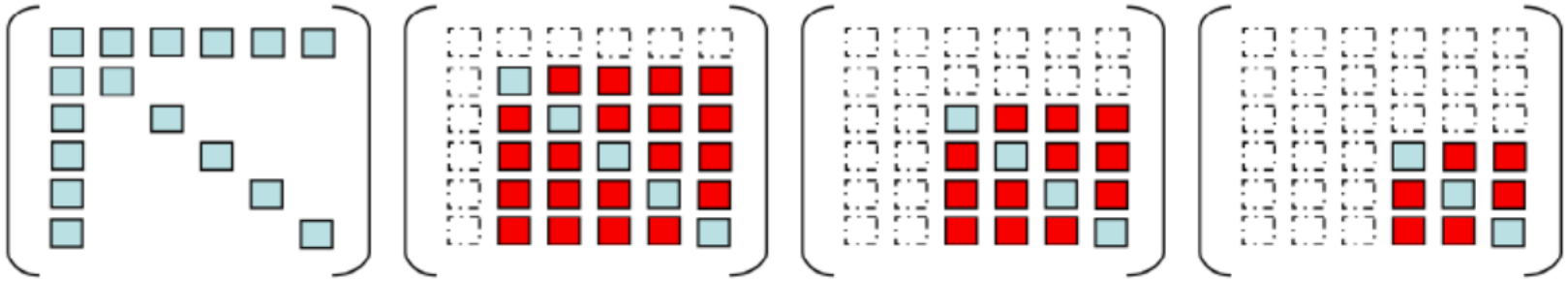
$$A = \left(\begin{array}{c|c} a_{11} & \star \\ \hline a_{21} & A_{22} \end{array}\right) = \left(\begin{array}{c|c} \lambda_{11} & 0 \\ \hline l_{21} & L_{22} \end{array}\right)\left(\begin{array}{c|c} \lambda_{11} & l_{21}^T \\ \hline 0 & L_{22}^T \end{array}\right) = \left(\begin{array}{c|c} \lambda_{11}^2 & \star \\ \hline \lambda_{11} l_{21} & l_{21} l_{21}^T + L_{22} L_{22}^T \end{array}\right)$$

1. Partition $A = \left(\begin{array}{c|c} \alpha_{11} & \star \\ \hline a_{21} & A_{22} \end{array}\right)$
2. $\alpha_{11} \leftarrow \lambda_{11} = \sqrt{\alpha_{11}}$
3. $a_{21} \leftarrow l_{21} = a_{21} / \lambda_{11}$
4. $A_{22} \leftarrow A_{22} - l_{21} l_{21}^T$
5. continue recursively with A_{22}

Sparse Factorization methods

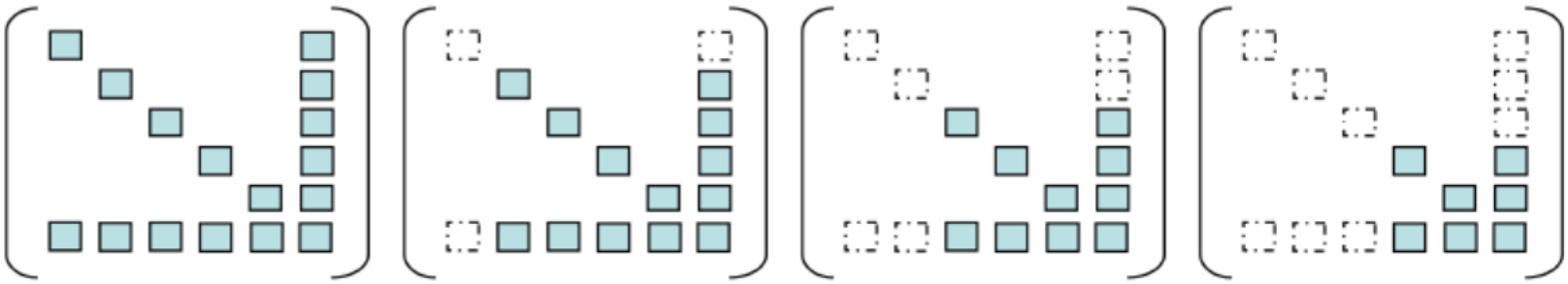
- Since both the Hessian and the reduced camera system is sparse for large scale systems, sparse factorization methods are preferred.
  - Variable Ordering
  - Preconditioning
  - Conjugate Gradient Method
  - Parallel Multigrid Methods

Basic Cholesky Factorization on Sparse Matrices



- There is a phenomenon of *fill – in.*
- After each step, we have more number of non – zeros which lead to more number of floating point operations.

Basic Cholesky Factorization on Sparse Matrices



- The effect of cholesky factorization after variables are re ordered creates the least fill-in

- The task of variable ordering is to reorder the matrix to create the least fill in.

Matrix Re-ordering

Finding the ordering which results in the least fill-in is a NP-complete problem

Some of the heuristics used are:

- Minimum Degree Reordering (Bottom – up approach)
- Nested Dissection (Top – Down approach)
  - These methods gives an idea of sparsity and structure of matrices.

Elimination Graph

- Graph G(A) of symmetric $n \times n$ matrix A is undirected graph having n vertices with edges between vertices *i* and *j* if $a_{ij} \neq 0$

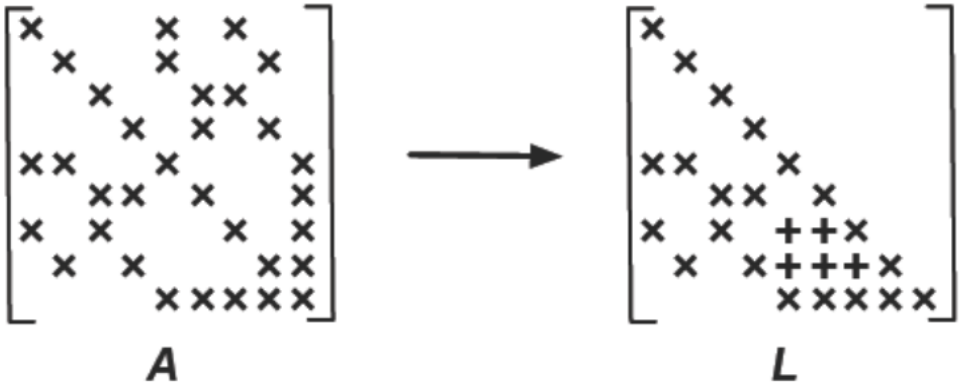
- At each step of Cholesky factorization algorithm, corresponding vertex is eliminated from the graph

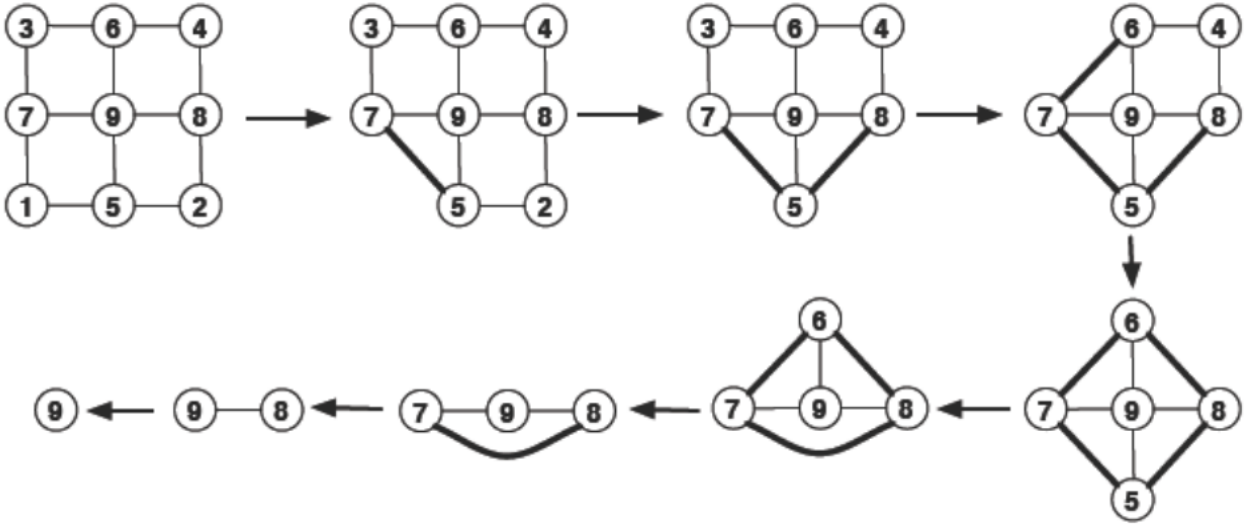
Elimination Graph

- Neighbors of eliminated vertex in previous graph become clique (fully connected subgraph) in modified graph.

- Entries of **A** that were initially zero, may become non zero entries, called fill

Elimination Graph





Minimum Degree Reordering

- Since finding the order of vertices with minimum fill in is a NP – Complete problem
- This is a greedy algorithm such that after each iteration we select a vertex with minimum degree.
- This is a bottom up method trying to minimize fill-in locally and greedily at each step, at the risk of global short sightedness

Nested Dissection

- Form the Elimination Graph.
- Recursively partition the graph into subgraphs using separators, small subsets of vertices the removal of which allows the graph to be partitioned into subgraphs with at most a constant fraction of the number of vertices.
- Perform Cholesky decomposition (a variant of Gaussian elimination for symmetric matrices), ordering the elimination of the variables by the recursive structure of the partition: each of the two subgraphs formed by removing the separator is eliminated first, and then the separator vertices are eliminated.

Preconditioning

- A Preconditioner P of a matrix A is a matrix such that $\mathrm{P}^{-1}A$ has a smaller condition number than A
- $\kappa(A) = \|A\| \, \|A^{-1}\|$
- If $P = A$, it gives a single iteration convergence, and finding the pre conditioner is as difficult as solving the linear system

Condition Number

- Defines the ill- conditioning or well- conditioning of a matrix
- $\kappa(A) = \|A\| \, \|A^{-1}\|$
- We cannot trust the solution if the system is ill-conditioned
- $\kappa(H) = \kappa(J^T J) = \kappa^2(J)$, so Hessian has a very large condition number, it requires a good preconditioning for conjugate gradient method
- Rate of convergence increases as the condition number of the matrix decreases

Conjugate Gradient Method

- It is an iterative method to solve a sparse system large enough to be handled by Cholesky decomposition

- Converges in at most *n* steps where n is the size of the matrix

Conjugate Gradient Method

$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$
$\mathbf{p}_0 := \mathbf{r}_0$
$k := 0$
repeat

$$\alpha_k := \frac{\mathbf{r}_k^{\mathrm{T}}\mathbf{r}_k}{\mathbf{p}_k^{\mathrm{T}}\mathbf{A}\mathbf{p}_k}$$
$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
if $\mathbf{r}_{k+1}$ is sufficiently small **then** exit loop **end if**

$$\beta_k := \frac{\mathbf{r}_{k+1}^{\mathrm{T}}\mathbf{r}_{k+1}}{\mathbf{r}_k^{\mathrm{T}}\mathbf{r}_k}$$
$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k$
$k := k + 1$

end repeat
The result is $\mathbf{x}_{k+1}$

Thank You