Parameter Estimation

Considerations

- Error measures
- Physical meaning (algebraic, geometric, reprojection)
- # of parameters (DOF)
- # of data sets
- How parameters interact
- Solution methods
- Linear vs. nonlinear
- Exact vs. overdetermined
- Constrained vs. unconstrained
- SVD vs. normal equation

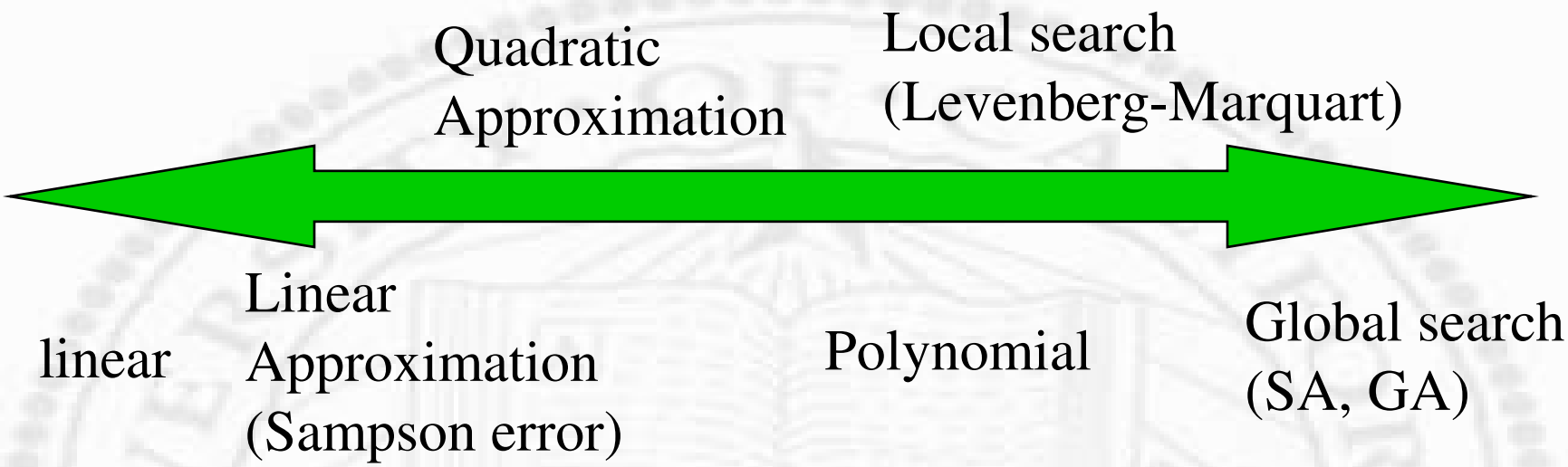
Important For many problems

- Planar homography
- Camera calibration
- Fundamental matrix
- Triangulation
- Trifocal tensor
- N-view geometry

Over-determined

- In real-world application, over-determination is the rule, not an exception
- "Exact" system of equations often time do not give "exact" solution (noise in data is unavoidable)

Complexity



- Linear requires data transformation
- Quadratic approximation does not require search
- Global search is not often used

Problems and Their Linear Formulations – Algebraic error

- Planar Homography

$$\mathbf{x}' = \mathbf{H}\mathbf{x}$$

$$\mathbf{x}' \times \mathbf{H}\mathbf{x} = \mathbf{0}$$

$$\mathbf{A}\mathbf{h} = \mathbf{0}$$

$$\|\mathbf{A}\mathbf{h}\|, \|\mathbf{h}\| = 1$$

- Fundamental matrix

$$\mathbf{x}'\mathbf{F}\mathbf{x} = 0$$

$$\mathbf{A}\mathbf{f} = \mathbf{0},$$

$$\|\mathbf{A}\mathbf{f}\|, \|\mathbf{f}\| = 1$$

- Camera Calibration

$$\mathbf{x} = \mathbf{P}\mathbf{X}$$

$$\mathbf{x} \times \mathbf{P}\mathbf{X} = \mathbf{0}$$

$$\mathbf{A}\mathbf{p} = \mathbf{0}$$

$$\|\mathbf{A}\mathbf{p}\|, \|\mathbf{p}\| = 1$$

- Triangulation

$$\mathbf{x} = \mathbf{P}\mathbf{X}$$

$$\mathbf{x} \times \mathbf{P}\mathbf{X} = \mathbf{0}$$

$$\mathbf{A}\mathbf{X} = \mathbf{0}$$

$$\|\mathbf{A}\mathbf{X}\|, \|\mathbf{X}\| = 1$$

Linear Methods

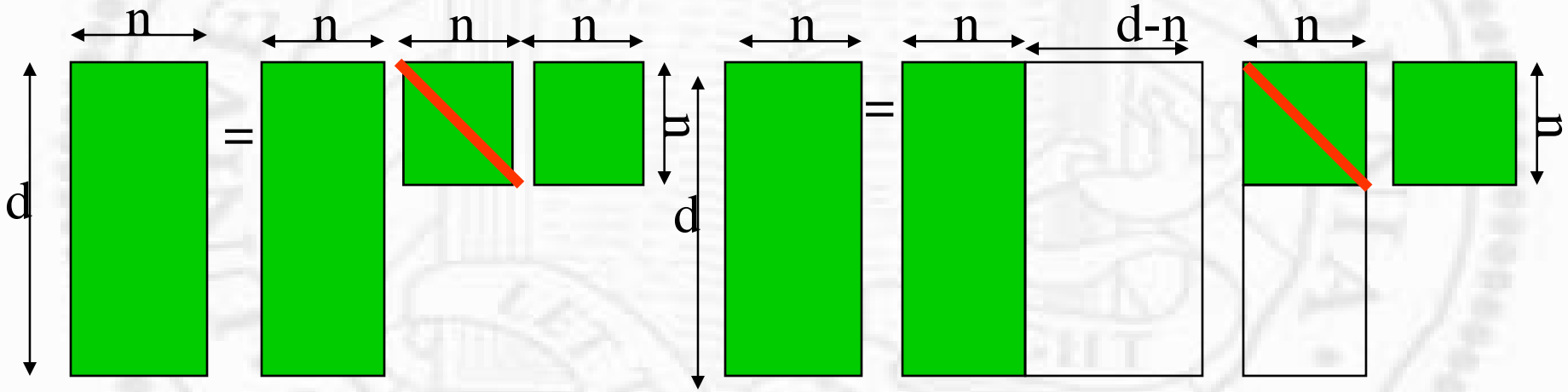
- SVD is a very general and powerful numerical method
  - Over-determined systems of equations
  - Constrained systems of equations

Physical Meaning

- $A = UDV^T$
  - U forms new orthogonal bases
  - D captures the importance (spread) along these new bases
  - V is the representation of A in new bases
  - Efficiency
    - You may not need all the bases
    - You may not need all singular values
    - You may not care about V

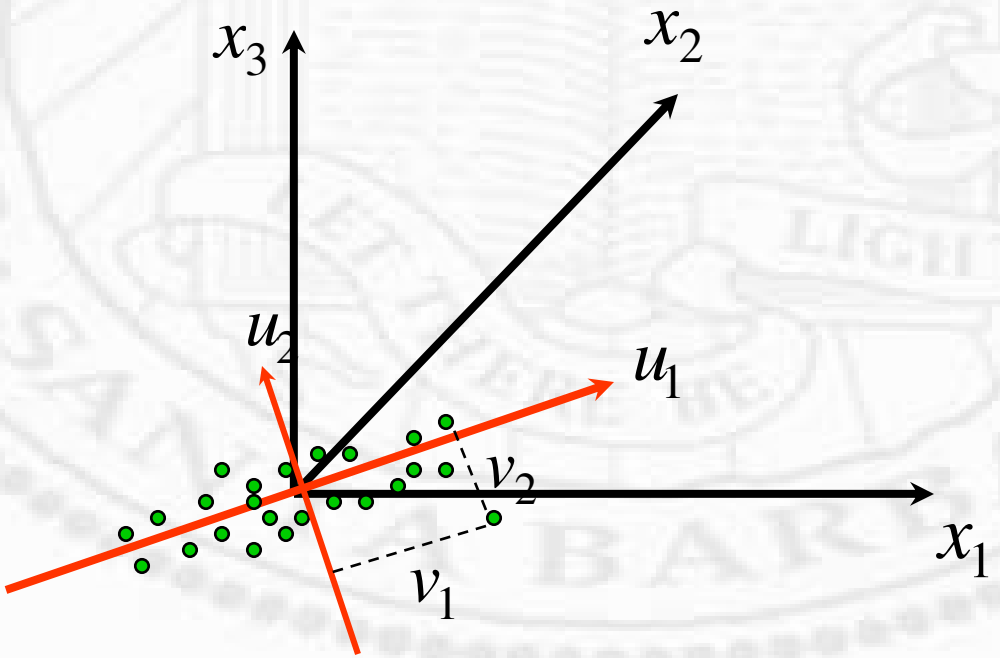
Two Different Conventions

$$\mathbf{X}_{d\times n} = \mathbf{U}_{d\times n}\boldsymbol{\Sigma}_{n\times n}\mathbf{V}^{t}{}_{n\times n} \qquad \mathbf{X}_{d\times n} = \mathbf{U}_{d\times d}\boldsymbol{\Sigma}_{d\times n}\mathbf{V}^{t}{}_{n\times n}$$



Important SVD properties

- Orthogonal bases
- Importance ranked axis direction
- Body-fitted coordinate system
- Uncorrelated components



In More Details

$$\begin{bmatrix} x_{11} & x_{12} & . & x_{1n} \\ x_{21} & x_{22} & . & x_{2n} \\ . & . & x_{ij} & . \\ x_{d1} & x_{d2} & . & x_{dn} \end{bmatrix}_{d\times n} = \begin{bmatrix} u_{11} & u_{12} & . & u_{1n} \\ u_{21} & u_{22} & . & u_{2n} \\ . & . & u_{ij} & . \\ u_{d1} & u_{d2} & . & u_{dn} \end{bmatrix}_{d\times n} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & . & 0 \\ 0 & 0 & \sigma_n \end{bmatrix}_{n\times n} \begin{bmatrix} v_{11} & . & v_{n1} \\ . & . & . \\ v_{1n} & . & v_{nn} \end{bmatrix}_{n\times n}$$

$$\begin{bmatrix} | & | & | & | \\ \mathbf{x}_1 & \mathbf{x}_2 & | & \mathbf{x}_n \\ | & | & | & | \\ | & | & | & | \end{bmatrix}_{d\times n} = \begin{bmatrix} | & | & | & | \\ \mathbf{u}_1 & \mathbf{u}_2 & | & \mathbf{u}_n \\ | & | & | & | \\ | & | & | & | \end{bmatrix}_{d\times n} \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & \sigma_n \end{bmatrix}_{n\times n} \begin{bmatrix} | & | & | & | \\ \mathbf{v}_1^T & \mathbf{v}_2^T & | & \mathbf{v}_n^T \\ | & | & | & | \\ | & | & | & | \end{bmatrix}_{n\times n}$$

$$\begin{bmatrix} | & | & | & | \\ \mathbf{x}_1 & \mathbf{x}_2 & | & \mathbf{x}_n \\ | & | & | & | \\ | & | & | & | \end{bmatrix}_{d\times n} = \begin{bmatrix} | & | & | & | \\ \sigma_1\mathbf{u}_1 & \sigma_2\mathbf{u}_2 & | & \sigma_n\mathbf{u}_n \\ | & | & | & | \\ | & | & | & | \end{bmatrix}_{d\times n} \begin{bmatrix} | & | & | & | \\ \mathbf{v}_1^T & \mathbf{v}_2^T & | & \mathbf{v}_n^T \\ | & | & | & | \\ | & | & | & | \end{bmatrix}_{n\times n}$$

$$\mathbf{X}_{d\times n} = \mathbf{U}_{d\times n}\boldsymbol{\Sigma}_{n\times n}\mathbf{V}^t_{n\times n}$$

$$\mathbf{x}_i = \mathbf{U}\,\Sigma\,\mathbf{v}_i = \sum_j \sigma_j \mathbf{u}_j v_{ij}^T$$

$$= v_{i1}^T \sigma_1 \mathbf{u}_1 + v_{i2}^T \sigma_2 \mathbf{u}_2 + \cdots + v_{in}^T \sigma_n \mathbf{u}_n$$

In More Details

$$\begin{bmatrix} | & | & | & | \\ \mathbf{x}_1 & \mathbf{x}_2 & | & \mathbf{x}_n \\ | & | & | & | \end{bmatrix}_{d\times n} = \begin{bmatrix} | & | & | & | \\ \sigma_1\mathbf{u}_1 & \sigma_2\mathbf{u}_2 & | & \sigma_n\mathbf{u}_n \\ | & | & | & | \end{bmatrix}_{d\times n} \begin{bmatrix} | & | & | & | \\ \mathbf{v}_1^T & \mathbf{v}_2^T & | & \mathbf{v}_n^T \\ | & | & | & | \end{bmatrix}_{n\times n}$$

$$\mathbf{X}_{d\times n} = \mathbf{U}_{d\times n}\boldsymbol{\Sigma}_{n\times n}\mathbf{V}^{t}_{n\times n}$$

$$\mathbf{x}_i = \mathbf{U}\,\Sigma\,\mathbf{v}_i^T = \sum_j \sigma_j \mathbf{u}_j v_{ij}^T$$

$$= v_{i1}^T\sigma_1\mathbf{u}_1 + v_{i2}^T\sigma_2\mathbf{u}_2 + \cdots + v_{in}^T\sigma_n\mathbf{u}_n$$

Projection onto a new basis

In More Details

$$\begin{bmatrix} | & | & | & | \\ \mathbf{x}_1 & \mathbf{x}_2 & | & \mathbf{x}_n \\ | & | & | & | \end{bmatrix}_{d\times n} = \begin{bmatrix} | & | & | & | \\ \sigma_1\mathbf{u}_1 & \sigma_2\mathbf{u}_2 & | & \sigma_n\mathbf{u}_n \\ | & | & | & | \end{bmatrix}_{d\times d} \begin{bmatrix} | & | & | & | \\ \mathbf{v}_1^T & \mathbf{v}_2^T & | & \mathbf{v}_n^T \\ | & | & | & | \end{bmatrix}_{n\times n}$$

$$\mathbf{X}_{d\times n} = \mathbf{U}_{d\times n}\,\mathbf{\Sigma}_{n\times n}\,\mathbf{V}^{t}_{n\times n}$$

$$\mathbf{x}_i = \mathbf{U}\,\Sigma\,\mathbf{v}_i^T = \sum_j \sigma_j \mathbf{u}_j v_{ij}^T$$

$$= v_{i1}^T \sigma_1 \mathbf{u}_1 + v_{i2}^T \sigma_2 \mathbf{u}_2 + \cdots + v_{in}^T \sigma_n \mathbf{u}_n = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}_{n\times n} \begin{bmatrix} | & | & | & | \\ \mathbf{v}_1^T & \mathbf{v}_2^T & | & \mathbf{v}_n^T \\ | & | & | & | \end{bmatrix}_{n\times n}$$

$$\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}_{n\times 1} \qquad \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}_{n\times 1} \qquad \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}_{n\times 1}$$

Comparison

- Original space
- Transformed space

$$\mathbf{x}_i = \mathbf{U}\,\Sigma\,\mathbf{v}_i^T = \sum_j \sigma_j \mathbf{u}_j v_{ij}^T$$

$$= v_{i1}^T \sigma_1 \mathbf{u}_1 + v_{i2}^T \sigma_2 \mathbf{u}_2 + \cdots + v_{in}^T \sigma_n \mathbf{u}_n$$

$$\mathbf{v}_i^T = \Sigma^{-1}\mathbf{U}^T \mathbf{x}_i$$

Projection + stretching + Rotation operator

Furthermore

$$\mathbf{X}_{d\times n} = \mathbf{U}_{d\times d}\boldsymbol{\Sigma}_{d\times n}\mathbf{V}^{t}{}_{n\times n}$$

$$\mathbf{X}_{d\times n}\mathbf{X}^{t}{}_{n\times d} = (\mathbf{U}_{d\times d}\boldsymbol{\Sigma}_{d\times n}\mathbf{V}^{t}{}_{n\times n})(\mathbf{V}_{n\times n}\boldsymbol{\Sigma}^{t}{}_{d\times n}\mathbf{U}^{t}{}_{d\times d})$$

$$= \mathbf{U}_{d\times d}\boldsymbol{\Sigma}^{2}{}_{d\times n}\mathbf{U}^{t}{}_{d\times d}$$

where

$$\mathbf{X}_{d\times n}\mathbf{X}^{t}{}_{n\times d} = \begin{bmatrix} x_{11} & x_{12} & . & x_{1n} \\ x_{21} & x_{22} & . & x_{2n} \\ . & . & x_{ij} & . \\ x_{d1} & x_{d2} & . & x_{dn} \end{bmatrix}_{d\times n} \begin{bmatrix} x_{11} & x_{21} & . & x_{d1} \\ x_{12} & x_{22} & . & x_{d2} \\ . & . & x_{ji} & . \\ x_{1n} & x_{2n} & . & x_{dn} \end{bmatrix}_{n\times d}$$

$$\begin{bmatrix} \sum_k x_{1k}^2 & \sum_k x_{1k}x_{2k} & . & \sum_k x_{1k}x_{dk} \\ \sum_k x_{2k}x_{1k} & \sum_k x_{2k}^2 & . & \sum_k x_{2k}x_{dk} \\ . & . & \sum_k x_{ik}x_{jk} & . \\ \sum_k x_{dk}x_{1k} & \sum_k x_{dk}x_{2k} & . & \sum_k x_{dk}^2 \end{bmatrix}_{d\times d}$$

Furthermore

$$\mathbf{X}_{d\times n}\mathbf{X}^{T}{}_{n\times d} = \left(\begin{bmatrix} | & 0 & 0 \\ \mathbf{X}_1 & 0 & 0 \\ | & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & | & 0 \\ 0 & \mathbf{X}_2 & 0 \\ 0 & | & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & | \\ 0 & 0 & \mathbf{X}_3 \\ 0 & 0 & | \end{bmatrix}\right)$$

$$\left(\begin{bmatrix} - & \mathbf{X}_1 & - \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ - & \mathbf{X}_2 & - \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ - & \mathbf{X}_3 & - \end{bmatrix}\right)$$

$$= \begin{bmatrix} | & 0 & 0 \\ \mathbf{X}_1 & 0 & 0 \\ | & 0 & 0 \end{bmatrix}\begin{bmatrix} - & \mathbf{X}_1 & - \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & | & 0 \\ 0 & \mathbf{X}_2 & 0 \\ 0 & | & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 \\ - & \mathbf{X}_2 & - \\ 0 & 0 & 0 \end{bmatrix} +$$

$$\begin{bmatrix} 0 & 0 & | \\ 0 & 0 & \mathbf{X}_3 \\ 0 & 0 & | \end{bmatrix}\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ - & \mathbf{X}_3 & - \end{bmatrix}$$

Furthermore

$$\begin{bmatrix} | & 0 & 0 \\ \mathbf{x}_1 & 0 & 0 \\ | & 0 & 0 \end{bmatrix} \begin{bmatrix} - & \mathbf{x}_1 & - \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} x_{11}x_{11} & x_{11}x_{12} & x_{11}x_{13} \\ x_{12}x_{11} & x_{12}x_{12} & x_{12}x_{13} \\ x_{13}x_{11} & x_{13}x_{12} & x_{13}x_{13} \end{bmatrix}$$

$$\begin{bmatrix} 0 & | & 0 \\ 0 & \mathbf{x}_2 & 0 \\ 0 & | & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ - & \mathbf{x}_2 & - \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} x_{21}x_{21} & x_{21}x_{22} & x_{21}x_{23} \\ x_{22}x_{21} & x_{22}x_{22} & x_{22}x_{23} \\ x_{23}x_{21} & x_{23}x_{22} & x_{23}x_{23} \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & | \\ 0 & 0 & \mathbf{x}_3 \\ 0 & 0 & | \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ - & \mathbf{x}_3 & - \end{bmatrix} = \begin{bmatrix} x_{31}x_{31} & x_{31}x_{32} & x_{31}x_{33} \\ x_{32}x_{31} & x_{32}x_{32} & x_{32}x_{33} \\ x_{33}x_{31} & x_{33}x_{32} & x_{33}x_{33} \end{bmatrix}$$

Furthermore

where

$$\mathbf{X}_{d\times n}\mathbf{X}^{t}{}_{n\times d} = \begin{bmatrix} x_{11} & x_{12} & . & x_{1n} \\ x_{21} & x_{22} & . & x_{2n} \\ . & . & x_{ij} & . \\ x_{d1} & x_{d2} & . & x_{dn} \end{bmatrix}_{d\times n} \begin{bmatrix} x_{11} & x_{21} & . & x_{d1} \\ x_{12} & x_{22} & . & x_{d2} \\ . & . & x_{ji} & . \\ x_{1n} & x_{2n} & . & x_{dn} \end{bmatrix}_{n\times d}$$

$$\begin{bmatrix} \sum_k x_{1k}{}^2 & \sum_k x_{1k}x_{2k} & . & \sum_k x_{1k}x_{dk} \\ \sum_k x_{2k}x_{1k} & \sum_k x_{2k}{}^2 & . & \sum_k x_{2k}x_{dk} \\ . & . & \sum_k x_{ik}x_{jk} & . \\ \sum_k x_{dk}x_{1k} & \sum_k x_{dk}x_{2k} & . & \sum_k x_{dk}{}^2 \end{bmatrix}_{d\times d}$$

Covariance matrix – if zero centered

Mahalanobis distance

$$\mathbf{C}_{d\times d} = \mathbf{U}_{d\times d}\boldsymbol{\Sigma}^2{}_{d\times n}\mathbf{U}^T{}_{d\times d}$$

$$\mathbf{C}^{-1}_{d\times d} = (\mathbf{U}_{d\times d}\boldsymbol{\Sigma}^2{}_{d\times n}\mathbf{U}^T{}_{d\times d})^{-1} = \mathbf{U}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}^{-1}\mathbf{U}^t$$

$$(\mathbf{x}-\mathbf{y})^T\mathbf{C}^{-1}_{d\times d}(\mathbf{x}-\mathbf{y}) = (\mathbf{x}-\mathbf{y})^T\mathbf{U}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}^{-1}\mathbf{U}^t(\mathbf{x}-\mathbf{y})$$

$$= (\boldsymbol{\Sigma}^{-1}\mathbf{U}^t(\mathbf{x}-\mathbf{y}))^T(\boldsymbol{\Sigma}^{-1}\mathbf{U}^t(\mathbf{x}-\mathbf{y}))$$

$$= \mathbf{e}^T\mathbf{e}$$

$$= \mathbf{e}\cdot\mathbf{e}$$

- Find orthogonal directions
- properly compensates for variance (spread) along different orthogonal directions

Correlation Coefficients

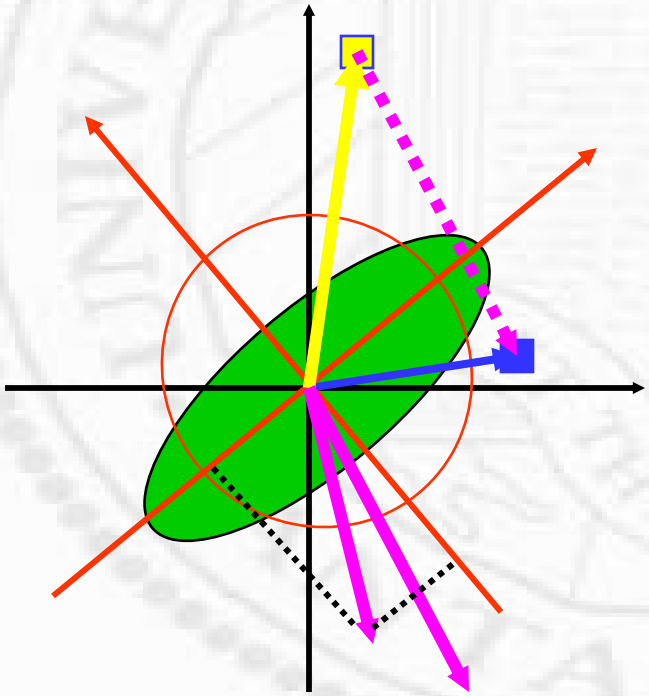
$$\frac{\mathbf{x}^T \mathbf{C}_{d\times d}^{-1} \mathbf{y}}{\sqrt{(\mathbf{x}^T \mathbf{C}_{d\times d}^{-1} \mathbf{x})(\mathbf{y}^T \mathbf{C}_{d\times d}^{-1} \mathbf{y})}} = \frac{(\boldsymbol{\Sigma}^{-1}\mathbf{U}^t\mathbf{x})^T (\boldsymbol{\Sigma}^{-1}\mathbf{U}^t\mathbf{y})}{\sqrt{\left[(\boldsymbol{\Sigma}^{-1}\mathbf{U}^t\mathbf{x})^T (\boldsymbol{\Sigma}^{-1}\mathbf{U}^t\mathbf{x})\right]\left[(\boldsymbol{\Sigma}^{-1}\mathbf{U}^t\mathbf{y})^T (\boldsymbol{\Sigma}^{-1}\mathbf{U}^t\mathbf{y})\right]}}$$

$$= \frac{\mathbf{v}_x{}^T \mathbf{v}_y}{\sqrt{(\mathbf{v}_x{}^T \mathbf{v}_x)(\mathbf{v}_y{}^T \mathbf{v}_y)}} = \frac{\mathbf{v}_x \cdot \mathbf{v}_y}{\sqrt{(\mathbf{v}_x \cdot \mathbf{v}_x)(\mathbf{v}_y \cdot \mathbf{v}_y)}}$$

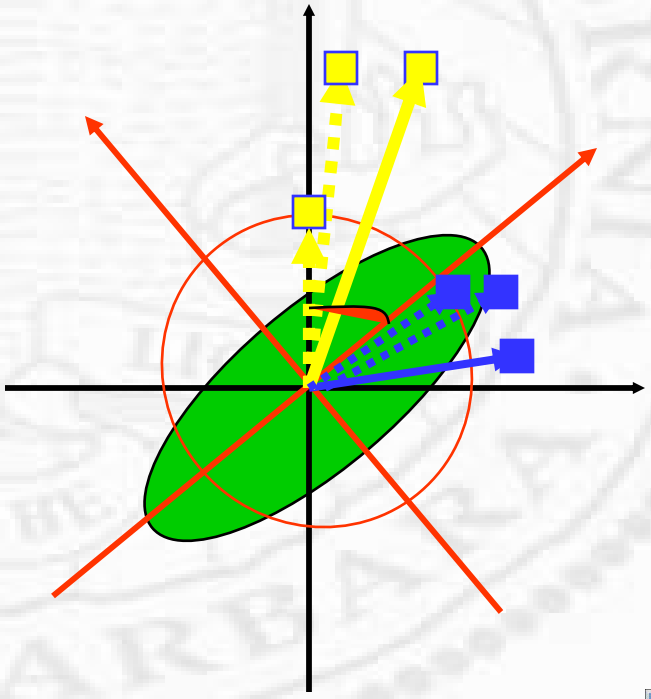
- Assume zero centered
- Range from -1 to 1
- Similarity measure based on angle between normalized vector

Comparison

- Mahalanobis distance
  - Distance measure



- Correlation coefficients
  - Angle measure



Popular Formulations

- AX=B

$$\|\mathbf{Ax}-\mathbf{b}\|$$
$$\Rightarrow \|\mathbf{UDV}^T\mathbf{x}-\mathbf{b}\|$$
$$\Rightarrow \|\mathbf{DV}^T\mathbf{x}-\mathbf{U}^T\mathbf{b}\|$$
$$\Rightarrow \|\mathbf{Dy}-\mathbf{b}'\|$$

- AX=0 |X|=1

$$\|\mathbf{Ax}\| \quad \|\mathbf{x}\|=1$$
$$\Rightarrow \|\mathbf{UDV}^T\mathbf{x}\| \quad \|\mathbf{x}\|=1$$
$$\Rightarrow \|\mathbf{DV}^T\mathbf{x}\| \quad \|\mathbf{x}\|=1$$
$$\Rightarrow \|\mathbf{Dy}\| \quad \|\mathbf{y}\|=1$$
$$\Rightarrow \mathbf{y}=(0,0,\cdots,1)^T$$

- One can solve AX=B with normal equation
- It can be faster, but not as general as SVD in handling other popular formulations

Reminder

- $A=UDV^T$
- A and U have the same column space
- A and V^T have the same row space
- U and V are orthogonal matrices that do not change norm

$$\mathbf{A} = \mathbf{UDV}^T = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{u_1} & \mathbf{u}_2 & \mathbf{u}_3 & \mathbf{u}_4 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & 0 & \\ & & & 0 \end{bmatrix} \begin{bmatrix} \cdots & \cdots & \mathbf{v}_1^T & \cdots \\ \cdots & \cdots & \mathbf{v}_2^T & \cdots \\ \cdots & \cdots & \mathbf{v}_3^T & \cdots \\ \cdots & \cdots & \mathbf{v}_4^T & \cdots \end{bmatrix}$$

$$= \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ d_1\mathbf{u_1} & d_2\mathbf{u}_2 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} \cdots & \cdots & \mathbf{v}_1^T & \cdots \\ \cdots & \cdots & \mathbf{v}_2^T & \cdots \\ \cdots & \cdots & \mathbf{v}_3^T & \cdots \\ \cdots & \cdots & \mathbf{v}_4^T & \cdots \end{bmatrix}$$

$$= \begin{bmatrix} \cdots & \cdots & d_1u_{11}\mathbf{v}_1^T + d_2u_{21}\mathbf{v}_2^T & \cdots \\ \cdots & \cdots & d_1u_{12}\mathbf{v}_1^T + d_2u_{22}\mathbf{v}_2^T & \cdots \\ \cdots & \cdots & d_1u_{13}\mathbf{v}_1^T + d_2u_{23}\mathbf{v}_2^T & \cdots \\ \cdots & \cdots & d_1u_{14}\mathbf{v}_1^T + d_2u_{24}\mathbf{v}_2^T & \cdots \end{bmatrix}$$

Row view

Column view

$$= \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ d_1v_{11}^T\mathbf{u_1} + d_2v_{21}^T\mathbf{u}_2 & d_1v_{12}^T\mathbf{u_1} + d_2v_{22}^T\mathbf{u}_2 & d_1v_{13}^T\mathbf{u_1} + d_2v_{23}^T\mathbf{u}_2 & d_1v_{14}^T\mathbf{u_1} + d_2v_{24}^T\mathbf{u}_2 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

Popular Formulations

- Min |AX|, subject to |X|=1 CX=0
  - Certain property of H, P, F is known to be true
- X lies in the orthogonal complement of row space of C
- $C=UDV^T$ (rank of C is *r,* D *has r nonzero*)
- The first *r* rows of V^T defines the row space of C
- The remaining *n–r* rows of V^T defines the orthogonal complement C*
- Or X lies in the column space of C^{*T}
- $X= C^{*T}X'$,$|X| = |C^{*T}X'| = |X'|$
- $|AC^*X'|\ |X'| = 1$

Popular Formulations

- Min |AX|, subject to |X|=1 X=GX'
  - X is in certain subspace
  - X is in the column space of G (with r<n columns)
- G= UDV^T (D has r nonzero entries)
- G has the same column space as U or U' (the first r columns of U)
- X=GX' is the same as X=U'X'
- |AU'X'| |X'| = 1, X=U'X'

Extra

- X=UX', then |X|=1 implies |X'|=1 if U has orthogonal columns

$$\mathbf{X} = \mathbf{UX}' = \begin{bmatrix} | & | & | & | & | \\ \mathbf{u}_1 & \cdots & \mathbf{u}_r & 0 & \cdots & 0 \\ | & | & | & | & | \end{bmatrix} \begin{bmatrix} x_1' \\ \\ \\ x_n' \end{bmatrix}$$

$$= x_1'\mathbf{u}_1 + x_2'\mathbf{u}_2 + \cdots + x_r'\mathbf{u}_r$$

$$|\mathbf{UX}'|^2 = (x_1'\mathbf{u}_1 + x_2'\mathbf{u}_2 + \cdots + x_r'\mathbf{u}_r)^T (x_1'\mathbf{u}_1 + x_2'\mathbf{u}_2 + \cdots + x_r'\mathbf{u}_r)$$

$$= x_1'^2 + \cdots + x_r'^2 = \mathbf{x}'^2$$

Recall Camera Calibration

$$\mathbf{x}_{real} = \begin{bmatrix} k_u & 0 & u_o \\ 0 & k_v & v_o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{bmatrix} \mathbf{X}_{world}$$

$$\begin{bmatrix} k_u & 0 & u_o \\ 0 & k_v & v_o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 & t_x \\ \mathbf{r}_2 & t_y \\ \mathbf{r}_3 & t_z \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \alpha_u & 0 & u_o & 0 \\ 0 & \alpha_v & v_o & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 & t_x \\ \mathbf{r}_2 & t_y \\ \mathbf{r}_3 & t_z \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_u \mathbf{r}_1 + u_o \mathbf{r}_3 & \alpha_u t_x + u_o t_z \\ \alpha_v \mathbf{r}_2 + v_o \mathbf{r}_3 & \alpha_v t_y + v_o t_z \\ \mathbf{r}_3 & t_z \end{bmatrix}$$

$$= \begin{bmatrix} \mathbf{q}_1^T & q_{14} \\ \mathbf{q}_2^T & q_{24} \\ \mathbf{q}_3^T & q_{34} \end{bmatrix}$$

$$|\mathbf{q}_3| = 1, (\mathbf{q}_1 \times \mathbf{q}_3) \cdot (\mathbf{q}_2 \times \mathbf{q}_3) = 0$$

$$\mathbf{Ap} = \mathbf{0}$$

$$\min \|\mathbf{Ap}\|^2 \text{ subject to } |\mathbf{q}_3| = 1$$

Popular Formulation

- Min |AX| |CX|=1
- C= UDV^T -> |UDV^T X|=1, |DV^T X|=1
- X' =V^T X -> |AVX'| (|A'X'|)|DX'|=1
- D has r nonzero columns and s zero columns, decompose A' accordingly into A_1' (r columns), A_2' (s columns)
- A'=[A_1'|A_2'] , X'=[X_1'^{T}, X_2'^{T}]^T-> |A_1'X_1'+A_2'X_2'| |DX_1'| = 1
- X_2' = -A_2'$^+$$A_1$'$X_1$'
- | A_1'X_1'-A_2'A_2'$^+$$A_1$'$X_1$' | |D$X_1$'| = 1
- X"= DX_1'
- | A"X"| |X"|=1

Nonlinear Methods

- Generally, much harder
- There are different classes of nonlinearity
- Global vs. local search
- Local search needs good initial guess
- Popular approaches in CV
  - Linear method to provide initial guess
  - Nonlinear method to polish final results
  - Most popular – Levenberg-Marquart

Options

- First-order approximation

$$f(x_o+\delta)=f(x_o)+\delta f'(x_o)+\cdots=y$$

$$\delta=-\frac{f(x_o)-y}{f'(x_o)}$$

- Second-order approximation

$$f(x_o+\delta)=f(x_o)+\delta f'(x_o)+\frac{\delta^2}{2}f''(x_o)\cdots$$

$$f'(x_o+\delta)\approx f'(x_o)+\delta f''(x_o)=0$$

$$\Rightarrow \delta=-\frac{f'(x_o)}{f''(x_o)}$$

- Gradient descent

$$\delta=-\lambda f'$$

$$f(x_o+\delta)\approx f(x_o)-f'(x_o)\delta$$

Newton's Method for Root Finding: 1st order Approximation

- Find root to a scalar equation
  - Iterative solution
  - Using Taylor series expansion

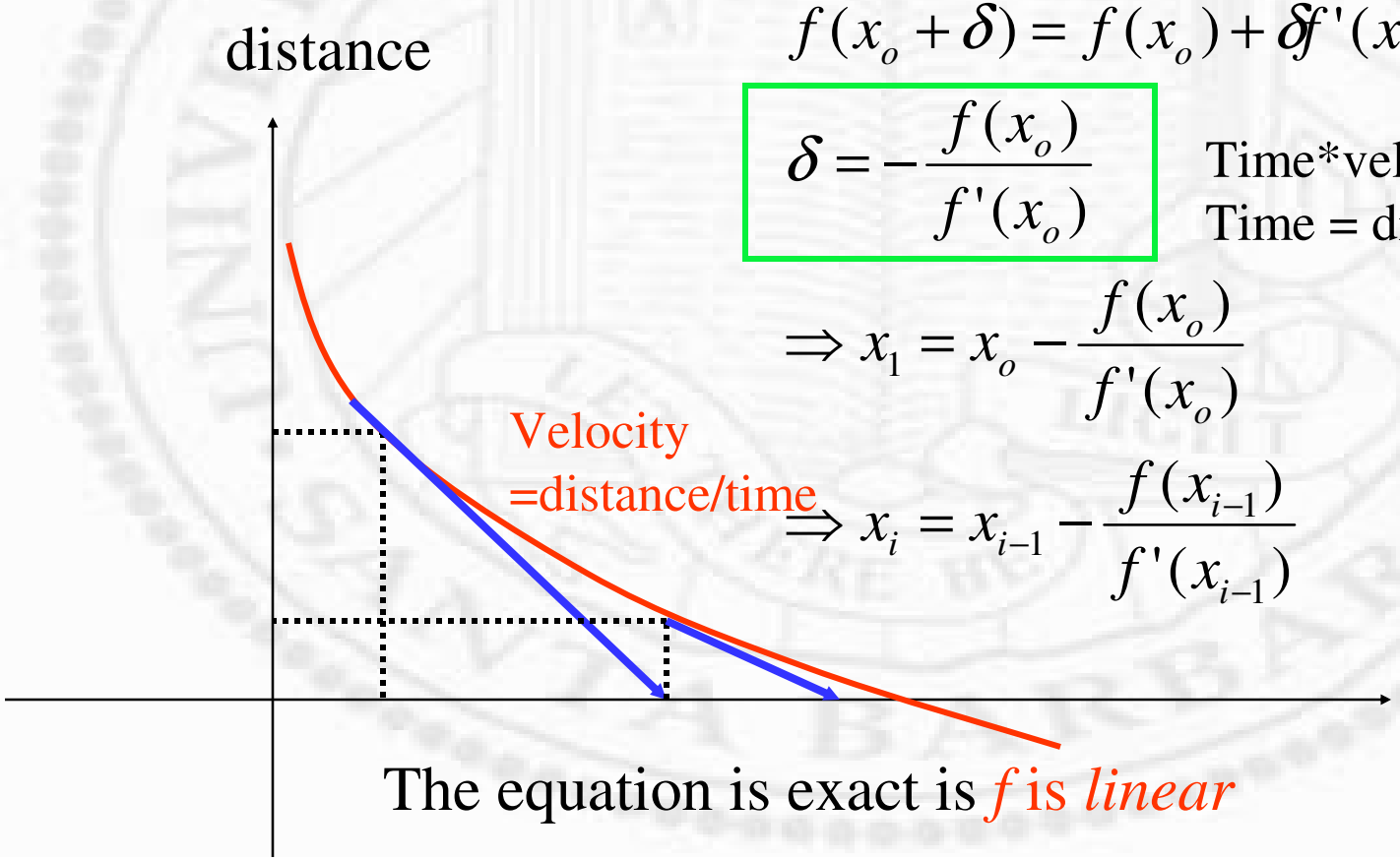
$$f(x_o+\delta) = f(x_o) + \delta f'(x_o) + \cdots = 0$$

$$\delta = -\frac{f(x_o)}{f'(x_o)}$$

Time*velocity=distance
Time = distance/velocity

$$\Rightarrow x_1 = x_o - \frac{f(x_o)}{f'(x_o)}$$

$$\Rightarrow x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}$$



The equation is exact is *f* is *linear*

Generalization #1

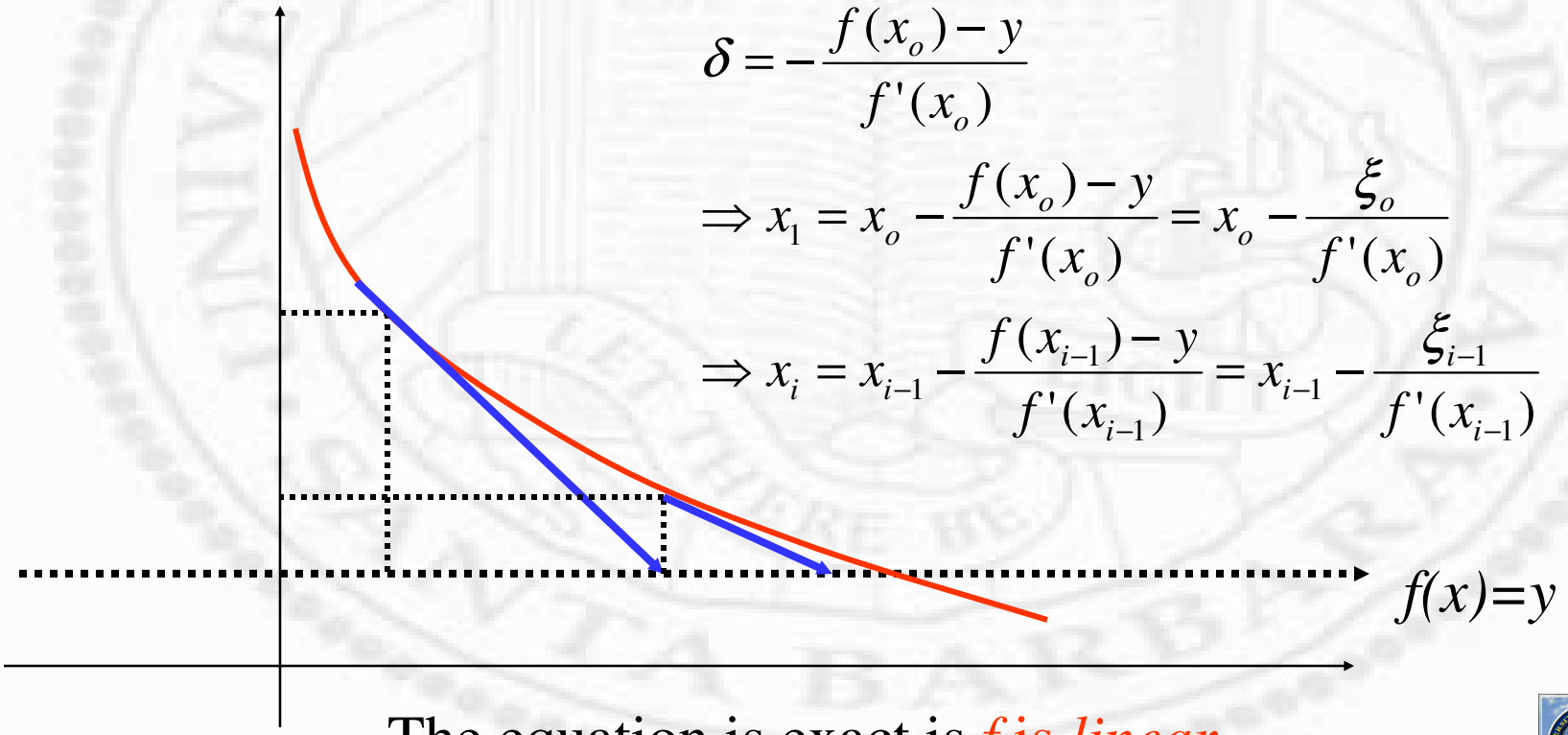
- Find particular level to a scalar eq.
  - *x*: parameter, *f*: function, *y*: data

$$f(x_o+\delta)=f(x_o)+\delta f'(x_o)+\cdots=y$$

$$\delta=-\frac{f(x_o)-y}{f'(x_o)}$$

$$\Rightarrow x_1=x_o-\frac{f(x_o)-y}{f'(x_o)}=x_o-\frac{\xi_o}{f'(x_o)}$$

$$\Rightarrow x_i=x_{i-1}-\frac{f(x_{i-1})-y}{f'(x_{i-1})}=x_{i-1}-\frac{\xi_{i-1}}{f'(x_{i-1})}$$

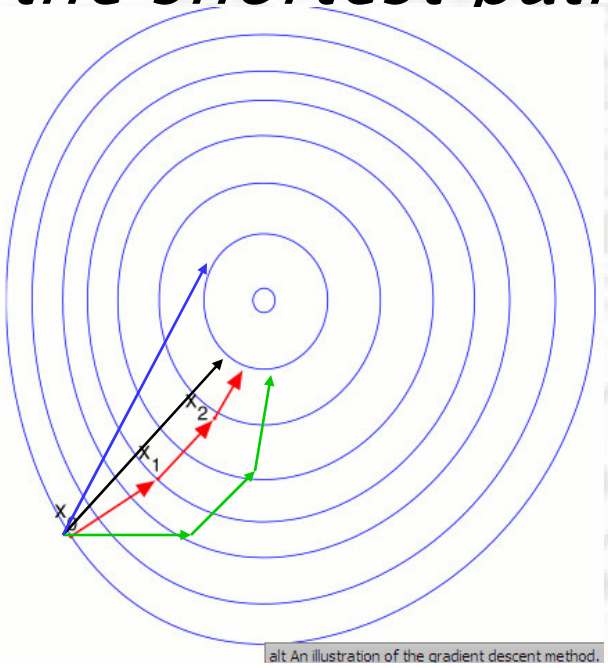


The equation is exact is *f* is *linear*

Generalization #2

- Vector parameters
  - Find lowest temperature in a plane
  - Generally speaking, the problem is not fully constrained, as there are many ways to get from $f(\mathbf{x}_o)$ to $f(\mathbf{x})=y$
  - *Constraint: the shortest path*

$$\nabla f(\mathbf{x}_o) = \left(\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y} \right)^T$$



alt An illustration of the gradient descent method.

Generalization #2 (cont)

- Goal:

$$\min\|\Delta\| = \min \Delta^T \Delta$$

- Subject to:

$$f(\mathbf{x}_o + \Delta) = f(\mathbf{x}_o) + \nabla f(\mathbf{x}_o)^T \Delta + \cdots = y$$

$$\nabla f(\mathbf{x}_o)^T \Delta = -(f(\mathbf{x}_o) - y) = -\xi$$

$$e = \Delta^T \Delta + 2\lambda(\nabla f^T \Delta + \xi)$$

$$\frac{\partial e}{\partial \Delta} = \Delta^T + \lambda \nabla f^T = 0 \Rightarrow \Delta^T = -\lambda \nabla f^T$$

$$\frac{\partial e}{\partial \lambda} = \nabla f^T \Delta + \xi = 0 \Rightarrow \nabla f^T \lambda \nabla f + \xi = 0, \lambda = -\frac{\xi}{\nabla f^T \nabla f}$$

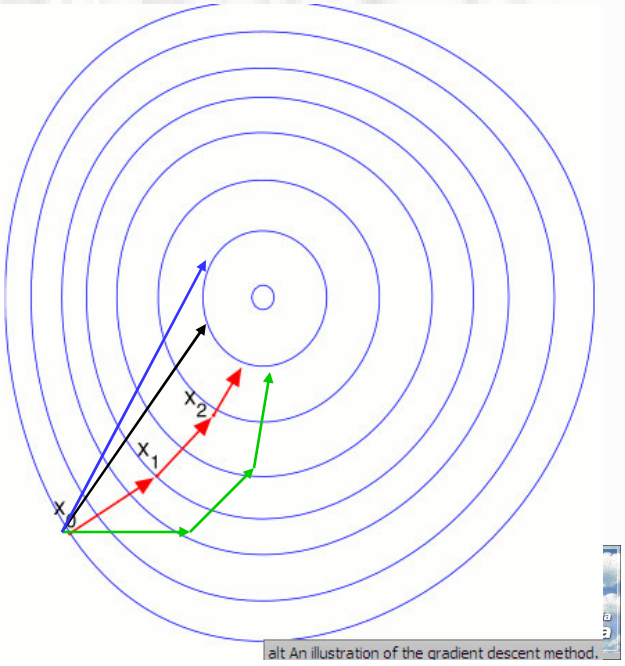
$$\Delta = -\frac{\xi}{\nabla f^T \nabla f} \nabla f$$

$$\mathbf{x}_1 = \mathbf{x}_o - \frac{\xi_o}{\nabla f^T(\mathbf{x}_o) \nabla f(\mathbf{x}_o)} \nabla f(\mathbf{x}_o)$$

$$\mathbf{x}_i = \mathbf{x}_{i-1} - \frac{\xi_{i-1}}{\nabla f^T(\mathbf{x}_{i-1}) \nabla f(\mathbf{x}_{i-1})} \nabla f(\mathbf{x}_{i-1})$$

- Go in gradient direction (otherwise, you are not accomplishing anything)
- Go by distance/velocity

Exactly for linear error surfaces!!!



alt An illustration of the gradient descent method.

Generalization #3

- Composite error function: error function can be a vector and we seek to minimize the norm of the vector
  - Find highest wind velocity in a plane

$$f = \frac{1}{2}\sum_{i=1}^{m} f_i^{\,2}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m} f_i^{\,2}(x_1, \cdots, x_j, \cdots, x_n) = \mathbf{F}^T\mathbf{F}$$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}_{n\times 1} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_2} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}_{n\times m} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}_{m\times 1} = \mathbf{J}^T\mathbf{F}$$

Generalization #3 (cont)

- Goal:

$$\min\|\Delta\| = \min \Delta^T \Delta$$

- Subject to:

$$f(\mathbf{x}_o + \Delta) = f(\mathbf{x}_o) + \nabla f(\mathbf{x}_o)^T \Delta + \cdots = y$$

$$\nabla f(\mathbf{x}_o)^T \Delta = -(f(\mathbf{x}_o) - y) = -\xi$$

$$e = \Delta^T \Delta + 2\lambda(\nabla f^T \Delta + \xi)$$

$$\frac{\partial e}{\partial \Delta} = \Delta^T + \lambda \nabla f^T = 0 \Rightarrow \Delta^T = -\lambda \nabla f^T$$

$$\frac{\partial e}{\partial \lambda} = \nabla f^T \Delta + \xi = 0 \Rightarrow \nabla f^T \lambda \nabla f + \xi = 0, \lambda = -\frac{\xi}{\nabla f^T \nabla f}$$

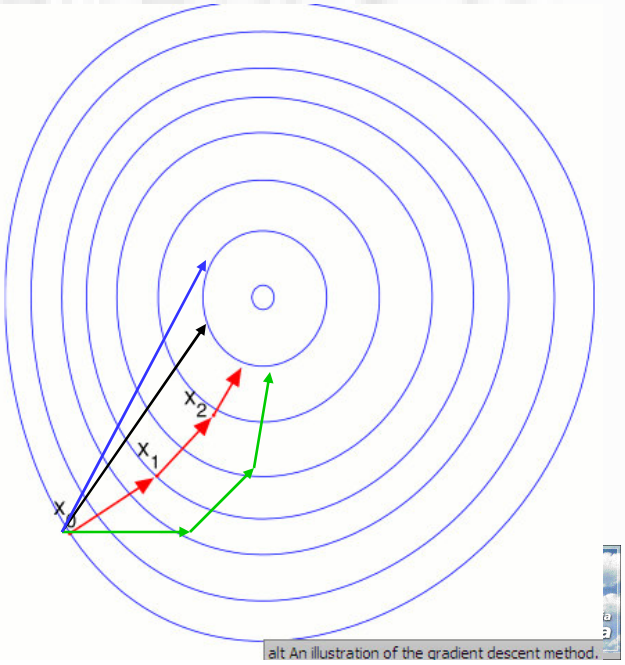
$$\Delta = -\frac{\xi}{\nabla f^T \nabla f} \nabla f$$

$$\mathbf{x}_1 = \mathbf{x}_o - \frac{\xi_o}{\nabla f^T(\mathbf{x}_o) \nabla f(\mathbf{x}_o)} \nabla f(\mathbf{x}_o)$$

$$\mathbf{x}_i = \mathbf{x}_{i-1} - \frac{\xi_o}{\nabla f^T(\mathbf{x}_{i-1}) \nabla f(\mathbf{x}_{i-1})} \nabla f(\mathbf{x}_{i-1})$$

- Nothing has changed, except $\nabla f = J^T F$

Exactly for linear error surfaces!!!



alt An illustration of the gradient descent method.

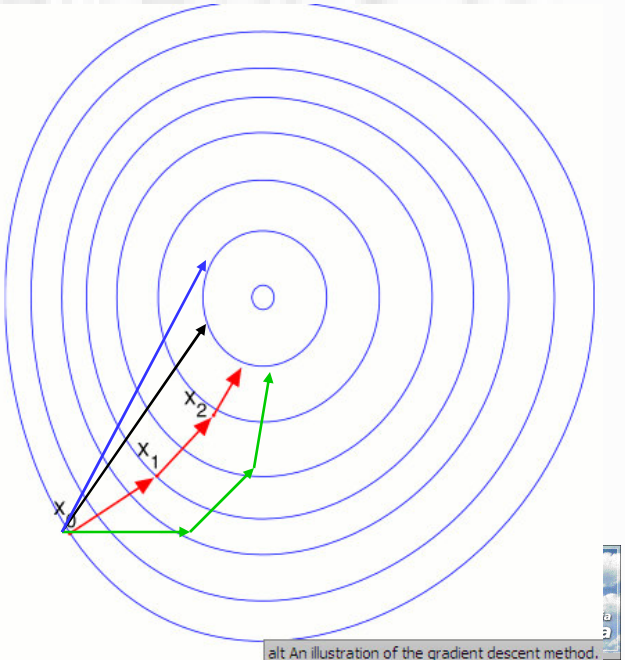
Generalization #4

- Vector error function, vector parameters
  - If m < n, not enough constraints
  - If m=n, equally constrained
  - If m>n overconstrained

$$\mathbf{f}(\mathbf{x}_o + \boldsymbol{\Delta}) = \mathbf{f}(\mathbf{x}_o) + \mathbf{J}\boldsymbol{\Delta} + \cdots = \mathbf{y}$$

$$\mathbf{J}\boldsymbol{\Delta} = -(\mathbf{f}(\mathbf{x}_o) - \mathbf{y}) = -\boldsymbol{\xi}$$

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$



alt An illustration of the gradient descent method.

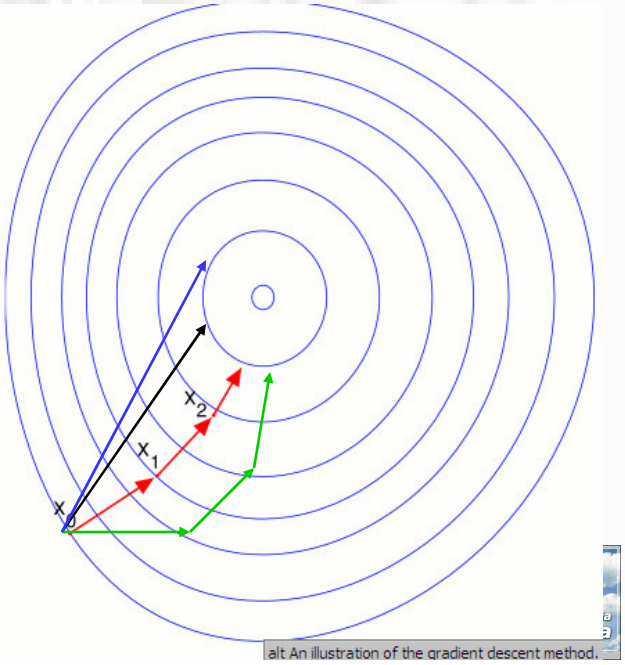
Generalization #4

$$\mathbf{f}(\mathbf{x}_o + \boldsymbol{\Delta}) = \mathbf{f}(\mathbf{x}_o) + \mathbf{J}\boldsymbol{\Delta} + \cdots = \mathbf{y}$$

$$\mathbf{J}\boldsymbol{\Delta} = -(\mathbf{f}(\mathbf{x}_o) - \mathbf{y}) = -\boldsymbol{\xi}$$

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_n \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}\Delta + \cdots + \frac{\partial f_1}{\partial x_n}\Delta x_n \\ \frac{\partial f_2}{\partial x_1}\Delta + \cdots + \frac{\partial f_2}{\partial x_n}\Delta x_n \\ \vdots \\ \frac{\partial f_m}{\partial x_1}\Delta + \cdots + \frac{\partial f_m}{\partial x_n}\Delta x_n \end{bmatrix} = \begin{bmatrix} \Delta f_1 \\ \Delta f_2 \\ \vdots \\ \Delta f_m \end{bmatrix}$$

- Total change is sum of component changes
- Total change should match the requires decrease



alt An illustration of the gradient descent method.

Generalization #4 (cont)

- Goal: $\min\|\Delta\| = \min \Delta^T \Delta$
- Subject to: $\mathbf{f}(\mathbf{x}_o + \Delta) = \mathbf{f}(\mathbf{x}_o) + \mathbf{J}\Delta + \cdots = \mathbf{y}$

$$\mathbf{J}\Delta = -(\mathbf{f}(\mathbf{x}_o) - \mathbf{y}) = -\xi$$

$$e = \Delta^T \Delta + 2\lambda^T (\mathbf{J}\Delta + \xi)$$

$$\frac{\partial e}{\partial \Delta} = \Delta^T + \lambda^T \mathbf{J} = 0 \Rightarrow \Delta = -\mathbf{J}^T \lambda$$

$$\frac{\partial e}{\partial \lambda} = \mathbf{J}\Delta + \xi = 0 \Rightarrow -\mathbf{J}\mathbf{J}^T \lambda + \xi = 0, \lambda = (\mathbf{J}\mathbf{J}^T)^{-1}\xi$$

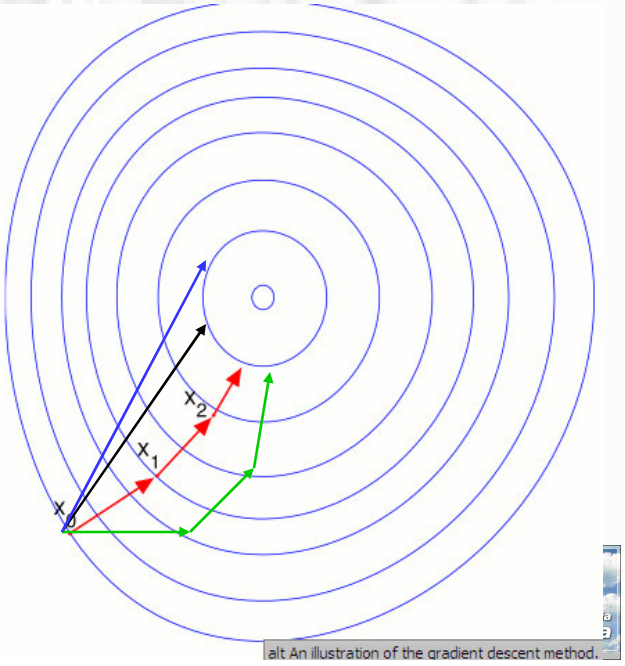
$$\Delta = -\mathbf{J}^T (\mathbf{J}\mathbf{J}^T)^{-1}\xi$$

$$\mathbf{x}_1 = \mathbf{x}_o - \mathbf{J}^T (\mathbf{J}\mathbf{J}^T)^{-1}\xi_o$$

$$\mathbf{x}_i = \mathbf{x}_{i-1} - \mathbf{J}^T (\mathbf{J}\mathbf{J}^T)^{-1}\xi_{i-1}$$

- Go in gradient direction (otherwise, you are not accomplishing anything
- Go by distance/velocity

Exact for linear error surfaces!!!



alt An illustration of the gradient descent method.

$$\mathbf{J}^T = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_2} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} = \begin{bmatrix} | & | & | & | \\ \nabla f_1 & \nabla f_2 & \vdots & \nabla f_m \\ | & | & | & | \end{bmatrix}_{n \times m}$$

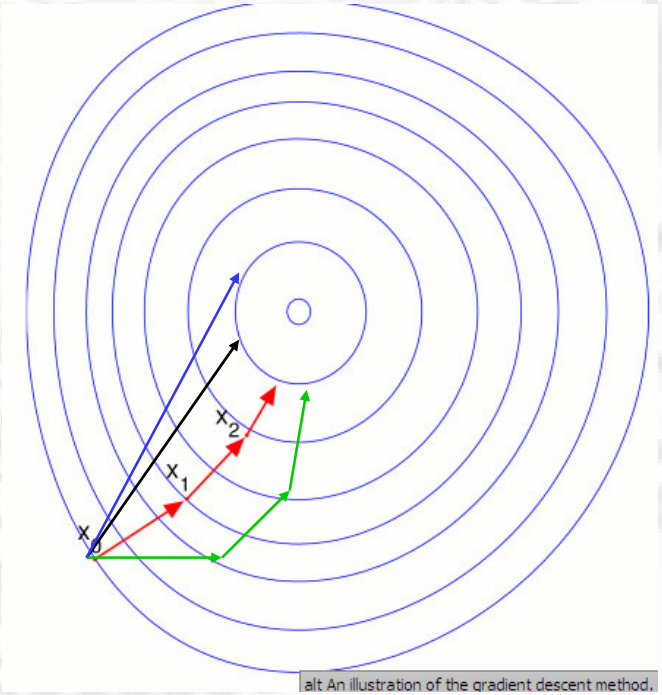
$$\mathbf{J}^T \boldsymbol{\lambda} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_2} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}_{n \times m} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{bmatrix}_{m \times 1} = \begin{bmatrix} | & | & | & | \\ \nabla f_1 & \nabla f_2 & \vdots & \nabla f_m \\ | & | & | & | \end{bmatrix}_{n \times m} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{bmatrix}_{m \times 1}$$

$$= \lambda_1 \nabla f_1 + \lambda_2 \nabla f_2 + \cdots \lambda_m \nabla f_m$$

Gradient directions

Sampson Error

- Assume that the error function is linear in the neighborhood
- No iteration



alt An illustration of the gradient descent method.

Essence

- Direction * change rate along the direction = error to zero out
- Direction: negative gradient direction
- Change rate: size of gradient
- First-order methods require only gradient, which is some form of Jacobian
- Second-order methods may require Hessian (harder to compute)

2^{nd}-Order Approximation

- Newton's method

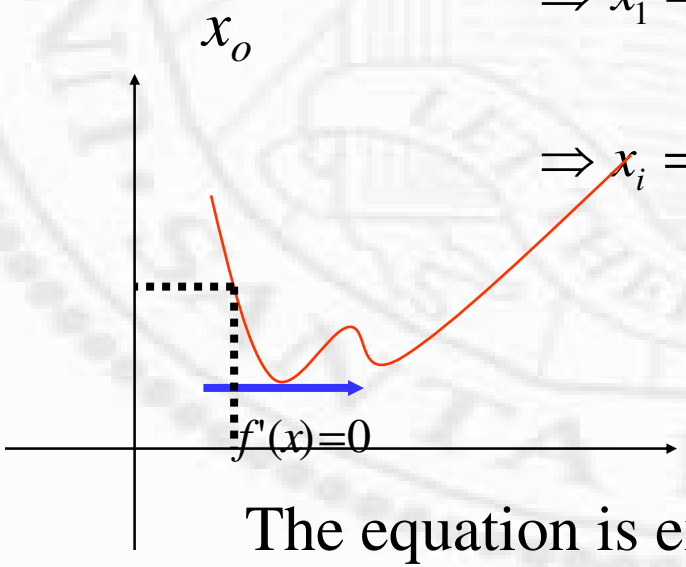
$$f(x_o+\delta) = f(x_o) + \delta f'(x_o) + \frac{\delta^2}{2} f''(x_o) \cdots$$

$$f'(x_o+\delta) \approx f'(x_o) + \delta f''(x_o) = 0$$

$$\Rightarrow \delta = -\frac{f'(x_o)}{f''(x_o)}$$

$$\Rightarrow x_1 = x_o - \frac{f'(x_o)}{f''(x_o)}$$

$$\Rightarrow x_i = x_{i-1} - \frac{f'(x_{i-1})}{f''(x_{i-1})}$$



The equation is exact is *f* is *quadratic*

Comparison

- 2nd–order
- 1st–order

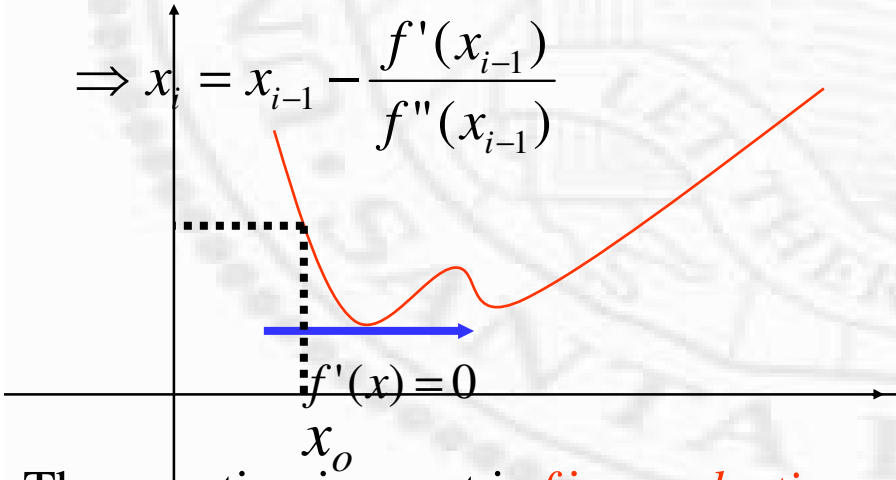
$$f(x_o+\delta) = f(x_o) + \delta f'(x_o) + \frac{\delta^2}{2} f''(x_o)\cdots$$

$$f'(x_o+\delta) = f'(x_o) + \delta f''(x_o) + \cdots = 0$$

$$\Rightarrow \delta = -\frac{f'(x_o)}{f''(x_o)}$$

$$\Rightarrow x_1 = x_o - \frac{f'(x_o)}{f''(x_o)}$$

$$\Rightarrow x_i = x_{i-1} - \frac{f'(x_{i-1})}{f''(x_{i-1})}$$



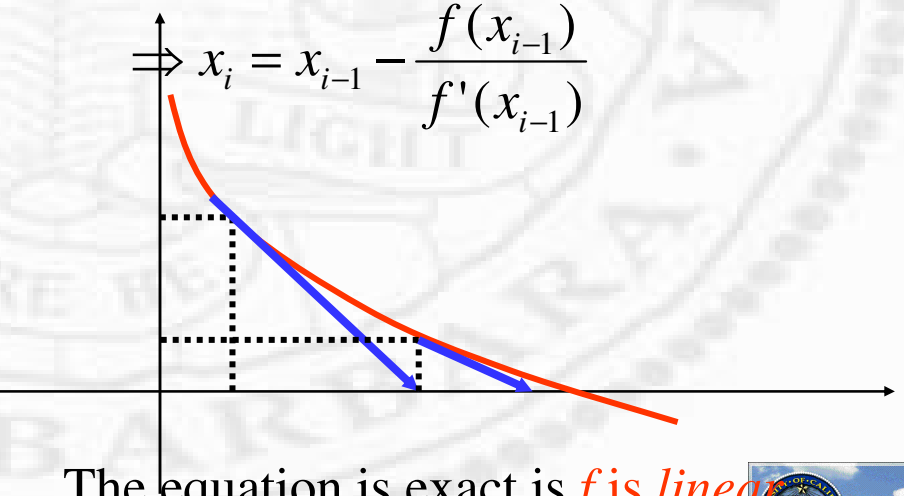
The equation is exact is *f* is *quadratic*

$$f(x_o+\delta) = f(x_o) + \delta f'(x_o) + \cdots = 0$$

$$\delta = -\frac{f(x_o)}{f'(x_o)}$$

$$\Rightarrow x_1 = x_o - \frac{f(x_o)}{f'(x_o)}$$

$$\Rightarrow x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}$$



The equation is exact is *f* is *linear*

Generalization

- The argument does no have to be a scalar
- Derivatives becomes partial derivatives
- The function does not have to be a singleton
- Another level of indirection

Generalization #1

- Vector parameters

$$f(\mathbf{x}_o+\boldsymbol{\Delta}) = f(\mathbf{x}_o)+\boldsymbol{\Delta}^T\nabla f(\mathbf{x}_o)+\boldsymbol{\Delta}^T\nabla^2 f(\mathbf{x}_o)\boldsymbol{\Delta}+\cdots$$

$$\nabla f(\mathbf{x}_o+\boldsymbol{\Delta}) = \nabla f(\mathbf{x}_o)+\nabla^2 f(\mathbf{x}_o)\boldsymbol{\Delta} = \mathbf{0}$$

Hessian matrix

$$\Rightarrow \nabla f(\mathbf{x}_o)+\nabla^2 f(\mathbf{x}_o)\boldsymbol{\Delta} = 0$$

$$\Rightarrow \boldsymbol{\Delta} = -(\nabla^2 f(\mathbf{x}_o))^{-1}\nabla f(\mathbf{x}_o)$$

$$\Rightarrow \mathbf{x}_1 = \mathbf{x}_o - (\nabla^2 f(\mathbf{x}_o))^{-1}\nabla f(\mathbf{x}_o)$$

$$\Rightarrow \mathbf{x}_i = \mathbf{x}_{i-1} - (\nabla^2 f(\mathbf{x}_{i-1}))^{-1}\nabla f(\mathbf{x}_{i-1})$$

$$\nabla^2 f = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \dfrac{\partial^2 f}{\partial x_k \partial x_l} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}_{n\times n}$$

$$\Delta^T{}_{1\times n}\nabla^2 f_{n\times n}\Delta_{n\times 1}$$

Generalization #2

- Vector error function, vector parameters
  - This is not so simple now

$$\mathbf{f}(\mathbf{x}_o+\Delta)=\mathbf{f}(\mathbf{x}_o)+\Delta^T\nabla\mathbf{f}(\mathbf{x}_o)+\Delta^T\nabla^2\mathbf{f}(\mathbf{x}_o)\Delta+\cdots$$

$$\nabla f(\mathbf{x}_o+\Delta)=\nabla f(\mathbf{x}_o)+\nabla^2 f(\mathbf{x}_o)\Delta=\mathbf{0}$$

$$\Rightarrow \nabla f(\mathbf{x}_o)+\nabla^2 f(\mathbf{x}_o)\Delta=0$$

$$\Rightarrow \Delta=-(\nabla^2 f(\mathbf{x}_o))^{-1}\nabla f(\mathbf{x}_o)$$

$$\Rightarrow \mathbf{x}_1=\mathbf{x}_o-(\nabla^2 f(\mathbf{x}_o))^{-1}\nabla f(\mathbf{x}_o)$$

$$\Rightarrow \mathbf{x}_i=\mathbf{x}_{i-1}-(\nabla^2 f(\mathbf{x}_{i-1}))^{-1}\nabla f(\mathbf{x}_{i-1})$$

Generalization #2

- Correct way: error function can be a vector, but we seek to minimize the norm of the vector
- Or there are multiple components that made up the error function

$$f = \frac{1}{2}\sum_{i=1}^{m} f_i^2(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m} f_i^2(x_1, \cdots, x_j, \cdots, x_n) = \mathbf{F}^T\mathbf{F}$$

Generalization #2

- Composite error function

$$f = \frac{1}{2}\sum_{i=1}^{m} f_i^{\,2}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m} f_i^{\,2}(x_1, \cdots, x_j, \cdots, x_n)$$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix}_{n\times 1} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_2} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial f_1}{\partial x_n} & \frac{\partial f_2}{\partial x_n} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}_{n\times m} \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix}_{m\times 1} = \mathbf{J}^T\mathbf{F}$$

Composite f function

$$\nabla^2 f = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \dfrac{\partial^2 f}{\partial x_k \partial x_l} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \displaystyle\sum_{i=1}^{m} f_{i,l} f_{i,k} + \sum_{i=1}^{m} f_i f_{i,kl} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}$$

$$= \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \displaystyle\sum_{i=1}^{m} f_{i,l} f_{i,k} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} + \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \displaystyle\sum_{i=1}^{m} f_i f_{i,kl} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}$$

$$= \mathbf{J}^T \mathbf{J} + \sum_{i=1}^{m} f_i \nabla^2 f_i \approx \mathbf{J}^T \mathbf{J}$$

Exact if f_i's are linear and f is quadratic

$$\frac{\partial f}{\partial x_k} = \sum_{i=1}^{m} f_i f_{i,k}, \quad \frac{\partial^2 f}{\partial x_k \partial x_l} = \sum_{i=1}^{m} f_{i,l} f_{i,k} + \sum_{i=1}^{m} f_i f_{i,kl}$$

Composite f function

$$\nabla^2 f = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \dfrac{\partial^2 f}{\partial x_k \partial x_l} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \displaystyle\sum_{i=1}^{m} f_{i,l} f_{i,k} + \sum_{i=1}^{m} f_i f_{i,kl} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}$$

$$= \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \displaystyle\sum_{i=1}^{m} f_{i,l} f_{i,k} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} + \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \displaystyle\sum_{i=1}^{m} f_i f_{i,kl} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix}$$

Fake Hessian with Jacobian only

$$= \mathbf{J}^T \mathbf{J} + \sum_{i=1}^{m} f_i \nabla^2 f_i \approx \mathbf{J}^T \mathbf{J}$$

Exact if f_i's are linear and f is quadratic

$$\frac{\partial f}{\partial x_k} = \sum_{i=1}^{m} f_i f_{i,k}, \quad \frac{\partial^2 f}{\partial x_k \partial x_l} = \sum_{i=1}^{m} f_{i,l} f_{i,k} + \sum_{i=1}^{m} f_i f_{i,kl}$$

Optimization in nD with Composite Cost Function

- Newton's method

$$f(\mathbf{x}_o+\Delta)=f(\mathbf{x}_o)+\Delta^T\nabla f(\mathbf{x}_o)+\Delta^T\nabla^2 f(\mathbf{x}_o)\Delta+\cdots$$

$$\nabla f(\mathbf{x}_o+\Delta)\approx\nabla f(\mathbf{x}_o)+\nabla^2 f(\mathbf{x}_o)\Delta=0$$

$$\Rightarrow\nabla f(\mathbf{x}_o)+\nabla^2 f(\mathbf{x}_o)\Delta=0$$

$$\Rightarrow\Delta=-(\nabla^2 f(\mathbf{x}_o))^{-1}\nabla f(\mathbf{x}_o)$$

$$\Rightarrow\mathbf{x}_1=\mathbf{x}_o-(\nabla^2 f(\mathbf{x}_o))^{-1}\nabla f(\mathbf{x}_o)$$

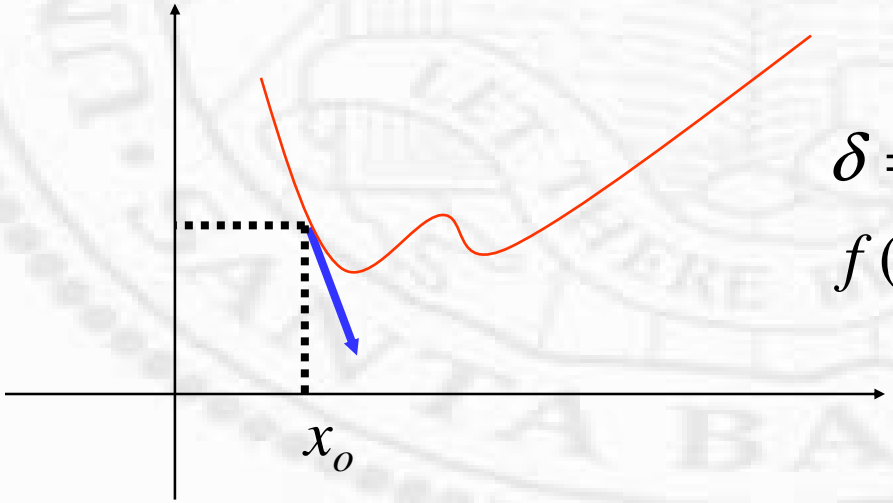
$$\Rightarrow\mathbf{x}_i=\mathbf{x}_{i-1}-(\nabla^2 f(\mathbf{x}_{i-1}))^{-1}\nabla f(\mathbf{x}_{i-1})$$

$$=\mathbf{x}_i=\mathbf{x}_{i-1}-(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{F}$$

Newton-Gauss methods
Faked Hessian

Gradient Descent

- Gradient descent
  - We do not know in general how much to proceed
  - Go along a certain direction (if $f'>0$, then left, if $f'<0$, then right) by some length (if function is decreasing)



$$\delta = -\lambda f'$$

$$f(x_o + \delta) \approx f(x_o) - f'(x_o)\delta$$

Comparison

- Newton's method

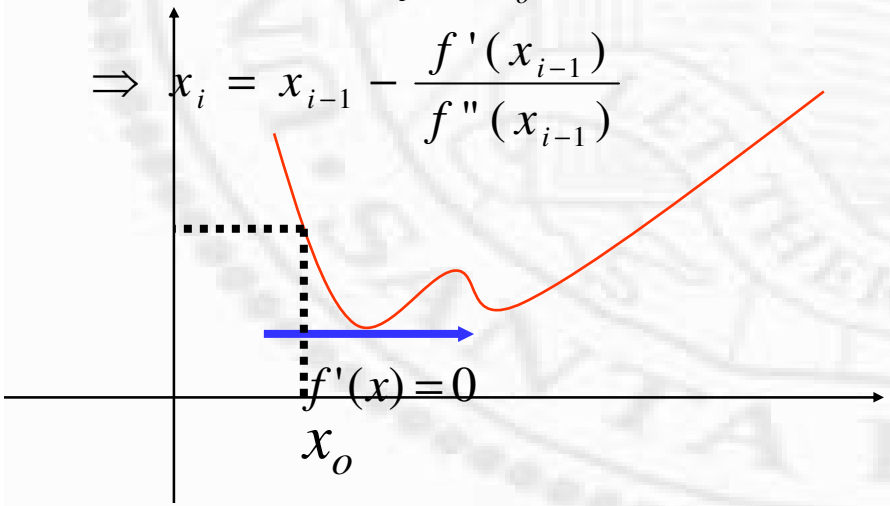
$$f(x_o+\delta) = f(x_o) + \delta f'(x_o) + \frac{\delta^2}{2} f''(x_o)\cdots$$

$$\frac{df}{d\delta} = f'(x_o) + \delta f''(x_o) = 0$$

$$\Rightarrow \delta = -\frac{f'(x_o)}{f''(x_o)}$$

$$\Rightarrow x_1 = x_o - \frac{f'(x_o)}{f''(x_o)}$$

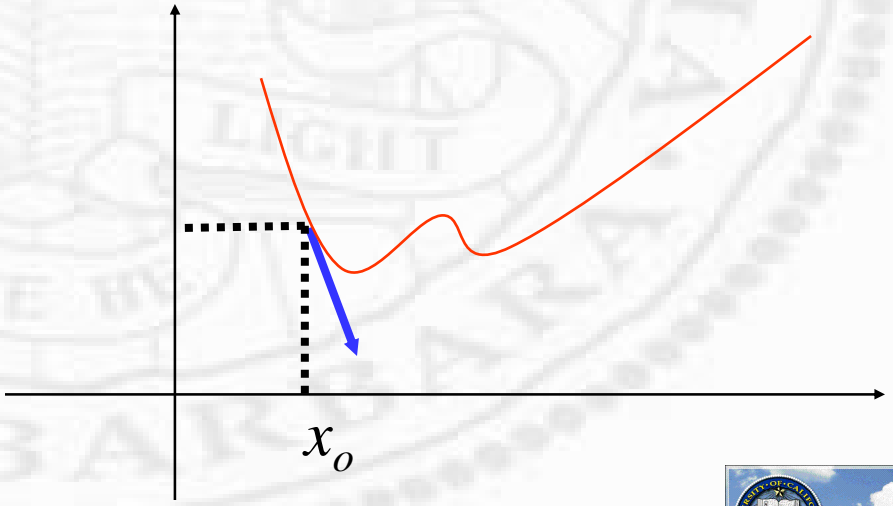
$$\Rightarrow x_i = x_{i-1} - \frac{f'(x_{i-1})}{f''(x_{i-1})}$$



- Gradient descent

$$\delta = -\lambda f'$$

$$f(x_o+\delta) \approx f(x_o) - f'(x_o)\delta$$



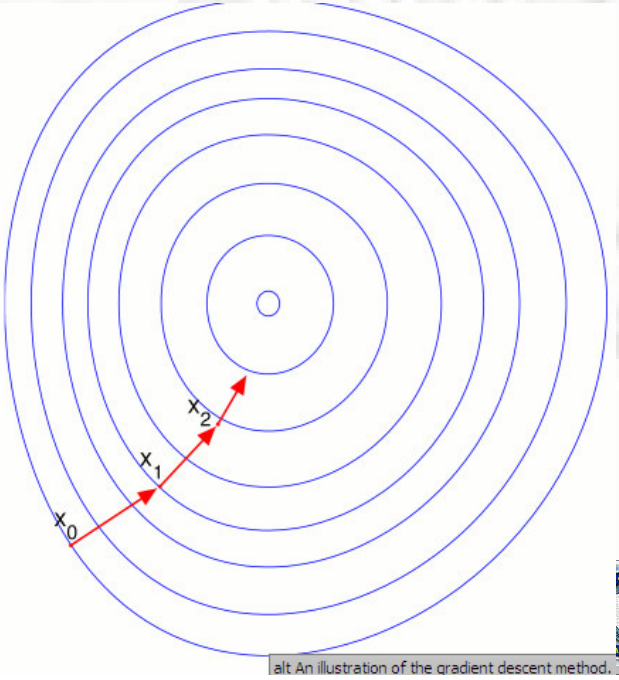
Generalization #1

- Optimization in nD

$$\Delta = \lambda \nabla f(\mathbf{x}_o)$$

$$f(\mathbf{x}+\Delta) \approx f(\mathbf{x}_o) - \Delta^T \nabla f(\mathbf{x}_o)$$

$$f(\mathbf{x}) \approx f(\mathbf{x}_o) - \lambda \nabla f(\mathbf{x}_o)^T \nabla f(\mathbf{x}_o)$$



alt An illustration of the gradient descent method.

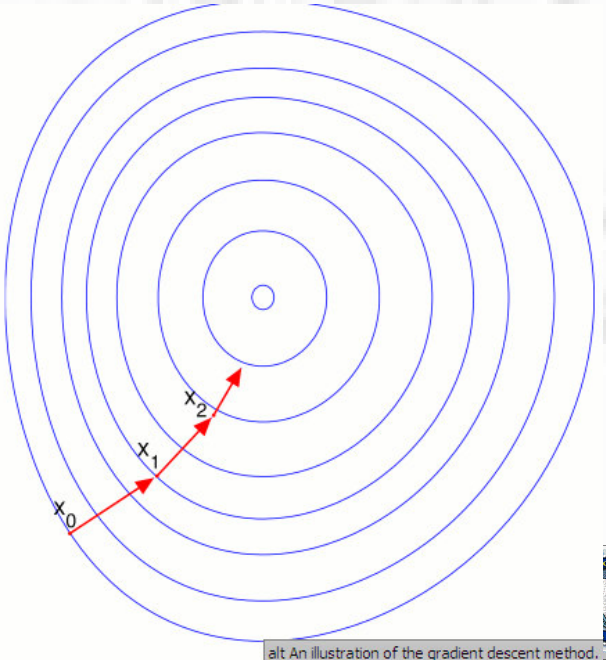
Generalization #2

- Composite Cost Function

$$\Delta = \lambda \mathbf{J}^T \mathbf{F}$$

$$f(\mathbf{x}+\Delta) \approx f(\mathbf{x}_o) - \Delta^T \nabla f(\mathbf{x}_o)$$

$$= f(\mathbf{x}_o) - \lambda \mathbf{F}^T \mathbf{J}\mathbf{J}^T \mathbf{F}$$



alt An illustration of the gradient descent method.

Levenberg-Marquardt

- A compromise of Newton (Newton-Gauss) and gradient descent
- If error goes down, Newton (Newton-Gauss) is working
  - Reduce λ to reduce gradient descent
- If error goes up, Newton (Newton-Gauss) is not working
  - Increase λ

$$\mathbf{x}_i = \mathbf{x}_{i-1} - (\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I})^{-1}\mathbf{J}^T\mathbf{F}$$

$$\mathbf{x}_i = \mathbf{x}_{i-1} - (\mathbf{J}^T\mathbf{J} + \lambda diag(\mathbf{J}^T\mathbf{J}))^{-1}\mathbf{J}^T\mathbf{F}$$

Graphical Interpretation

$$\mathbf{x}_i = \mathbf{x}_{i-1} - (\mathbf{J}^T\mathbf{J} + \lambda diag(\mathbf{J}^T\mathbf{J}))^{-1}\mathbf{J}^T\mathbf{F}$$

$$\mathbf{J}^T\mathbf{J} \approx \nabla^2 f = \begin{bmatrix} \cdots & \cdots & \cdots \\ \cdots & \dfrac{\partial^2 f}{\partial x_k \partial x_l} & \cdots \\ \cdots & \cdots & \cdots \end{bmatrix} \backslash$$

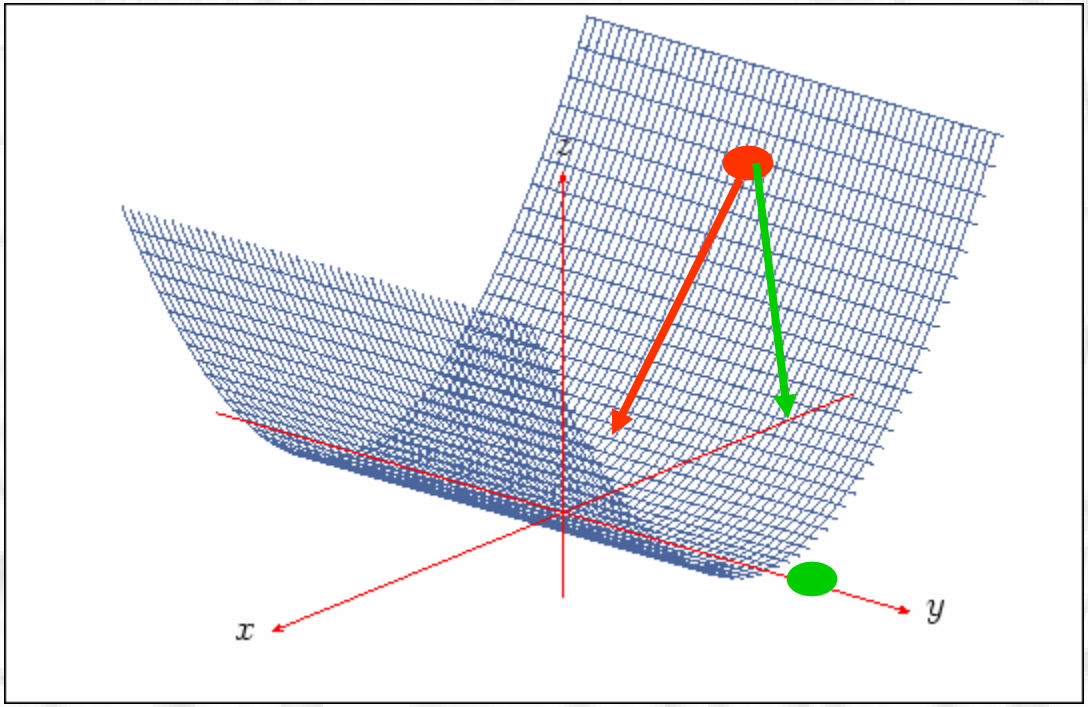
$$\mathbf{J}^T\mathbf{F} = \nabla f = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix}$$

$$diag(\mathbf{J}^T\mathbf{J}) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & & & \\ & \dfrac{\partial^2 f}{\partial x_2^2} & & \\ & & \ddots & \\ & & & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

- When λ is large
- J^TJ (Hessian) is wasted

Graphical Interpretation

- Penalize the search direction with large curvature
- Why?



Problems and Their Nonlinear Formulations – Reprojection Error

- Planar Homography

$$\hat{\mathbf{x}}' = \hat{\mathbf{H}}\hat{\mathbf{x}}$$

$$dist(\hat{\mathbf{x}}' - \mathbf{x}')^2 + dist(\hat{\mathbf{x}} - \mathbf{x})^2$$

- Camera Calibration

$$\hat{\mathbf{x}} = \hat{\mathbf{P}}\hat{\mathbf{X}}$$

$$dist(\hat{\mathbf{x}} - \mathbf{x})^2 + dist(\hat{\mathbf{X}} - \overline{\mathbf{X}})^2$$

$$\hat{\mathbf{x}} = \hat{\mathbf{P}}\overline{\mathbf{X}}$$

$$dist(\hat{\mathbf{x}} - \mathbf{x})^2$$

- Fundamental matrix

$$\mathbf{x}'\mathbf{F}\mathbf{x} = 0$$

$$dist(\mathbf{x} - \hat{\mathbf{P}}_o\hat{\mathbf{X}})^2 + dist(\mathbf{x}' - \hat{\mathbf{P}}_1\hat{\mathbf{X}})^2$$

- Trifocal tensor

Problems and Their Nonlinear Formulations – Reprojection Error

- Planar Homography

$$dist(\hat{\mathbf{x}}'-\mathbf{x}')^2 = dist(\hat{\mathbf{H}}\hat{\mathbf{x}}-\mathbf{x}')^2$$

$$= \left(\frac{\hat{\mathbf{H}}\hat{\mathbf{x}}|_1}{\hat{\mathbf{H}}\hat{\mathbf{x}}|_3} - \mathbf{x}'_1\right)^2 + \left(\frac{\hat{\mathbf{H}}\hat{\mathbf{x}}|_2}{\hat{\mathbf{H}}\hat{\mathbf{x}}|_3} - \mathbf{x}'_2\right)^2$$

- Camera Calibration

$$dist(\hat{\mathbf{x}}-\mathbf{x})^2 = dist(\hat{\mathbf{P}}\overline{\mathbf{X}}-\mathbf{x})^2$$

$$= \left(\frac{\hat{\mathbf{P}}\overline{\mathbf{X}}|_1}{\hat{\mathbf{P}}\overline{\mathbf{X}}|_3} - \mathbf{x}_1\right)^2 + \left(\frac{\hat{\mathbf{P}}\overline{\mathbf{X}}|_2}{\hat{\mathbf{P}}\overline{\mathbf{X}}|_3} - \mathbf{x}_2\right)^2$$

- Fundamental matrix

$$dist(\mathbf{x}-\hat{\mathbf{P}}_1\hat{\mathbf{X}})^2 =$$

$$= \left(\frac{\hat{\mathbf{P}}_1\hat{\mathbf{X}}|_1}{\hat{\mathbf{P}}_1\hat{\mathbf{X}}|_3} - \mathbf{x}_1\right)^2 + \left(\frac{\hat{\mathbf{P}}_1\hat{\mathbf{X}}|_2}{\hat{\mathbf{P}}_1\hat{\mathbf{X}}|_3} - \mathbf{x}_2\right)^2$$

Planar homography

$$f = \frac{1}{2}\sum_{i=1}^{m} f_i^{\,2}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m} f_i^{\,2}(x_1, \cdots, x_j, \cdots, x_n)$$

- k corresponding points
- 4k space
- Sub-manifold of 9 + 2k parameters
- Minimize distance to sub-manifold

- 9+2k is n
- 4k is m

$$f = \frac{1}{2}\sum_{i=1}^{m} f_i^{\,2}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{k} (\mathbf{x}_i^{'} - \hat{\mathbf{H}}(\hat{\mathbf{x}}_i))^2 + \frac{1}{2}\sum_{i=1}^{k} (\mathbf{x}_i - \hat{\mathbf{x}}_i)^2$$

Fundamental Matrix

$$f = \frac{1}{2}\sum_{i=1}^{m} f_i^{\,2}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m} f_i^{\,2}(x_1, \cdots, x_j, \cdots, x_n)$$

- k corresponding points
- 4k space
- Sub-manifold of 12+ 3k parameters
- Minimize distance to sub-manifold

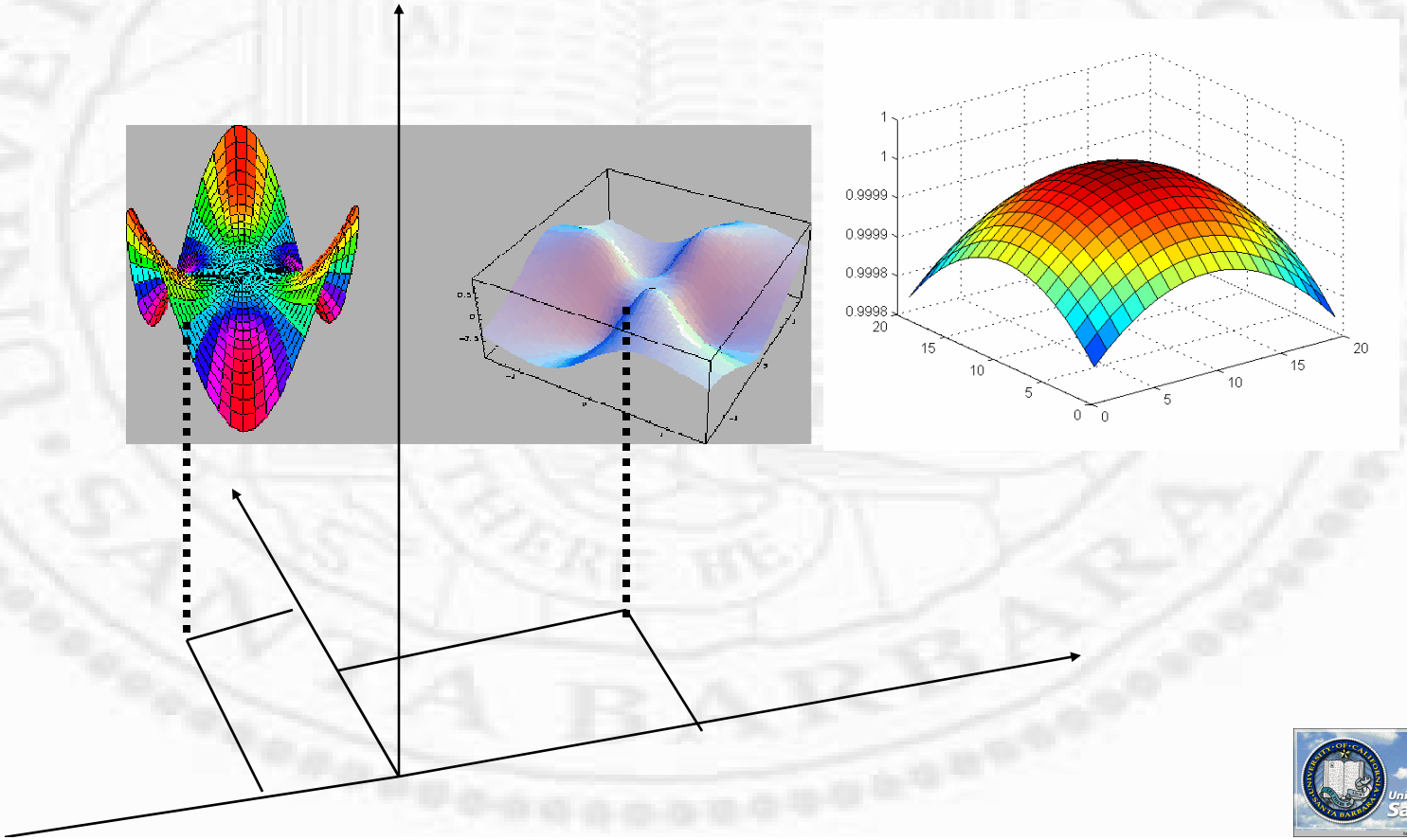
- 12+3k is n
- 4k is m

$$f = \frac{1}{2}\sum_{i=1}^{m} f_i^{\,2}(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{k} (\mathbf{x}_i - \hat{\mathbf{P}}_o \hat{\mathbf{X}}_i)^2 + \frac{1}{2}\sum_{i=1}^{k} (\mathbf{x}_i' - \hat{\mathbf{P}}_1 \hat{\mathbf{X}}_i)^2$$

$$\hat{\mathbf{P}}_o = [\mathbf{I} \,|\, \mathbf{0}]$$

Graphical Interpretation

- Given some parameters
- Calculate error
- What to do to minimize error?



Options

- If local surface is linear
  - Sampson approximation
- If local surface is quadratic
  - Newton's approximation
- If local surface is neither
  - LM (Newton + gradient descent)
- Unless you are extremely lucky, you have to assume local surface is of a complicated shape and use LM

Important Observation

- The Jacobian matrix is actually very sparse in these problems
  - Points do not affect each other
  - Multiple camera matrices do not affect each other
- Fast solution for sparse equations is possible

Example: Fundamental Matrix

$$f = \frac{1}{2}\sum_{i=1}^{m} f_i^2(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{k}(\mathbf{x}_i - \hat{\mathbf{P}}_o\hat{\mathbf{X}}_i)^2 + \frac{1}{2}\sum_{i=1}^{k}(\mathbf{x}_i' - \hat{\mathbf{P}}_1\hat{\mathbf{X}}_i)^2$$

$$f_i = (\mathbf{x}_i - \hat{\mathbf{P}}_o\hat{\mathbf{X}}_i)^2$$

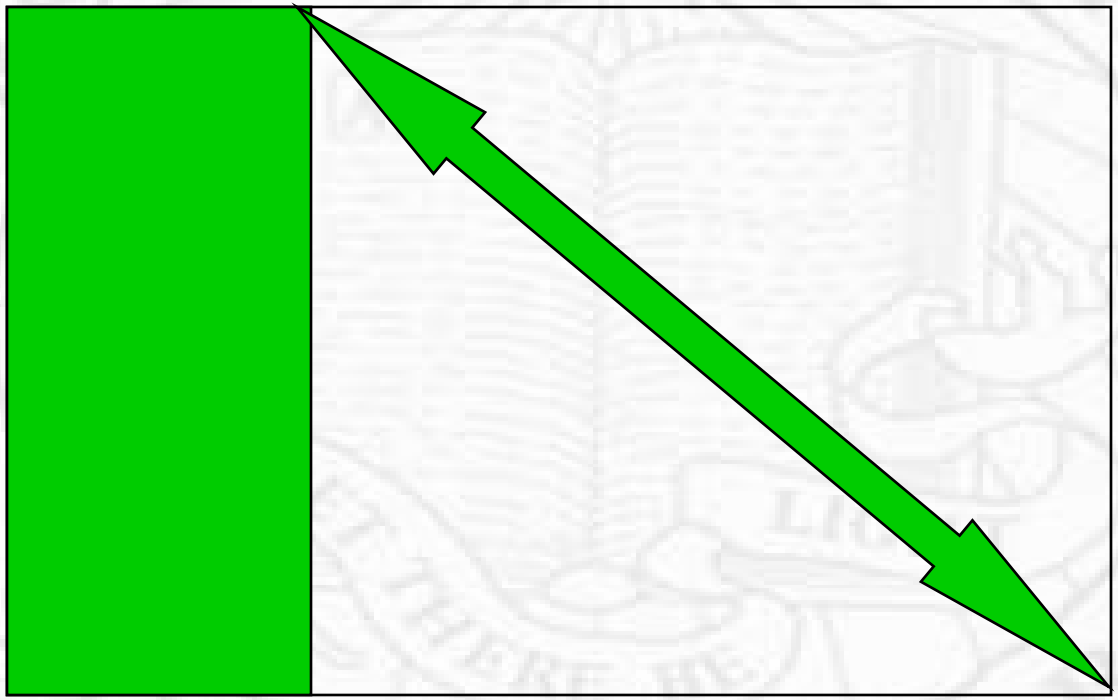
$$f_i' = (\mathbf{x}_i' - \hat{\mathbf{P}}_1\hat{\mathbf{X}}_i)^2$$

$$\nabla f_i = \left(\frac{\partial f_i}{\partial \hat{\mathbf{P}}_o}, \frac{\partial f_i}{\partial \hat{\mathbf{P}}_1}, \frac{\partial f_i}{\partial \hat{\mathbf{X}}_j}\right)^T = \left(\neq 0, 0, 0, \cdots, \frac{\partial f_i}{\partial \hat{\mathbf{X}}_i}, \cdots, 0\right)$$

$$\nabla f_i' = \left(\frac{\partial f_i'}{\partial \hat{\mathbf{P}}_o}, \frac{\partial f_i'}{\partial \hat{\mathbf{P}}_1}, \frac{\partial f_i'}{\partial \hat{\mathbf{X}}_j}\right)^T = \left(0, \neq 0, 0, \cdots, \frac{\partial f_i}{\partial \hat{\mathbf{X}}_i}, \cdots, 0\right)$$

Graphically

$$\frac{\partial f_i}{\partial \hat{\mathbf{P}}_o}, \frac{\partial f_i}{\partial \hat{\mathbf{P}}_1} \qquad \frac{\partial f_i}{\partial \hat{\mathbf{X}}_j}$$

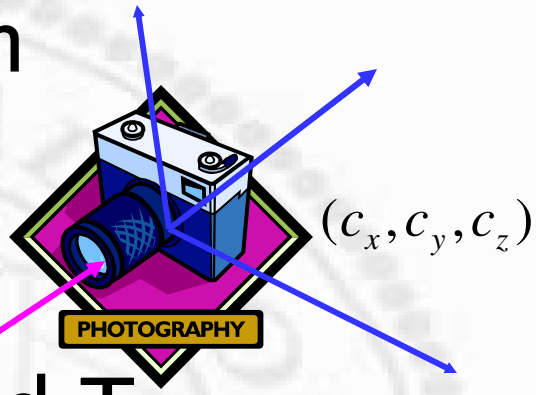


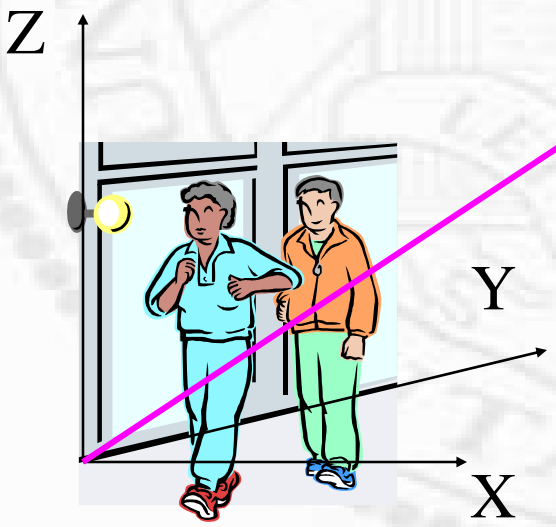
Other Important Numerical Issues

- Data Normalization
- Linear: weighted least square
- Linear: robust error function
- Nonlinear: Ransac feature selection
- Three keywords:
  - Normalization
  - Weighting
  - Selection

Data Normalization

- x=PX for camera calibration
- X=(i,j,k)+big offset
- Use (i,j,k) instead of X
- Add big offset back to R and T





Big offset

Example: 2D Homography

- $x' = Hx$, $y' = Gy$
- $y = Tx$ and $y' = T'x'$
- $y' = T'HT^{-1}y$
- $G = T'HT^{-1}$, $H = T'^{-1}GT$
- Ideally if T and T' are similarity, we expect the same results, if H is computed
  - from $x' = Hx$
  - from G, and G is computed from $y' = Gy$
- In reality, that is not the case if we are not careful

Direct Linear Transformation (DLT)

$$\mathbf{x}_i' = \mathbf{H}\mathbf{x}_i$$

$$\mathbf{x}_i' \times \mathbf{H}\mathbf{x}_i = 0$$

$$\mathbf{x}_i' = (x_i', y_i', w_i')^\mathsf{T} \qquad \mathbf{H}\mathbf{x}_i = \begin{pmatrix} \mathbf{h}^{1\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{2\mathsf{T}}\mathbf{x}_i \\ \mathbf{h}^{3\mathsf{T}}\mathbf{x}_i \end{pmatrix}$$

$$\xi = \mathbf{x}_i' \times \mathbf{H}\mathbf{x}_i = \begin{pmatrix} y_i'\mathbf{h}^{3\mathsf{T}}\mathbf{x}_i - w_i'\mathbf{h}^{2\mathsf{T}}\mathbf{x}_i \\ w_i'\mathbf{h}^{1\mathsf{T}}\mathbf{x}_i - x_i'\mathbf{h}^{3\mathsf{T}}\mathbf{x}_i \\ x_i'\mathbf{h}^{2\mathsf{T}}\mathbf{x}_i - y_i'\mathbf{h}^{1\mathsf{T}}\mathbf{x}_i \end{pmatrix}$$

$$\begin{bmatrix} \mathbf{0}^\mathsf{T} & -w_i'\mathbf{x}_i^\mathsf{T} & y_i'\mathbf{x}_i^\mathsf{T} \\ w_i'\mathbf{x}_i^\mathsf{T} & \mathbf{0}^\mathsf{T} & -x_i'\mathbf{x}_i^\mathsf{T} \\ -y_i'\mathbf{x}_i^\mathsf{T} & x_i'\mathbf{x}_i^\mathsf{T} & \mathbf{0}^\mathsf{T} \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = 0$$

$$\mathbf{A}_i\mathbf{h} = 0$$

Error in DLT

$$\xi_y = \mathbf{y}' \times \mathbf{G}\mathbf{y} = \mathbf{T}'\mathbf{x}' \times (\mathbf{T}'\mathbf{H}\mathbf{T}^{-1})\mathbf{T}\mathbf{x}$$

$$= \mathbf{T}'\mathbf{x}' \times \mathbf{T}'\mathbf{H}\mathbf{x} = \mathbf{T}'^{*}(\mathbf{x}' \times \mathbf{H}\mathbf{x}) = \mathbf{T}'^{*}\xi_x$$

$$\because \mathbf{T}\mathbf{x} \times \mathbf{T}\mathbf{y} = \mathbf{T}^{*}(\mathbf{x} \times \mathbf{y}) \quad \mathbf{T}^{*} = \det(\mathbf{T})\mathbf{T}^{-T}$$

$$\mathbf{T}' = \begin{bmatrix} s\mathbf{R} & \mathbf{T} \\ \mathbf{0}^T & 1 \end{bmatrix} \rightarrow \mathbf{T}'^{*} = s\begin{bmatrix} \mathbf{R} & \mathbf{0} \\ -\mathbf{T}^T\mathbf{R} & s \end{bmatrix}$$

$$\begin{bmatrix} \xi_1^{\,y} \\ \xi_2^{\,y} \\ \xi_3^{\,y} \end{bmatrix} = s\begin{bmatrix} \mathbf{R} & \mathbf{0} \\ -\mathbf{T}^T\mathbf{R} & s \end{bmatrix}\begin{bmatrix} \xi_1^{\,x} \\ \xi_2^{\,x} \\ \xi_3^{\,x} \end{bmatrix}$$

$$\begin{bmatrix} \xi_1^{\,y} \\ \xi_2^{\,y} \end{bmatrix} = s\mathbf{R}\begin{bmatrix} \xi_1^{\,x} \\ \xi_2^{\,x} \end{bmatrix}$$

$$\Rightarrow \left\| \begin{matrix} \xi_1^{\,y} \\ \xi_2^{\,y} \end{matrix} \right\| = s\left\| \begin{matrix} \xi_1^{\,x} \\ \xi_2^{\,x} \end{matrix} \right\|$$

Non-invariance of DLT

Given $\mathrm{x}_i \leftrightarrow \mathrm{x}'_i$ and H computed by DLT,

and $y_i = \mathrm{T}\mathrm{x}_i, y'_i = \mathrm{T}'\mathrm{x}'_i$

Does the DLT algorithm applied to $\mathrm{y}_i \leftrightarrow \mathrm{y}'_i$ *yield* $G = \mathrm{T}'\mathrm{H}\mathrm{T}^{-1}$*?*

$$\text{minimize} \sum_i d_{\text{alg}}(\mathrm{x}'_i, \mathrm{H}\mathrm{x}_i)^2 \text{subject to} \|\mathrm{H}\| = 1$$

$$\Leftrightarrow \text{minimize} \sum_i d_{\text{alg}}(\mathrm{y}'_i, \mathrm{G}\mathrm{y}_i)^2 \text{subject to} \|\mathrm{H}\| = 1$$

$$\not\Leftrightarrow \text{minimize} \sum_i d_{\text{alg}}(y'_i, \mathrm{G}\mathrm{y}_i)^2 \text{subject to} \|\mathrm{G}\| = 1$$

Invariance of Geometric Error

- x'=Hx, y'=Gy
- y=Tx and y'=T'x'
- y'=T'HT^{-1}y
- G=T'HT^{-1} H=T' $^{-1}$GT
- Difference is that: geometric error is minimized, say through a search, the error surface is scaled the same way

$$d(\mathbf{y}', \mathbf{Gy})$$
$$= d(\mathbf{T}'\mathbf{x}', \mathbf{T}'\mathbf{H}\mathbf{T}^{-1}\mathbf{T}\mathbf{x})$$
$$= d(\mathbf{T}'\mathbf{x}', \mathbf{T}'\mathbf{H}\mathbf{x})$$
$$= d(\mathbf{x}', \mathbf{H}\mathbf{x}) = sd(\mathbf{x}', \mathbf{H}\mathbf{x})$$

If **T**' is Euclidian If **T**' is similarity

Data Normalization in DLT

- Zero centered
- Uniform scaling so that average distance to origin is sqrt(2)
- Intuitively (x,y,w) can be (100, 100, 1) – a difference of 100:1
- xx', xy', yx' yy' will be 100:1 to xw', yw', and 10000:1 to ww'

Least Squares: Normal Equation

$$\mathbf{A}_{m\times n}\mathbf{X}_{n\times 1} = \mathbf{B}_{m\times 1} \Rightarrow \mathbf{A}^{T}{}_{n\times m}\mathbf{A}_{m\times n}\mathbf{X}_{n\times 1} = \mathbf{A}^{T}{}_{n\times m}\mathbf{B}_{m\times 1}$$

$$\mathbf{X}_{n\times 1} = (\mathbf{A}^{T}{}_{n\times m}\mathbf{A}_{m\times n})^{-1}\mathbf{A}^{T}{}_{n\times m}\mathbf{B}_{m\times 1}$$

m: number of constraints
n: number of parameters
if m<n multiple solutions
if m=n exact solution
if m>n least-squares solution

Type of Errors

- Noise – noncatastrophic error
  - Unavoidable in real data
- Outliers – catastrophic error
  - Often happen in real world
    - Matching error
    - Tracking error
    - Etc.

Weighted Least Square

- data are *not* equally reliable
  - C is covariance matrix
  - Radar tracking: signal strength
  - Feature tracking: SSD error

$$\mathbf{W}_{m\times m}\mathbf{A}_{m\times n}\mathbf{X}_{n\times 1}=\mathbf{W}_{m\times m}\mathbf{B}_{m\times 1}$$

$$(\mathbf{W}_{m\times m}\mathbf{A}_{m\times n})^{T}\mathbf{W}_{m\times m}\mathbf{A}_{m\times n}\mathbf{X}_{n\times 1}=(\mathbf{W}_{m\times m}\mathbf{A}_{m\times n})^{T}\mathbf{W}_{m\times m}\mathbf{B}_{m\times 1}$$

$$\mathbf{X}_{n\times 1}=((\mathbf{W}_{m\times m}\mathbf{A}_{m\times n})^{T}\mathbf{W}_{m\times m}\mathbf{A}_{m\times n})^{-1}(\mathbf{W}_{m\times m}\mathbf{A}_{m\times n})^{T}\mathbf{W}_{m\times m}\mathbf{B}_{m\times 1}$$

$$=\left(\mathbf{A}_{m\times n}{}^{T}\mathbf{W}^{T}{}_{m\times m}\mathbf{W}_{m\times m}\mathbf{A}_{m\times n}\right)^{-1}\mathbf{A}_{m\times n}{}^{T}\mathbf{W}^{T}{}_{m\times m}\mathbf{W}_{m\times m}\mathbf{B}_{m\times 1}$$

$$=\left(\mathbf{A}_{m\times n}{}^{T}\mathbf{C}_{m\times m}\mathbf{A}_{m\times n}\right)^{-1}\mathbf{A}_{m\times n}{}^{T}\mathbf{C}_{m\times m}\mathbf{B}_{m\times 1}$$

$$=\mathbf{L}_{m\times m}\mathbf{B}_{m\times 1}$$

$$\Rightarrow\ \hat{\mathbf{X}}=\mathbf{LB}$$

$$\mathbf{X}_{n\times 1}=(\mathbf{A}^{T}{}_{n\times m}\mathbf{A}_{m\times n})^{-1}\mathbf{A}^{T}{}_{n\times m}\mathbf{B}_{m\times 1}\quad\text{(before)}$$

What to do with outliers?

- A field called robust statistics
  - Identify outliers and remove them (easier said than done, especially for leverage points)
  - M-estimator
    - minimize the influence of outliers
  - LMS (least median squares), LTS (least trimmed squares)
    - nonlinear and eliminate outliers
- We can only show some simple examples

M-Estimators

- In standard LS regression, the error grows linearly and unbounded
  - outliers influence the fit because of large error
  - try to reduce the influence by using a different error norm

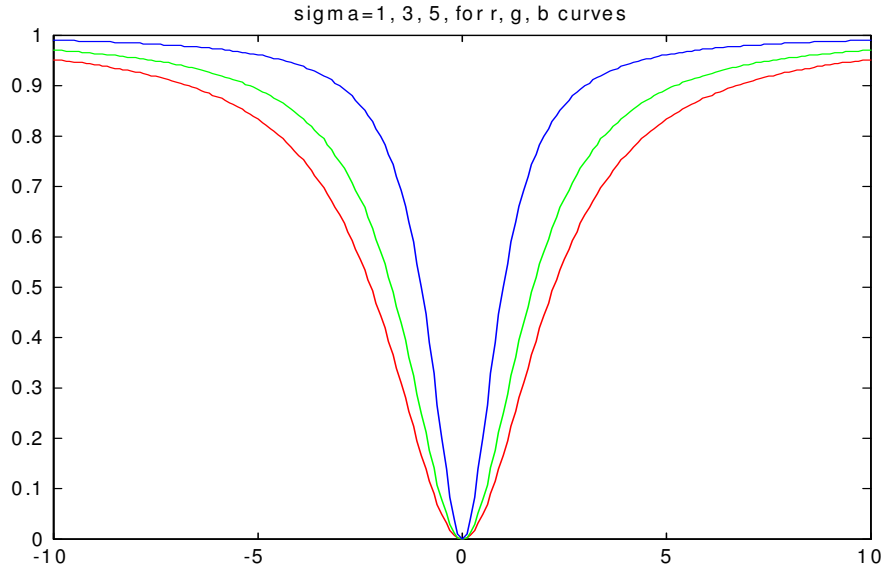
Robust Error Norm

- Use a robust error norm that down weighs the outliers

$$\min \sum_i \rho(A_i X = B_i, \sigma)$$

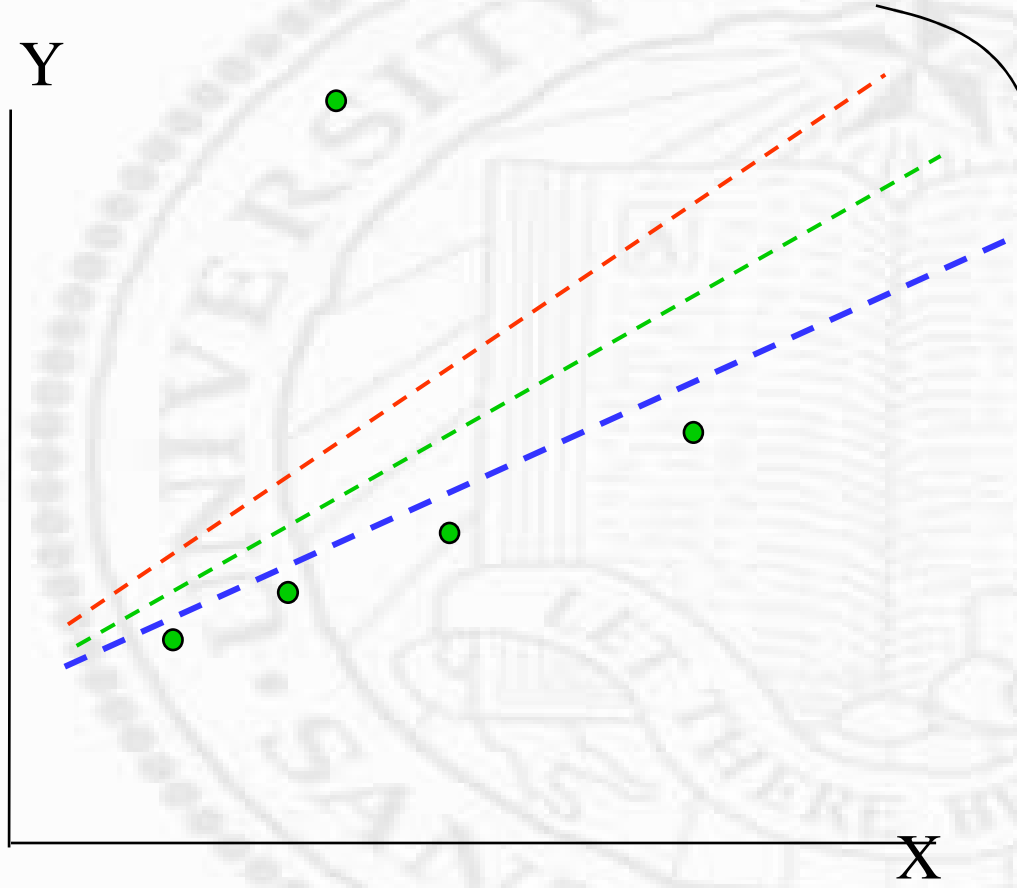
$$\rho(x,\sigma) = \frac{x^2}{x^2 + \sigma} \quad Geman\ \&\ McClure$$

$$\rho(x,\sigma) = \log(1 + \frac{1}{2}\frac{x^2}{\sigma^2}) \quad Lorentzian$$



- Weight is not a constant
- Depends on residue error
- Iterative solution

Robust Iterative Regression



Initialize

$$k = 0, w_i^{(k)} = 1$$

Iterate

$$X^{(k)} = \arg\min_X \sum_i w_i^{(k)} (A_i X - B_i)^2$$

$$w_i^{(k+1)} = \frac{(A_i X^{(k)} - B_i)^2}{(A_i X^{(k)} - B_i)^2 + \sigma^2}$$

$$k = k + 1$$

end

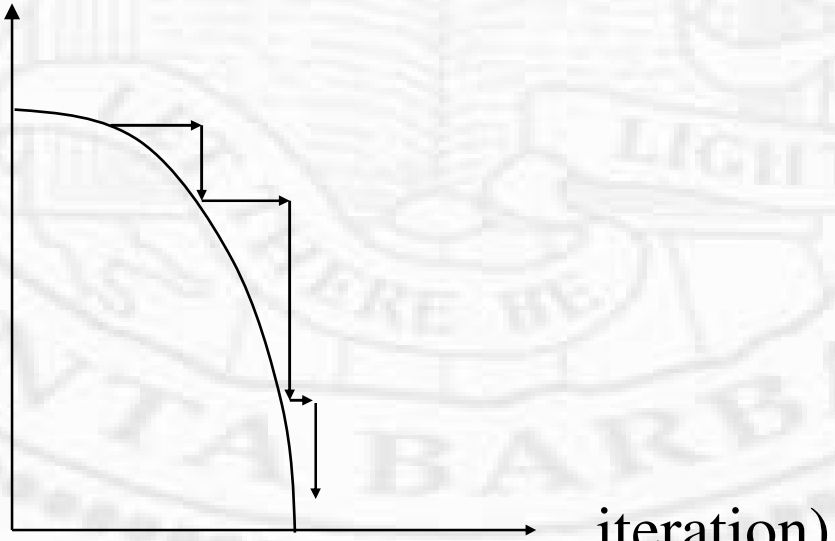
Robust Error Norms (cont.)

- But what should sigma be?
  - Too large or too small are no good
  - In fact, unless one knows how large an error a data point becomes an outlier, otherwise, sigma need be estimated just like the model parameters

$$\min_{X,\sigma} \sum_i \rho(A_i X - B_i, \sigma)$$

EM (Expectation & Minimization)

- A variation of the EM algorithm: iteratively
  - In one iteration, hold weight (mixture) constant and find the best sigma
  - In the next iteration, change weight (mixture)



LMS & LTS

- LMS (Least Median Squares)

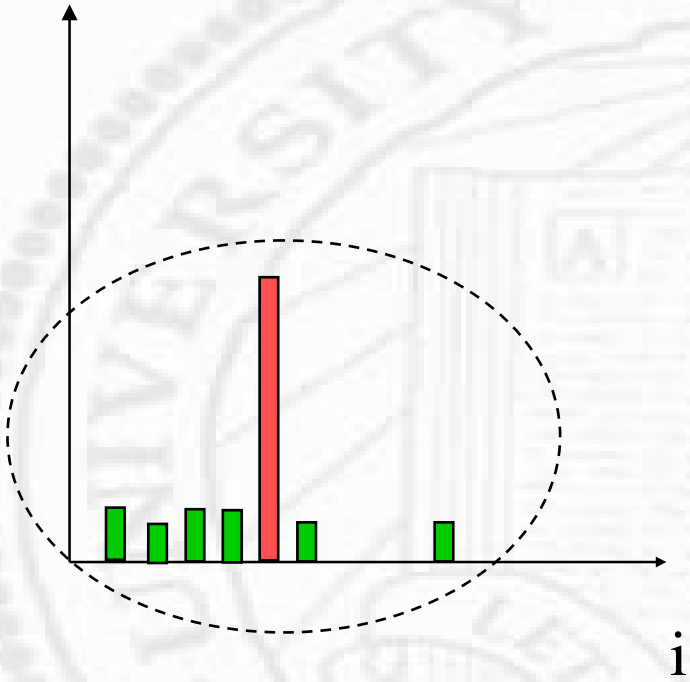
$$\min_X \; med_i \; (A_i X - B_i)^2$$

- LTS (Least Trimmed Squares)
  - Has better convergence efficiency

$$\min_X \sum_{i=1}^{h} (A_i X - B_i)^2 \quad where (A_i X - B_i)^2 \text{ is ordered}$$

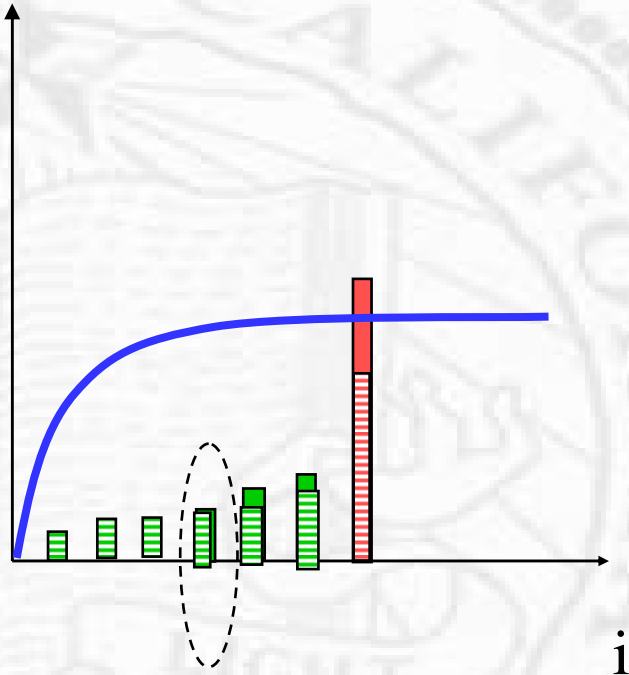
Comparison

LS Error



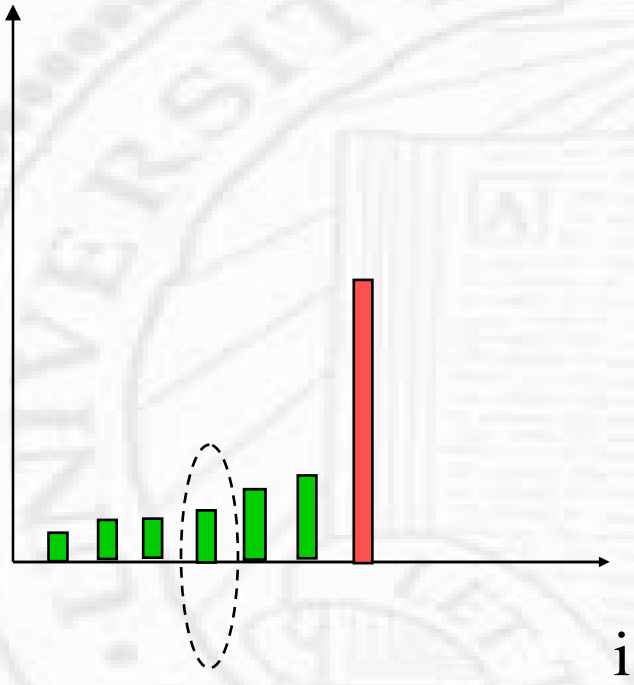
All errors are important
The sum of error is minimized

Weighted LS Error

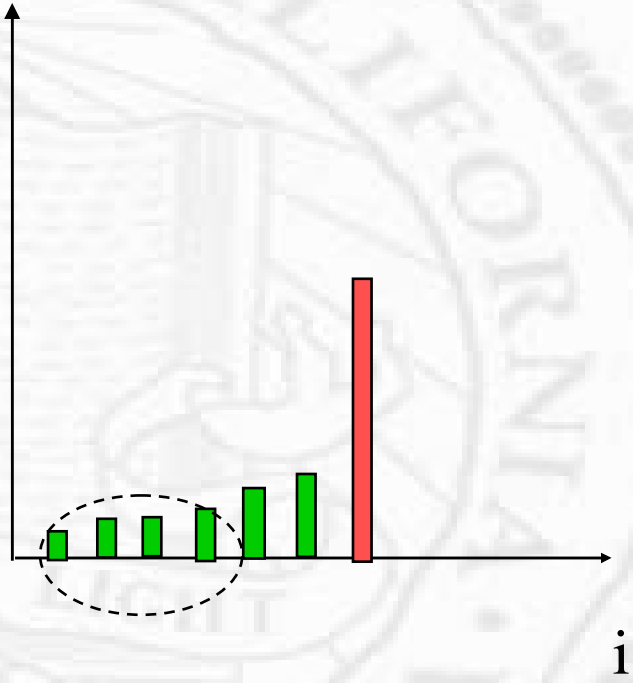


All errors are counted
Large errors are weighted less
Than they are in LS

Comparison



Not all errors are important
Large errors can be ignored



Not all errors are important
Large errors can be ignored

LMS Algorithm – RANSAC

(n data points and p parameters)

- Choose p points at random from the set of n data points
- Compute the fit of model to the p points
- Compute the median of the square of residuals for all n points
- The fitting procedure is repeated until a fit is found with sufficiently small median of squared residuals or up to some predetermined number of fitting steps

How Many Trials?

- Well, theoretically it is *C(n,p)* to find all possible *p*–tuples
- Very expensive

$1 - (1 - (1 - \varepsilon)^{p})^{m}$

ε : fraction of bad data

$(1 - \varepsilon)$: fraction of good data

$(1 - \varepsilon)^{p}$: all p samples are good

$1 - (1 - \varepsilon)^{p}$: at least one sample is bad

$(1 - (1 - \varepsilon)^{p})^{m}$: got bad data in all m tries

$1 - (1 - (1 - \varepsilon)^{p})^{m}$: got at least one good p set in m tries

How Many Trials (cont.)

- Make sure the probability is high (e.g. >95%)
- given p and epsilon, calculate m

p	5%	10%	20%	25%	30%	40%	50%
1	1	2	2	3	3	4	5
2	2	2	3	4	5	7	11
3	2	3	5	6	8	13	23
4	2	3	6	8	11	22	47
5	3	4	8	12	17	38	95