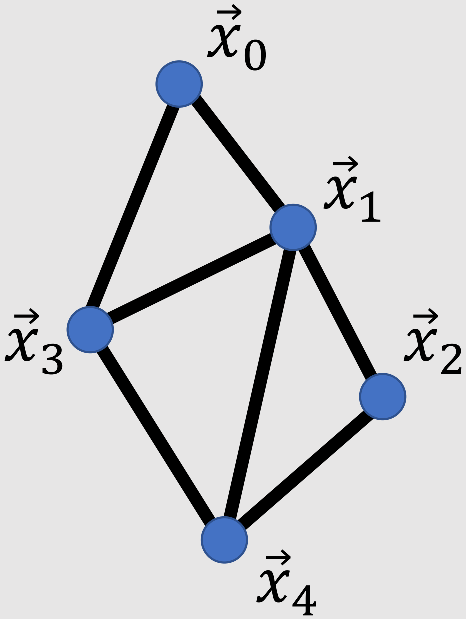
Linear System Solver

Adjacency Matrix

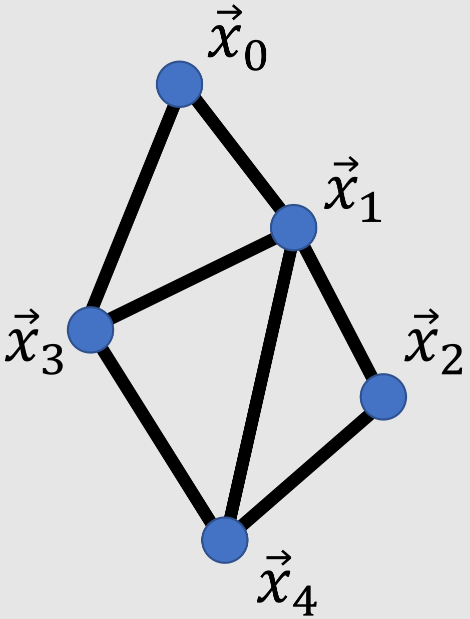
- Connected edges takes 1 in the matrix



$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Graph Laplacian Matrix

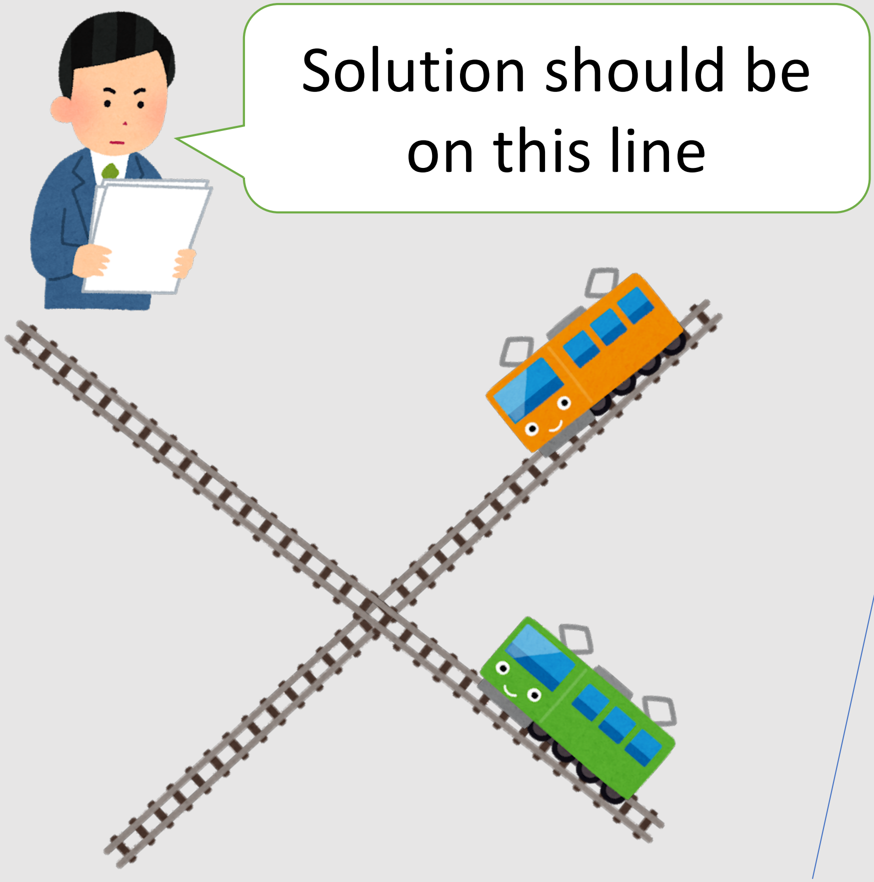
- All the connected edges takes -1 and diagonal takes valence



$$L = \begin{bmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ 0 & -1 & 2 & 0 & -1 \\ -1 & -1 & 0 & 3 & -1 \\ 0 & -1 & -1 & -1 & 3 \end{bmatrix}$$

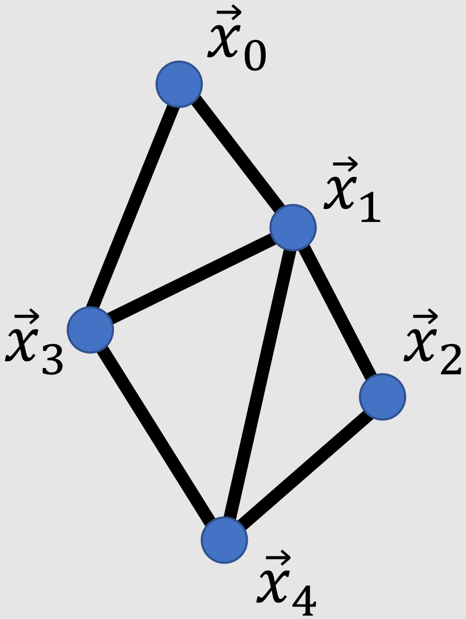
valence: # of connected points

Solving Constraints v.s. Optimization



Graph Laplacian Matrix as Constraints

- $L\vec{v} = 0$ means all the vertices are average of connected ones

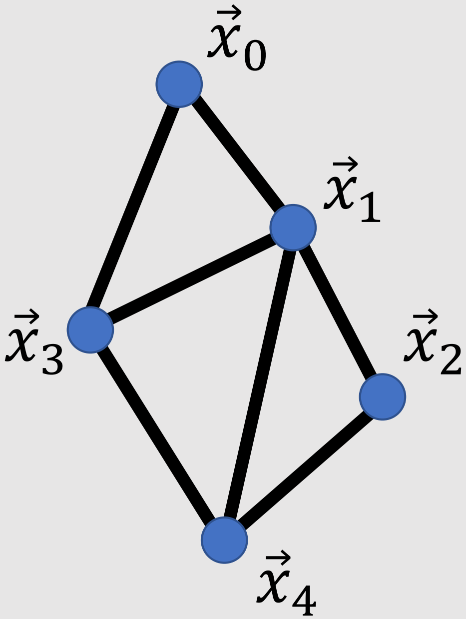


$$L\vec{v} = 0$$

$$\Longrightarrow \begin{bmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ 0 & -1 & 2 & 0 & -1 \\ -1 & -1 & 0 & 3 & -1 \\ 0 & -1 & -1 & -1 & 3 \end{bmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \end{pmatrix} = 0$$

Graph Laplacian Matrix as Optimization

- $L\vec{v} = 0$ means sum of square difference is minimized



$$W = \frac{1}{2}\sum_{e \subset \mathcal{E}} \left\| v_{e_1} - v_{e_2} \right\|^2$$

$$= \frac{1}{2}\vec{v}^T L \vec{v}$$

$$W \text{ is minimized} \rightarrow \frac{\partial W}{\partial \vec{v}} = L\vec{v} = 0$$

Diagonally Dominant Matrix

- Magnitude of diagonal element is larger than the sum of the magnitude of off-diagonal elements

$$|A_{ii}| \geq \sum_{j \neq i} |A_{ij}|$$

$$A = \begin{bmatrix} 2 & -1 & 0 & -1 & 0 \\ -1 & 4 & -1 & -1 & -1 \\ 0 & -1 & 2 & 0 & -1 \\ -1 & -1 & 0 & 3 & -1 \\ 0 & -1 & -1 & -1 & 3 \end{bmatrix}$$

Linear system with diagonally dominant matrix should be easy to solve

Types of Linear Solver

Direct Method
- Gaussian elimination
- LU decomposition

→ Compute the solution in a fixed procedure

Classical Iterative Methods
- Jacobi method
- Gauss-Seidel method

Krylov Subspace Method
- Conjugate gradient method

Classical Iterative Methods and Krylov Subspace Method → Update the solution iteratively

Krylov Subspace Method → Faster than the classical method

LU Decomposition

Triangular Matrix

lower triangle matrix

$$L = \begin{bmatrix} \ell_{1,1} & & & & 0 \\ \ell_{2,1} & \ell_{2,2} & & & \\ \ell_{3,1} & \ell_{3,2} & \ddots & & \\ \vdots & \vdots & \ddots & \ddots & \\ \ell_{n,1} & \ell_{n,2} & \dots & \ell_{n,n-1} & \ell_{n,n} \end{bmatrix}$$

upper triangle matrix

$$U = \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} & \dots & u_{1,n} \\ & u_{2,2} & u_{2,3} & \dots & u_{2,n} \\ & & \ddots & \ddots & \vdots \\ & & & \ddots & u_{n-1,n} \\ 0 & & & & u_{n,n} \end{bmatrix}$$

Forward Substitution

- It is very easy to solve linear system for triangular matrix

$$L\vec{x} = \vec{b}$$

$$\begin{matrix} \ell_{1,1}x_1 & & & & & & & = & b_1 \\ \ell_{2,1}x_1 & + & \ell_{2,2}x_2 & & & & & = & b_2 \\ \vdots & & \vdots & & \ddots & & & & \vdots \\ \ell_{m,1}x_1 & + & \ell_{m,2}x_2 & + & \cdots & + & \ell_{m,m}x_m & = & b_m \end{matrix}$$

$$x_1 = \frac{b_1}{\ell_{1,1}},$$

$$x_2 = \frac{b_2 - \ell_{2,1}x_1}{\ell_{2,2}},$$

$$\vdots$$

$$x_m = \frac{b_m - \sum_{i=1}^{m-1} \ell_{m,i}x_i}{\ell_{m,m}}.$$

Solving Linear System: LU Decomposition

$$A\vec{x} = \vec{b}$$

LU Decomposition

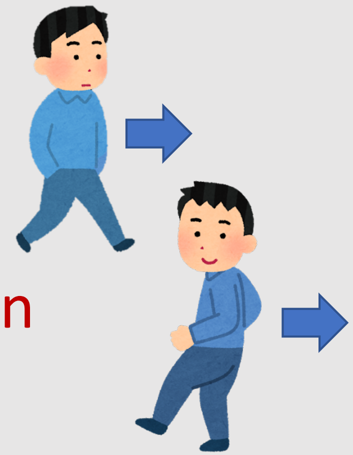
$$A = LU$$

$$L\underbrace{U\vec{x}}_{\vec{y}} = \vec{b}$$

Let $\vec{y} = U\vec{x}$, then $L\vec{y} = \vec{b}$

1. Solve $L\vec{y} = \vec{b}$ using forward substitution

2. Solve $U\vec{x} = \vec{y}$ using backward substitution



Block LU Decomposition

$$\begin{bmatrix} A & B \\ C & E \end{bmatrix} = \begin{bmatrix} I & 0 \\ CA^{-1} & I \end{bmatrix} \begin{bmatrix} A & B \\ 0 & E - CA^{-1}B \end{bmatrix}$$

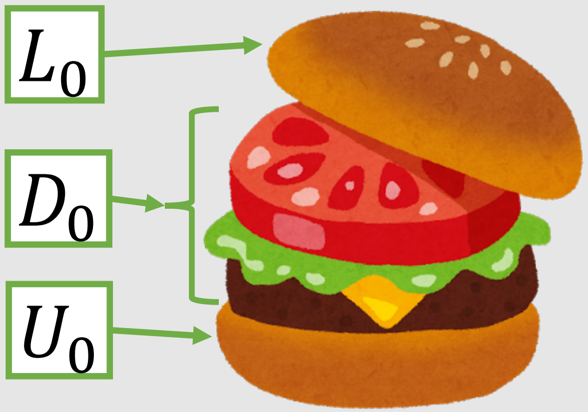
Schur compliment

$$= \begin{bmatrix} I & 0 \\ CA^{-1} & I \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & E - CA^{-1}B \end{bmatrix} \begin{bmatrix} I & A^{-1}B \\ 0 & I \end{bmatrix}$$

LDU Decomposition of 1st Row/Column

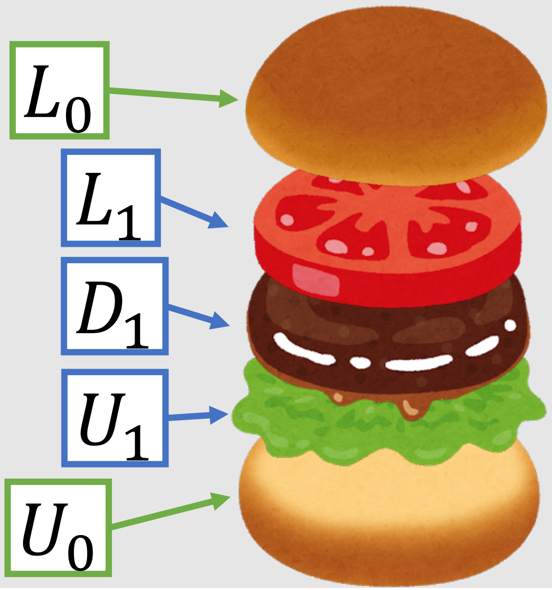
$$\begin{bmatrix} a_0 & \vec{b}_0^T \\ \vec{c}_0 & E_0 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ \vec{c}_0/a_0 & I \end{bmatrix}}_{L_0} \overbrace{\begin{bmatrix} a_0 & 0 \\ 0 & E_0 - \vec{c}_0 \vec{b}_0^T/a_0 \end{bmatrix}}^{D_0} \underbrace{\begin{bmatrix} 1 & \vec{b}_0^T/a_0 \\ 0 & I \end{bmatrix}}_{U_0}$$

$$E_0 - \vec{c}_0 \vec{b}_0^T/a_0 = \begin{bmatrix} a_1 & \vec{b}_1^T \\ \vec{c}_1 & E_1 \end{bmatrix}$$



LDU Decomposition of 2nd Row/Column

$$\begin{bmatrix} a_0 & \vec{b}_0^T \\ \vec{c}_0 & E_0 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 \\ \vec{c}_0/a_0 & I \end{bmatrix}}_{L_0} \underbrace{\begin{bmatrix} a_0 & & 0 \\ 0 & a_1 & \vec{b}_1^T \\ & \vec{c}_1 & E_1 \end{bmatrix}}_{} \underbrace{\begin{bmatrix} 1 & \vec{b}_0^T/a_0 \\ 0 & I \end{bmatrix}}_{U_0}$$



$$\underbrace{\begin{bmatrix} a_0 & & 0 \\ 0 & 1 & 0 \\ & \vec{c}_1/a_1 & I \end{bmatrix}}_{L_1} \begin{bmatrix} a_0 & & 0 \\ 0 & a_1 & 0 \\ & 0 & E_1 - \vec{c}_1 \vec{b}_1^T/a_1 \end{bmatrix} \underbrace{\begin{bmatrix} a_0 & & 0 \\ 0 & 1 & \vec{b}_1^T/a_1 \\ & 0 & I \end{bmatrix}}_{U_1}$$

LDU Decomposition

$$\begin{bmatrix} a_0 & \vec{b}_0^T \\ \vec{c}_0 & E_0 \end{bmatrix} = \underbrace{L_0 L_1 \cdots L_n}_{L} \underbrace{\begin{bmatrix} a_0 & & & \\ & a_1 & & \\ & & \ddots & \\ & & & a_n \end{bmatrix}}_{D} \underbrace{U_0 U_1 \cdots U_n}_{U}$$

Classical Iterative Solver

Gauss-Seidel Method

- Solve & update solution $\boldsymbol{x}$ row-by-row

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

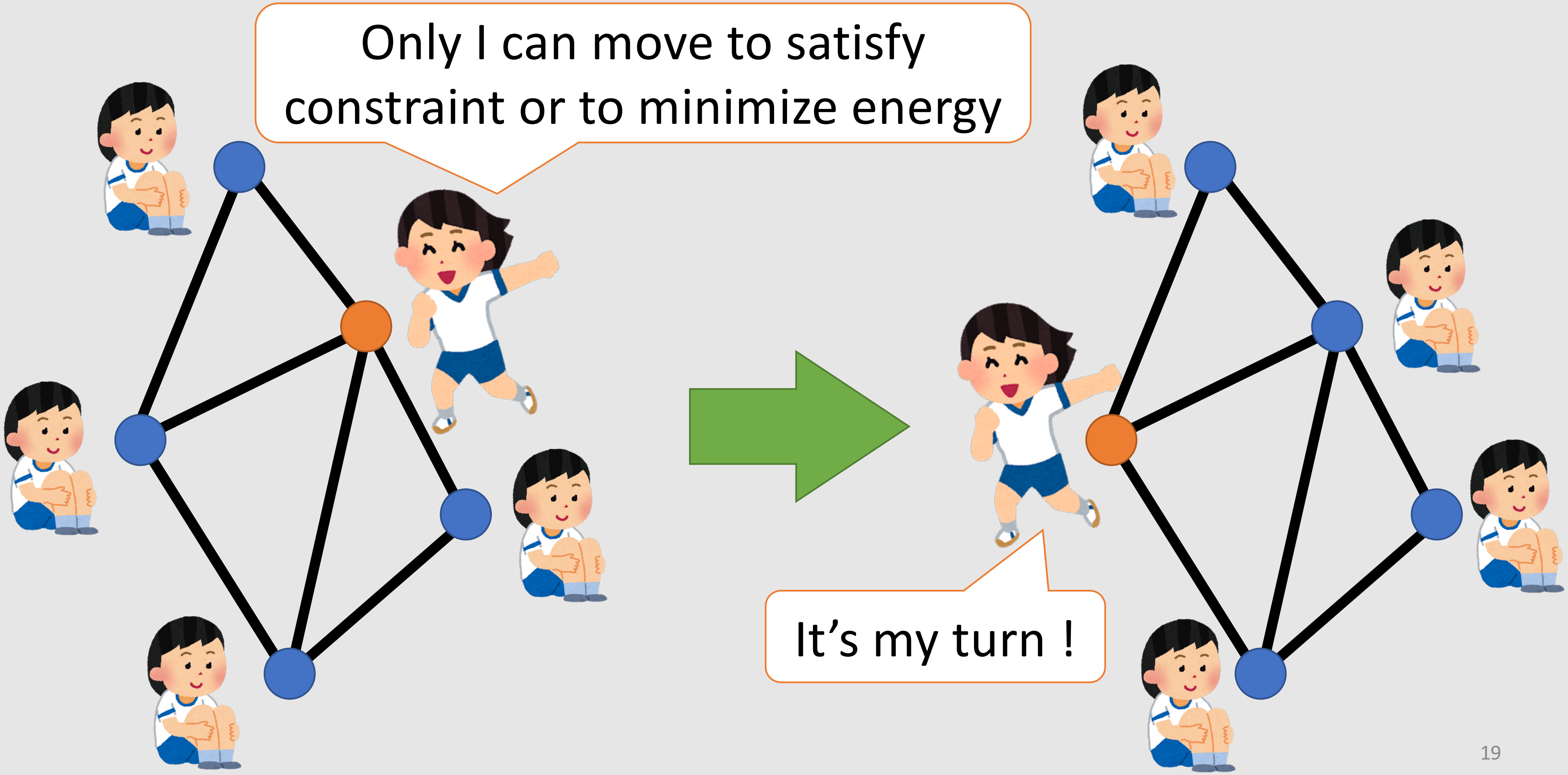
$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$

$$\Rightarrow x_1 = (b_1 - a_{12}x_2 - \cdots - a_{1n}x_n)/a_{11}$$

$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

$$\Rightarrow x_n = (b_n - a_{n1}x_1 - a_{n2}x_2 - \cdots)/a_{nn}$$

Gauss-Seidel Method in a Grid



Jacobi Method

1. Solve each row independently to obtain $\boldsymbol{x}'$

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1$$

$$\Rightarrow x_1{}' = (b_1 - a_{12}x_2 - \cdots - a_{1n}x_n)/a_{11}$$

$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n$$

$$\Rightarrow x_n{}' = (b_n - a_{n1}x_1 - a_{n2}x_2 - \cdots)/a_{nn}$$

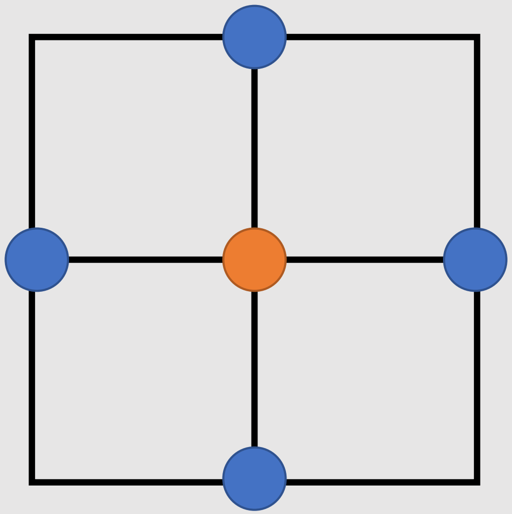
2. Update solution at the same time as $\boldsymbol{x} = \boldsymbol{x}'$

Stencil of a 2D Regular Grid

- Stencil represents the diagonal & off-diagonal component of matrix for a row

stencil in real life

credit: bukk @ wikipedia



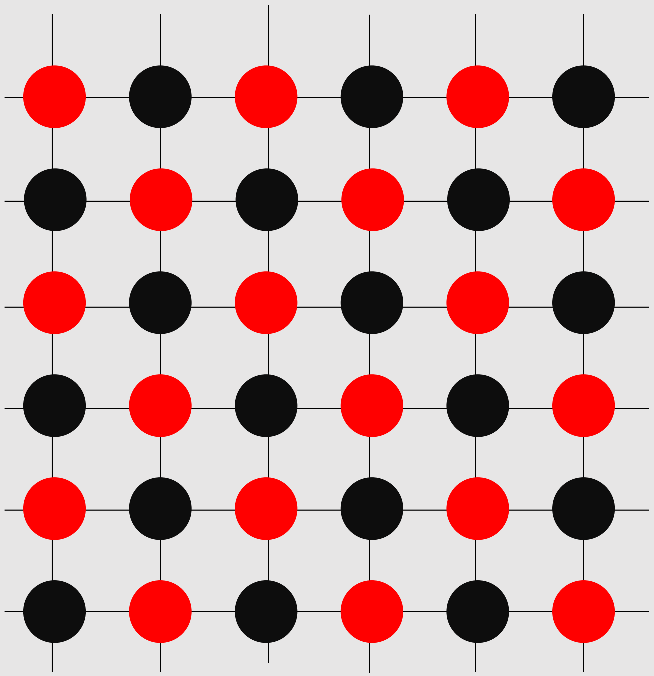
graph Laplacian stencil

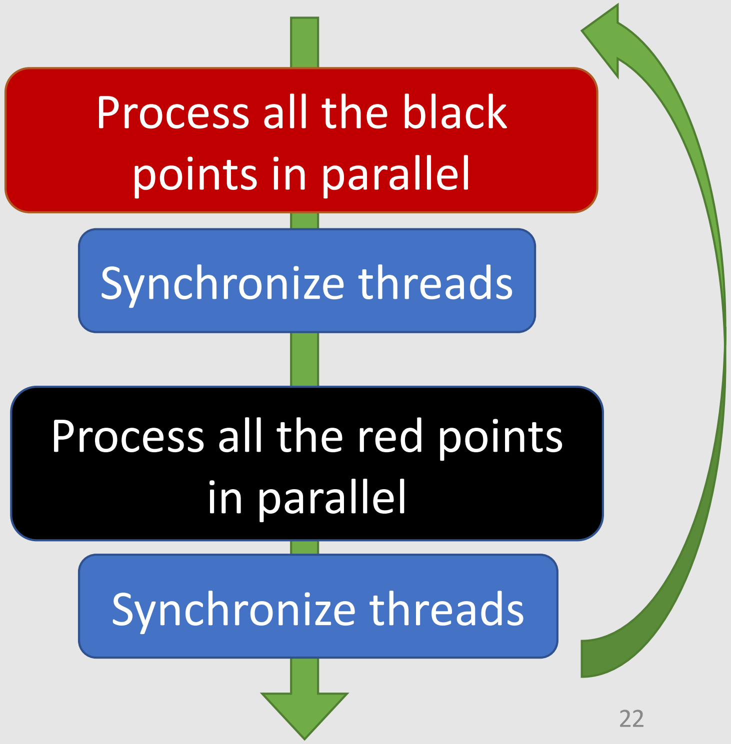
$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & \boxed{4} & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

diagonal component

Red-Black Ordering for Regular Grid

- The data of same color can be processed in any order (no-synchronization is necessary for parallel computation)





Krylov Subspace Method

Top 10 Algorithms of the 20 Century

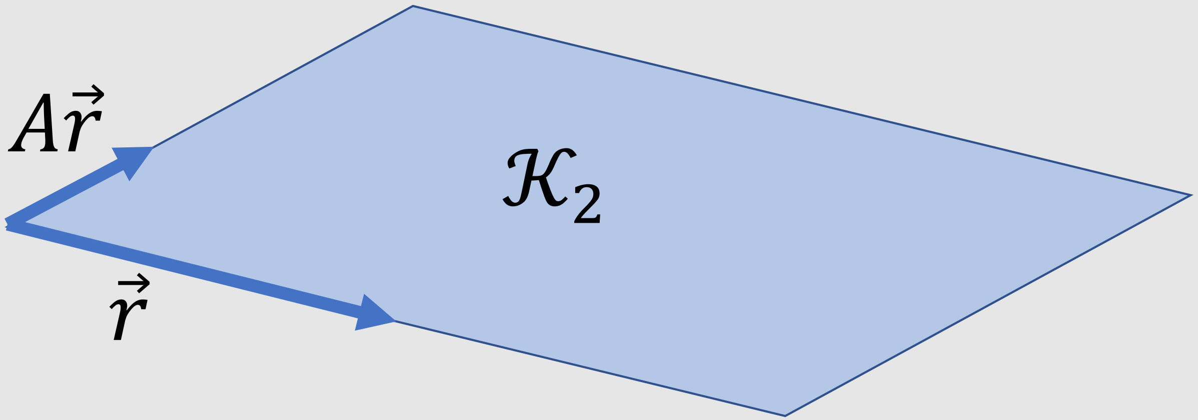
- 1946: The Metropolis Algorithm for Monte Carlo.
- 1947: Simplex Method for Linear Programming.
- **1950: Krylov Subspace Iteration Method.**
- 1951: The Decompositional Approach to Matrix Computations.
- 1957: The Fortran Optimizing Compiler.
- 1959: QR Algorithm for Computing Eigenvalues.
- 1962: Quicksort Algorithms for Sorting.
- 1965: Fast Fourier Transform.
- 1977: Integer Relation Detection.
- 1987: Fast Multipole Method

Jack Dongarra, Francis Sullivan, "Top Ten Algorithms of the Century", Computing in Science and Engineering, Volume 2, Number 1, January/February 2000, pages 22-23.

What is Krylov Subspace?

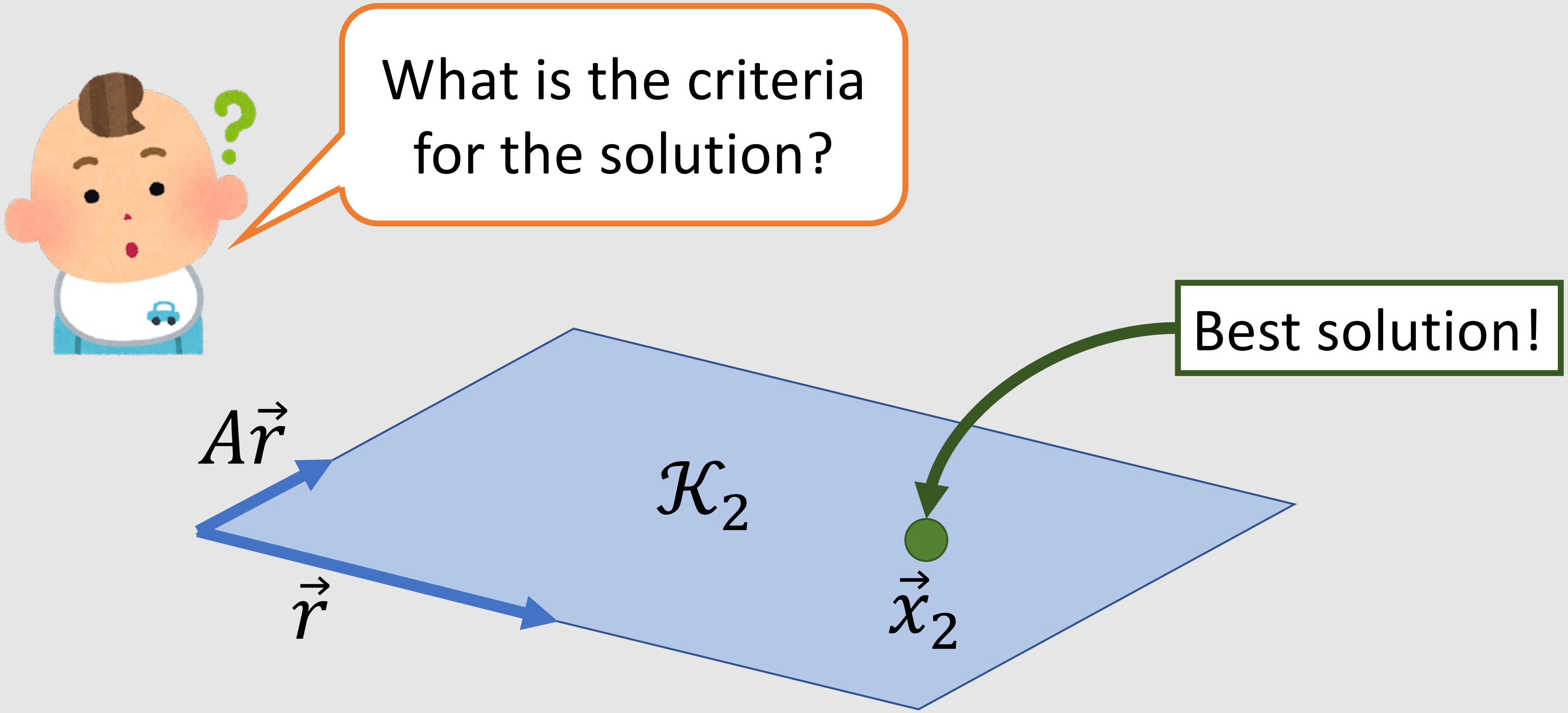
- Space spanned by a vector and its matrix multiplications

$$\mathcal{K}_k = \{\vec{r}, A\vec{r}, A^2\vec{r}, \cdots, A^{k-1}\vec{r}\}$$



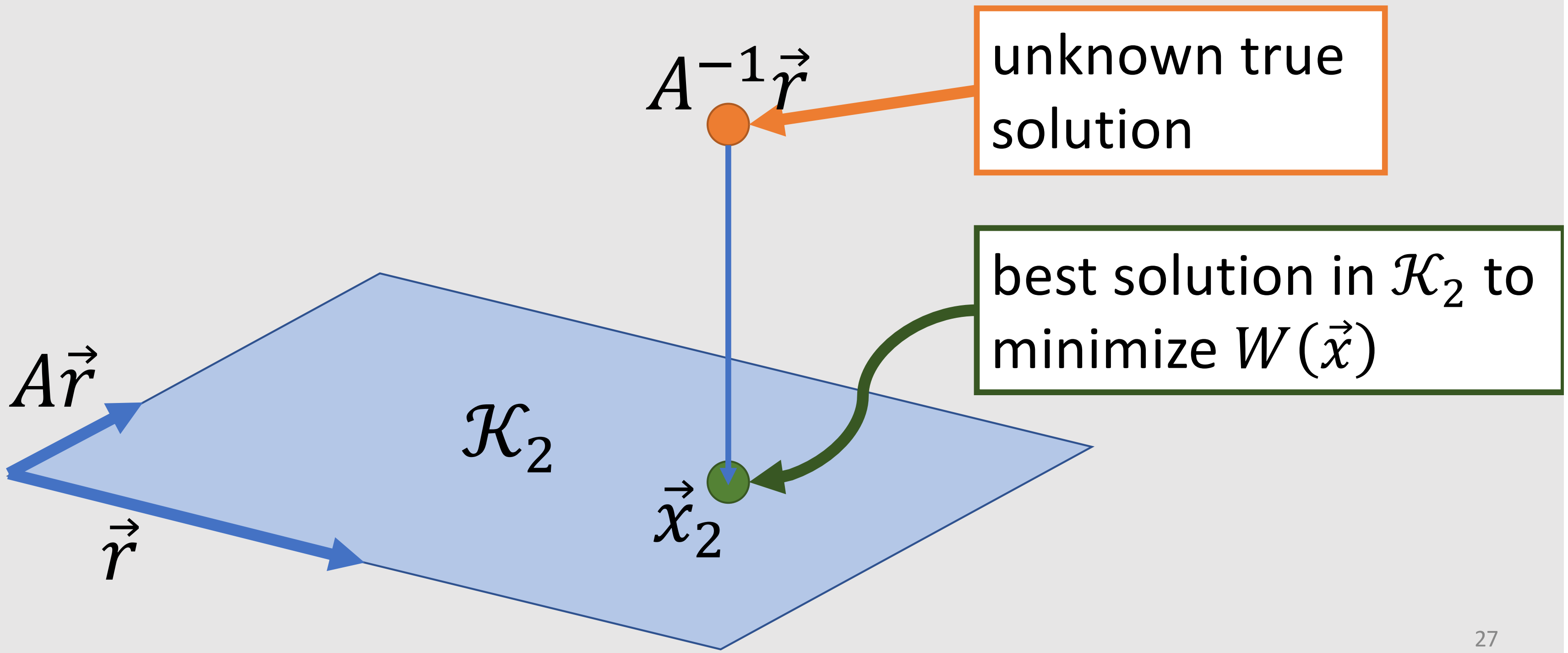
What is Krylov Subspace Method?

- Finding the best solution of a linear system in the Krylov subspace



What is Conjugate Gradient (CG) Method?

- Given a symmetric positive definite matrix A, the solution of $A\vec{x} = \vec{r}$ minimize $W(x) = 1/2\,\vec{x}^T A\vec{x} - \vec{r}^T\vec{x}$



Symmetric Positive Definite Matrix

- $\langle x, y \rangle_A = x^T A y$ has the property of inner product

1. $\langle x_1 + x_2, y \rangle_A = \langle x_1, y \rangle_A + \langle x_2, y \rangle_A$
2. $\langle \alpha x, y \rangle_A = \alpha \langle x, y \rangle_A$
3. $\langle x, y \rangle_A = \langle y, x \rangle_A$
4. $\langle x, y \rangle_A \geq 0$, and $\langle x, x \rangle_A = 0 \Longrightarrow x = 0$

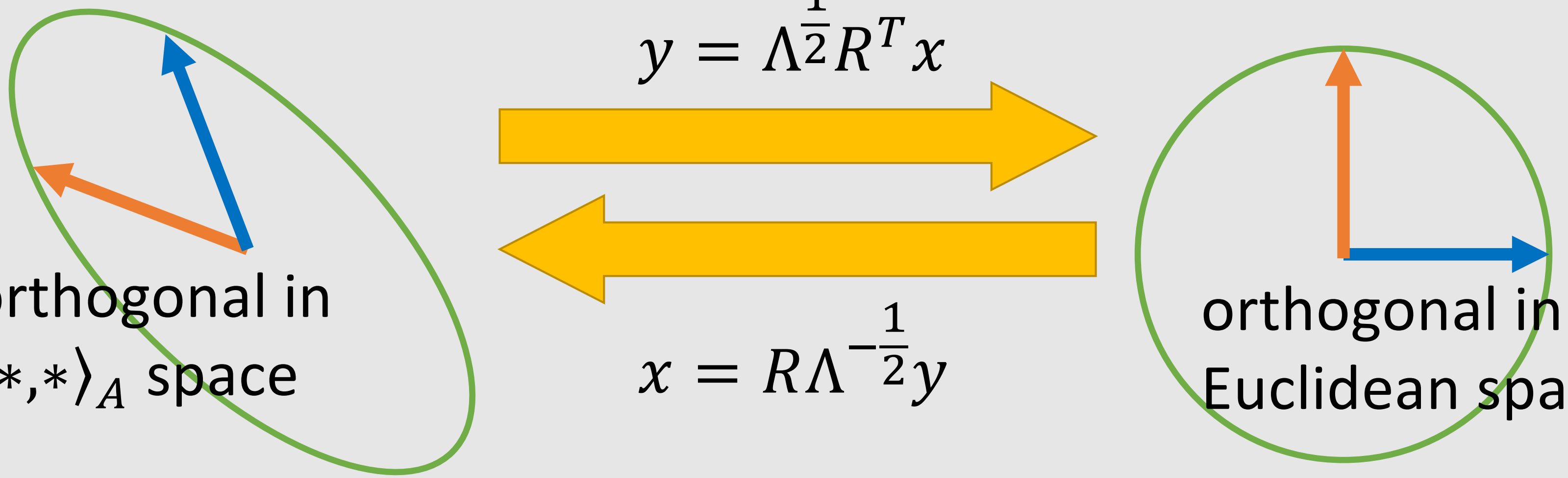
Symmetric Positive Definite Matrix

- All eigenvalues are positive, the eigenvectors are orthogonal

$$A = R\Lambda R^T$$

Unit circle in $\langle *,* \rangle_A$ space

Unit circle in Euclidean space



$$y = \Lambda^{\frac{1}{2}} R^T x$$

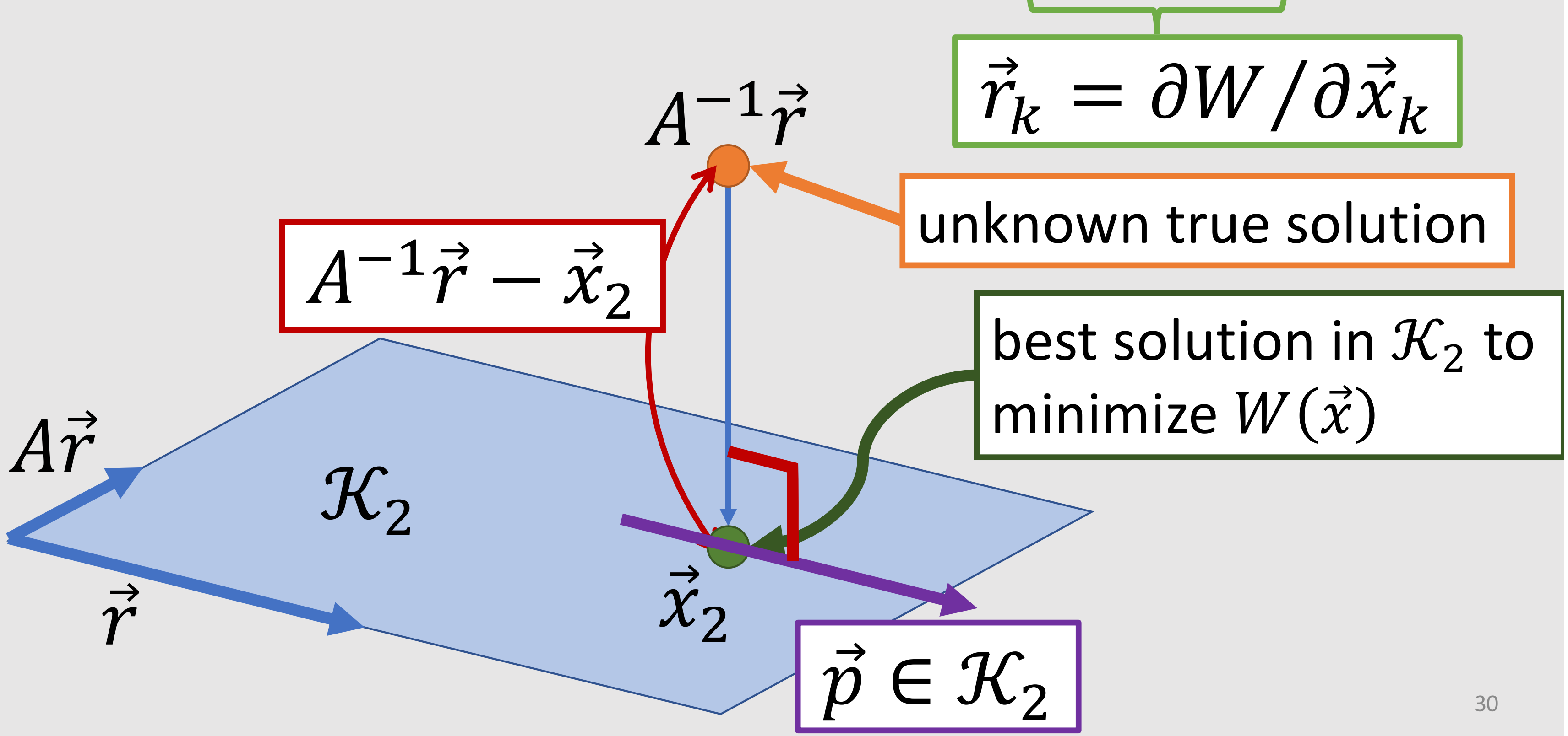
$$x = R\Lambda^{-\frac{1}{2}} y$$

orthogonal in $\langle *,* \rangle_A$ space

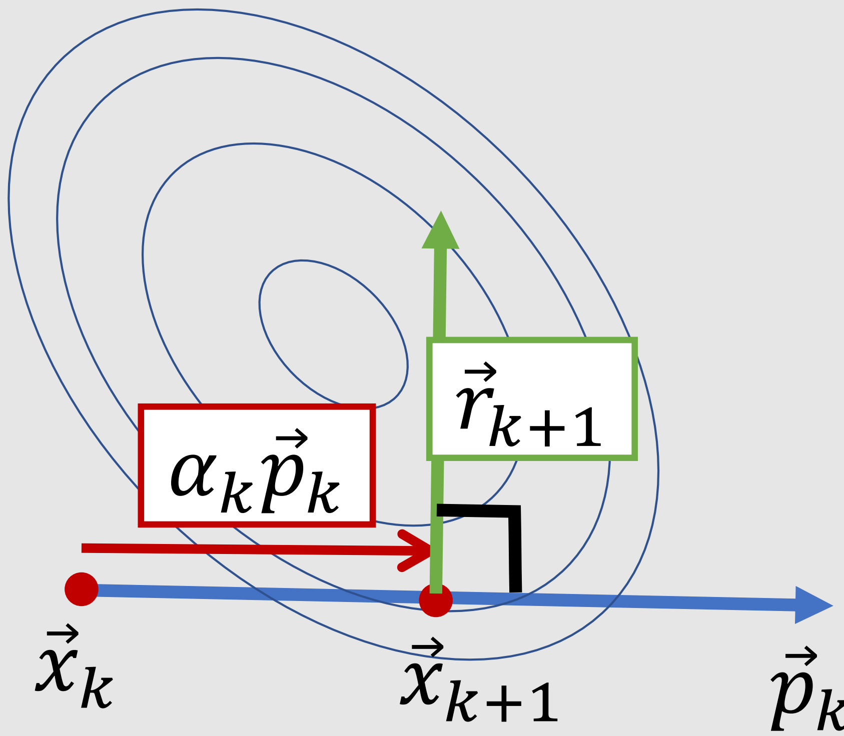
orthogonal in Euclidean space

A-Orthogonal Projection of the Solution

$$Find\ \vec{x}_k\ s.t.\ \langle \vec{p}, A^{-1}\vec{r} - \vec{x}_k \rangle_A = \vec{p} \cdot (\vec{r} - A\vec{x}_k) = 0$$



A-Orthogonal Projection on a Search Line



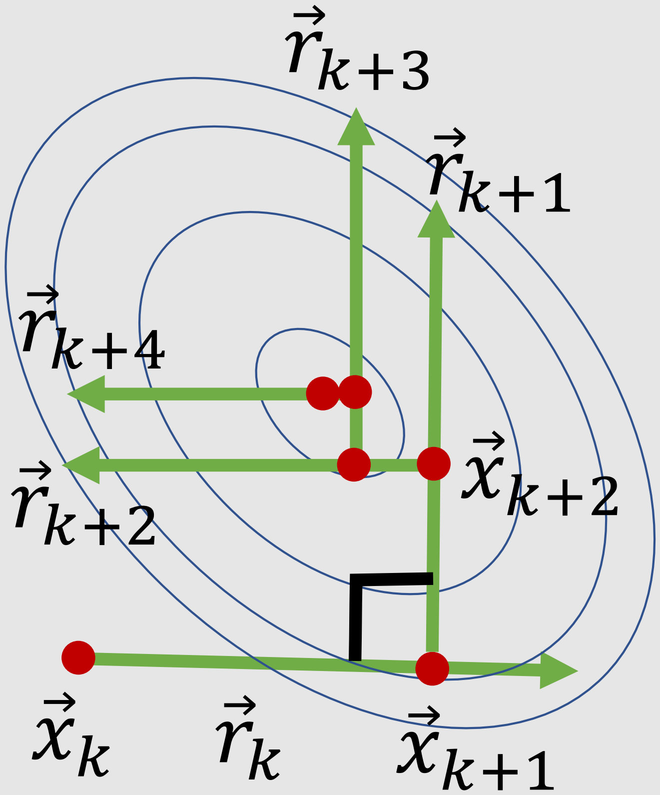
$$\vec{x}_{k+1} \coloneqq \vec{x}_k + \alpha_k \vec{p}_k$$

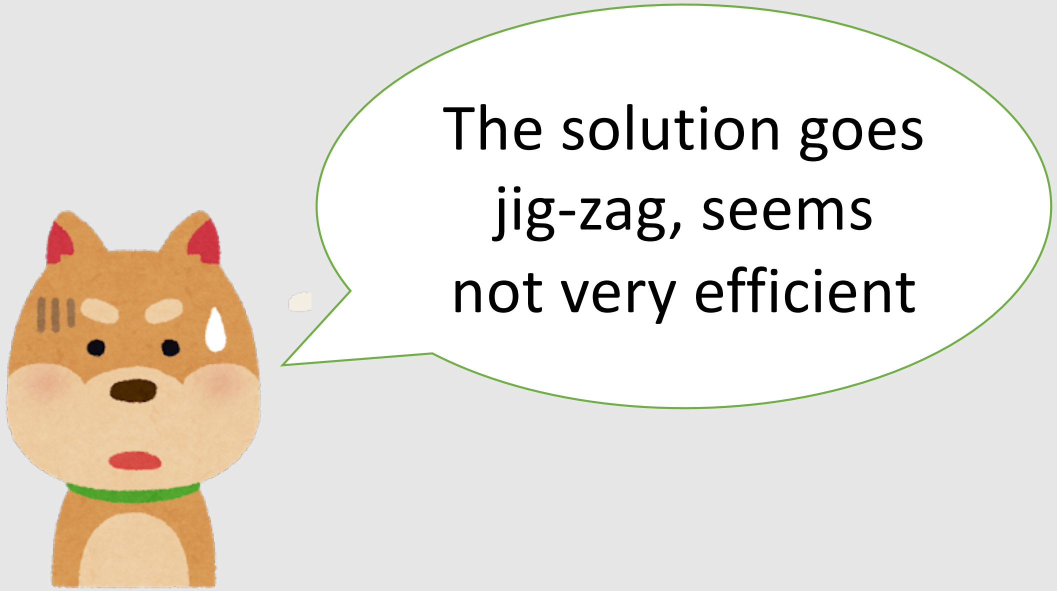
$$\langle \vec{p}_k, A^{-1}\vec{r} - \vec{x}_{k+1} \rangle_A = \vec{p}_k \cdot \vec{r}_{k+1} = 0$$

$$\alpha_k \coloneqq \frac{\vec{r}_{k+1}^T \vec{r}_{k+1}}{\vec{p}_k^T A \vec{p}_k}$$

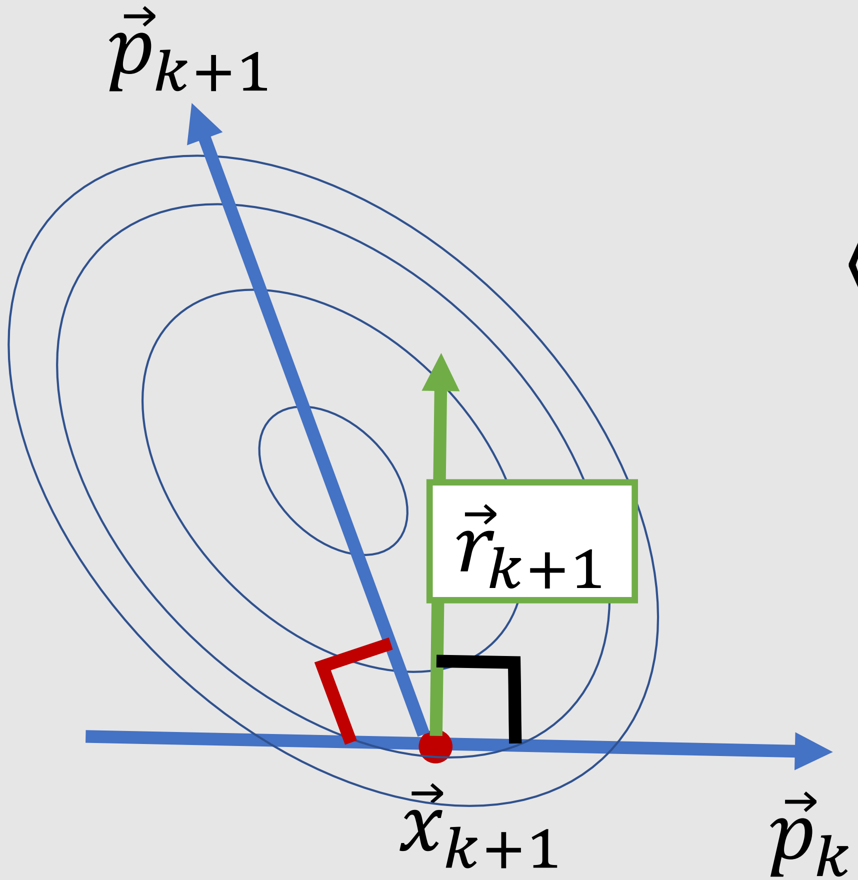
Poor Convergence of the Gradient Descent

- ☹ We cannot simply move along the residual $\vec{r}_k = \partial W / \partial \vec{x}_k$





Next Search Line is Chosen A-Orthogonal



$$\vec{p}_{k+1} \coloneqq \vec{r}_{k+1} + \beta_k \vec{p}_k$$

$$\langle \vec{p}_{k+1}, \vec{p}_k \rangle_A = 0$$

$$\beta_k \coloneqq -\frac{\vec{r}_{k+1}^T A \vec{p}_k}{\vec{p}_k^T A \vec{p}_k} = \frac{\vec{r}_{k+1}^T \vec{r}_{k+1}}{\vec{r}_k^T \vec{r}_k}$$

Conjugate Gradient Method Algorithm

$\vec{r}_0 = \vec{p}_0 = \vec{r}$

$\vec{x}_0 = 0$

for(k=0;k<k_max;++k){

$$\alpha_k := \frac{\vec{r}_k^T \vec{r}_k}{\vec{p}_k^T A \vec{p}_k}$$

$$\vec{x}_{k+1} := \vec{x}_k + \alpha_k \vec{p}_k$$

A-projection of the true solution on a search line

$$\vec{r}_{k+1} := \vec{r}_k - \alpha_k A \vec{p}_k$$

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

$$\vec{p}_{k+1} := \vec{r}_{k+1} + \beta_k \vec{p}_k$$

A-orthogonalization of the search line

}

Comparisons of Linear Solver

Direct Method

- Gaussian elimination
- LU decomposition

- ☺ Solve most non-singular matrices
- ☹ Costly for large matrix
- ☹ Cost is same for easy matrices

Classical Iterative Methods

- Jacobi method
- Gauss-Seidel method

- ☺ Simple implementation
- ☺ Cost is low for easy matrix
- ☹ Only for very easy matrix

Krylov Subspace Method

- Conjugate gradient method

- ☺ Simple implementation
- ☺ Faster than classical method
- ☺ More robust than classical method