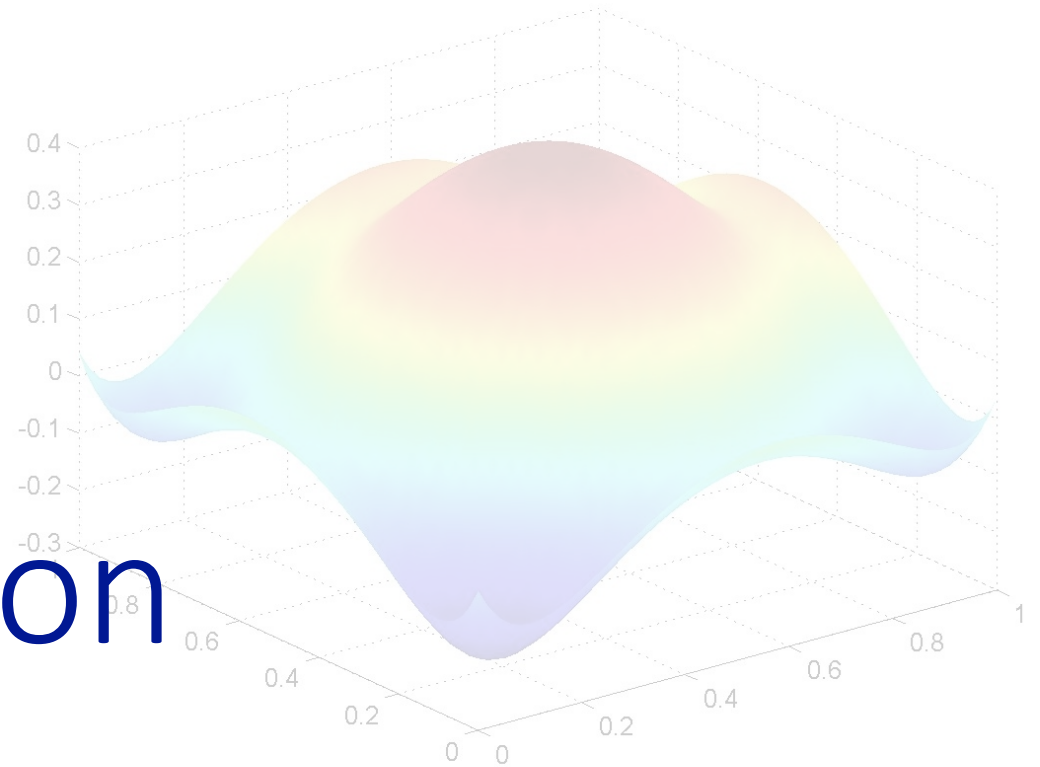


# Parameter estimation



---

Principles of Modeling for Cyber-Physical Systems

Instructor: Madhur Behl



# Parameter estimation overview

---

- Simple Linear Regression
- Least squares
- Non-linear least squares
- State-space sum of squared errors
- Non-linear optimization (estimation) methods
- Global and local search
- MATLAB implementations

# Simple Linear Regression

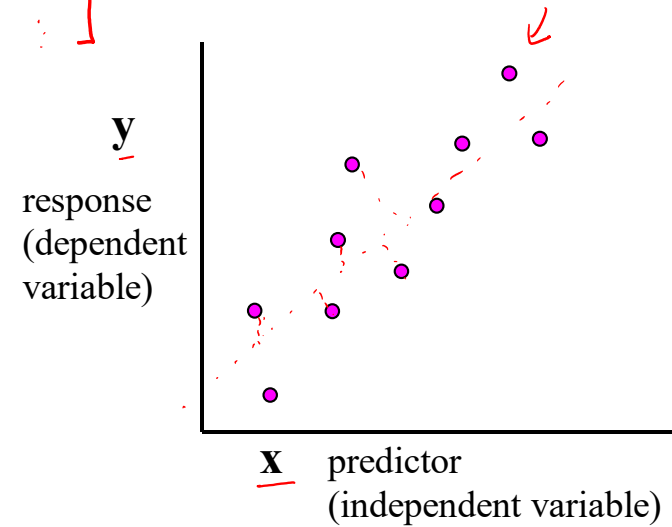
Suppose we collect some data and want to determine the relation between the observed values,  $y$ , and the independent variable,  $x$ :

$$\begin{bmatrix} (x_1, y_1) \\ \vdots \\ (x_n, y_n) \end{bmatrix}$$

We can model the data using a **linear model**

$$( \rightarrow y_i = \beta_0 + \beta_1 x_i + \epsilon_i ) \text{ True}$$

observed response    unknown intercept    unknown slope    unknown random error



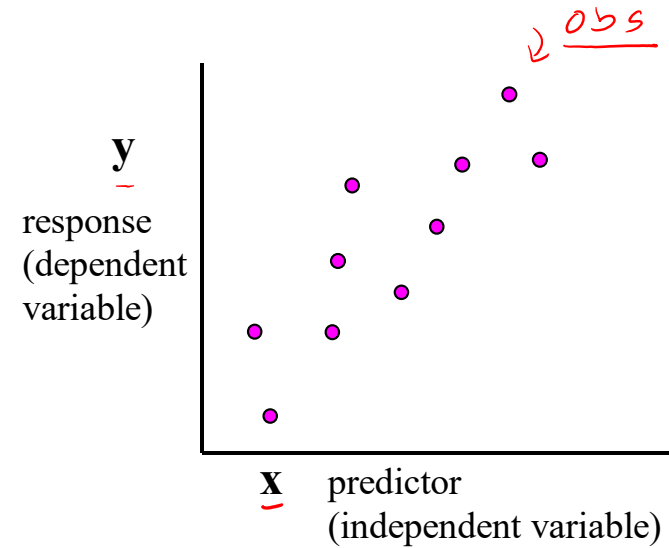
# Simple Linear Regression

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

observed response    unknown intercept    unknown slope    unknown random error

$\beta_0$  and  $\beta_1$  are the **parameters** of this linear model.

- Don't know the true values of the parameters.
- Estimate them using the assumed model and the observations (data)

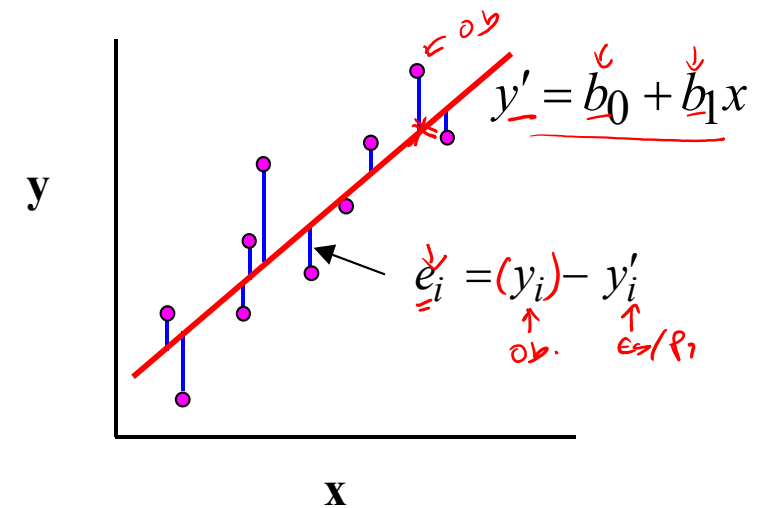


# Simple Linear Regression

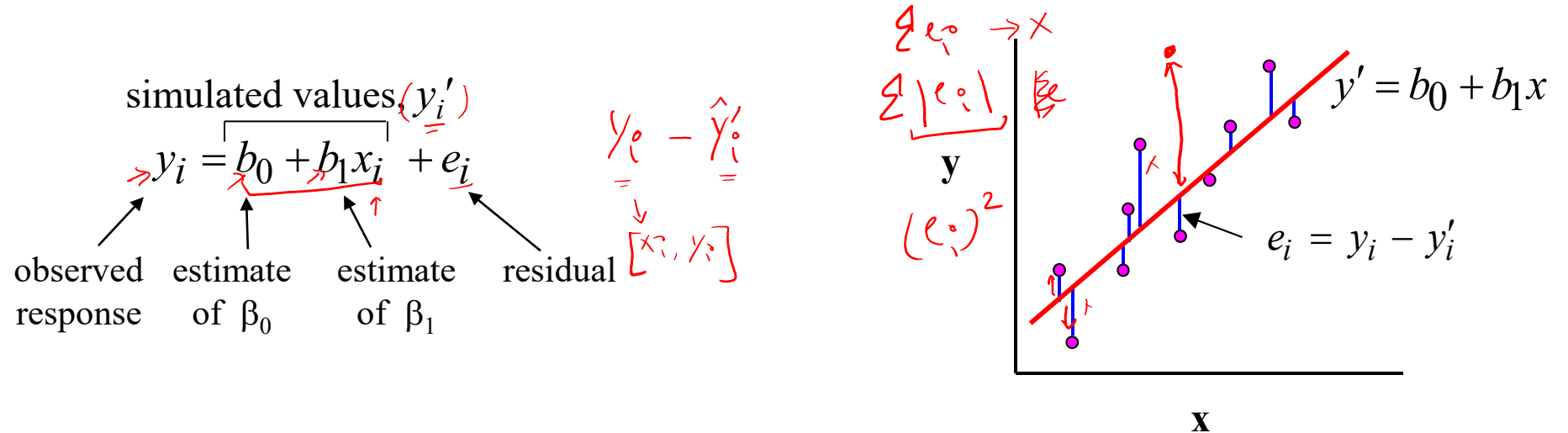
$$\underline{y}_i = \underbrace{(b_0 + b_1 x_i)}_{\text{estimate of } \beta_0 + \text{estimate of } \beta_1} + \underline{e_i^2}$$

observed response      estimate of  $\beta_0$       estimate of  $\beta_1$       residual

- Estimate  $b_0$  and  $b_1$  to obtain the best fit of the simulated values to the observations.
- One method: **Minimize sum of squared errors, or residuals.**



# Simple Linear Regression



Sum of squared residuals:

$$\rightarrow S(b_0, b_1) = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - y_i')^2 = \sum_{i=1}^n (y_i - b_0 - b_1 x_i)^2 \text{ SSE}$$

To minimize:

Set  $\frac{\partial S}{\partial b_0} = 0$  and  $\frac{\partial S}{\partial b_1} = 0$

# Simple Linear Regression

$$\rightarrow S(\underline{b_0}, \underline{b_1}) = \sum_{i=1}^n e_i^2 = \sum_{i=1}^n (y_i - y'_i)^2 = \sum_{i=1}^n (y_i - \overset{(-)}{\underline{b_0}} - \underline{b_1}x)^2$$

$$\text{Set } \frac{\partial S}{\partial \underline{b_0}} = \underline{0} \quad \text{and} \quad \frac{\partial S}{\partial \underline{b_1}} = 0$$

$$\rightarrow \underline{b_0}n + \underline{b_1} \sum_{i=1}^n x_i = \sum_{i=1}^n y_i \leftarrow$$

$$\rightarrow \underline{b_0} \sum_{i=1}^n x_i + \underline{b_1} \sum_{i=1}^n x_i^2 = \sum_{i=1}^n x_i y_i$$

This results in the **normal equations**:

- Solve these equations to obtain expressions for  $b_0$  and  $b_1$ , the parameter estimates that give the best fit of the simulated and observed values.



# Linear Regression in Matrix Form

Linear regression model:  $y_i = b_0 + b_1 x_i + e_i, i=1..n \rightarrow \underline{y} = \underline{X} \underline{b} + \underline{e}$

$$\underline{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

vector of  
observed  
values

$$\underline{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

matrix of  
Predictors/  
features

$$\underline{b} = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

vector of  
parameters

$$\underline{e} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{bmatrix}$$

vector of  
residuals

$$y_i = b_0 + x_i \cdot b_1 + e_i$$

# Linear Regression in Matrix Form

Linear regression model:  $y_i = b_0 + b_1 x_i + e_i$ ,  $i=1..n \rightarrow \underline{y} = \underline{X} \underline{b} + \underline{e}$

- The **normal equations** ( $\underline{b}'$  is the vector of least-squares estimates of  $\underline{b}$ ):

Using summations  
And setting the derivative to 0

$$\begin{aligned} b_0 n + b_1 \sum_{i=1}^n x_i &= \sum_{i=1}^n y_i \quad \checkmark \\ b_0 \sum_{i=1}^n x_i + b_1 \sum_{i=1}^n x_i^2 &= \sum_{i=1}^n x_i y_i \quad \checkmark \end{aligned}$$

Using matrix notation:

$$\underline{X}^T \underline{X} \underline{b}' = \underline{X}^T \underline{y} \rightarrow \underline{b}' = (\underline{X}^T \underline{X})^{-1} \underline{X}^T \underline{y}$$

# Ordinary least squares

$$y_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_{k+1} x_{ik}$$

$$\begin{array}{c} \downarrow \\ \left[ \begin{array}{c} Y_1 \\ Y_2 \\ \vdots \\ \vdots \\ Y_n \end{array} \right]_{n \times 1} \\ \leftarrow \end{array} = \begin{array}{c} \downarrow \\ \left[ \begin{array}{cccccc} 1 & X_{11} & X_{21} & \dots & X_{k1} \\ 1 & X_{12} & X_{22} & \dots & X_{k2} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & X_{1n} & X_{2n} & \dots & X_{kn} \end{array} \right] \\ \begin{array}{c} \underline{n} \times \underline{k+1} \end{array} \end{array} \begin{array}{c} \left[ \begin{array}{c} \beta_1 \\ \beta_2 \\ \vdots \\ \vdots \\ \beta_{k+1} \end{array} \right] \\ \begin{array}{c} \underline{k+1} \times \underline{1} \end{array} \end{array} + \begin{array}{c} \left[ \begin{array}{c} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \vdots \\ \epsilon_n \end{array} \right]_{n \times 1} \end{array}$$

This can be rewritten more simply as:

$$\underline{y} = \underline{X}\underline{\beta} + \underline{\epsilon}$$

# Ordinary least squares

---

$$\underline{e} = \underline{y} - \underbrace{X\hat{\beta}}_{\hat{y}}$$

The sum of squared residuals (RSS) is  $\underline{e}'\underline{e}$ .

$$\begin{bmatrix} e_1 & e_2 & \dots & \dots & e_n \end{bmatrix}_{1 \times n} \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ \vdots \\ e_n \end{bmatrix}_{n \times 1} = \begin{bmatrix} e_1 \times e_1 + e_2 \times e_2 + \dots + e_n \times e_n \end{bmatrix}_{1 \times 1}$$

# Ordinary least squares

---

The sum of squared residuals (RSS) is  $e'e$ .

$$\begin{aligned} \underline{e'e} &= (\underline{y} - \underline{X}\hat{\underline{\beta}})'(\underline{y} - \underline{X}\hat{\underline{\beta}}) \\ \text{SSE} &= \underline{y}'\underline{y} - \underline{\hat{\beta}}'\underline{X}'\underline{y} - \underline{y}'\underline{X}\hat{\underline{\beta}} + \underline{\hat{\beta}}'\underline{X}'\underline{X}\hat{\underline{\beta}} \\ \text{SS}(\hat{\underline{\beta}}) &= \underline{y}'\underline{y} - 2\underline{\hat{\beta}}'\underline{X}'\underline{y} + \underline{\hat{\beta}}'\underline{X}'\underline{X}\hat{\underline{\beta}} \end{aligned}$$

# Ordinary least squares

$$e'e = y'y - 2\hat{\beta}'X'y + \hat{\beta}'X'X\hat{\beta} \leftarrow$$

A handwritten diagram in red ink illustrating the matrix multiplication  $X'y$ . It shows a column vector  $y$  with elements  $y_1, \dots, y_n$  and a matrix  $X$  with elements  $x_{11}, \dots, x_{1k}$  in the first row and  $x_{n1}, \dots, x_{nk}$  in the  $n$ th row. A red arrow points from the  $X$  matrix to the  $X'y$  product.

$$\rightarrow \frac{\partial e'e}{\partial \hat{\beta}} = -2X'y + 2X'X\hat{\beta} = 0$$

$$(X'X)\hat{\beta} = X'y \quad \hat{\beta} = \underbrace{(X'X)^{-1}X'y}_{\text{closed form}} \rightarrow$$

$[k+1 \times 1]$

# Linear versus Nonlinear Models

**Linear models:** Sensitivities of the output are **not** a function of the model parameters:

$$y'_i = b_0 + b_1 x_i$$

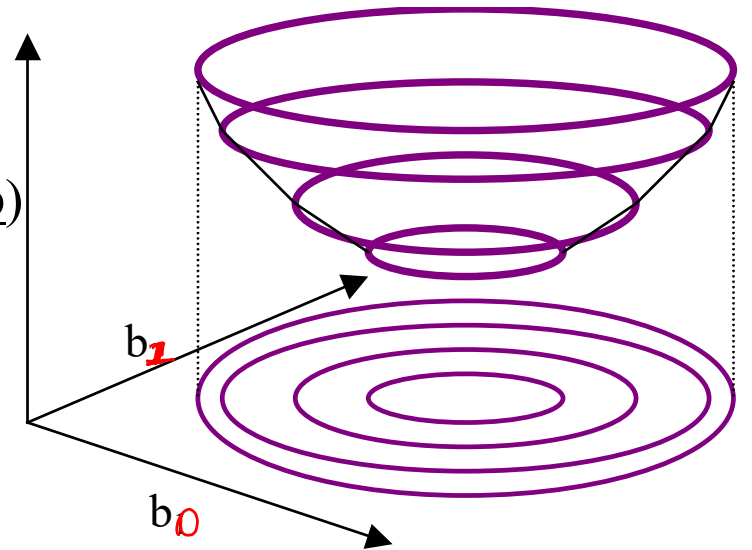
$$y = f(\beta, x)$$

$$\frac{dy}{d\beta} = g(\beta, x)$$

$$\frac{\partial y'_i}{\partial b_0} = 1 \text{ and } \frac{\partial y'_i}{\partial b_1} = x_i ; \text{ recall}$$

$$\underline{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

$$\sum (y_i - \hat{y}_i)^2 \leftarrow S(\underline{b})$$



# Linear versus Nonlinear parameters

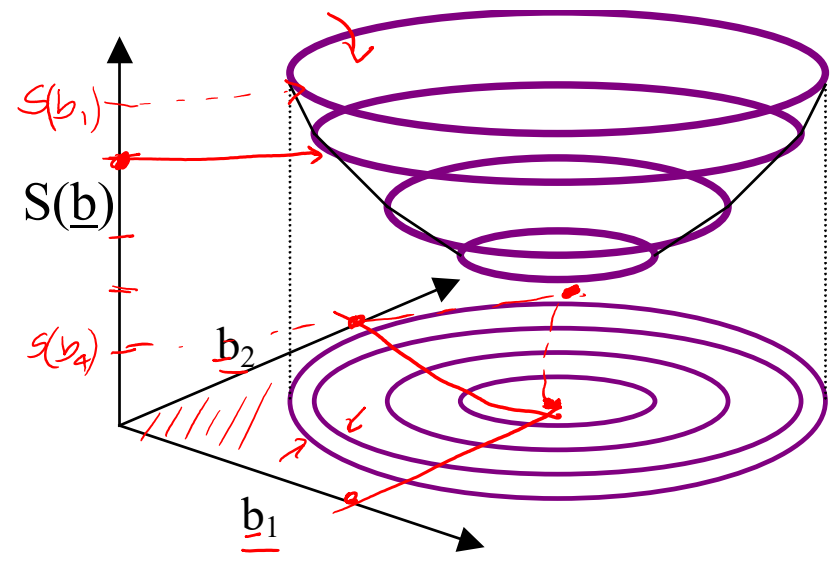
$$SSE(b_0, b_1) = \sum (y_i - (b_0 + b_1 x_i))^2$$

- Linear models have elliptical objective function surfaces.
- i.e. the level sets of the objective function (sum of errors squared) are ellipsis.

*One step to get to the minimum.*

$$\beta^* = (X'X)^{-1} X'y$$

**Nonlinear parametric models:** Sensitivities are a function of the model parameters.



With two parameter



# Nonlinearity is in parameter space.

---

$$x(k + 1) = \underline{A}_\theta(k)x(k) + \underline{B}_\theta(k)u(k) \quad \begin{matrix} \downarrow \\ \underline{A}_\theta \cdot \underline{C}_\theta \end{matrix}$$

$$y(k) = \underline{C}_\theta(k)x(k) + \underline{D}_\theta(k)u(k)$$

$$\theta = [\underline{U}_L, \underline{U}_{con}, \underline{U}_{int}, \dots, \underline{C}_e, \underline{C}_z, \dots]$$

Elements of A, B, C, and D could be non-linear in the parameter  $\theta$

# Nonlinear Estimation

---

Suppose that we have collected data on the output/response  $\underline{Y}$  (n samples),  
 $\rightarrow (y_1, y_2, \dots, y_n)$

corresponding to n sets of values of the independent variables/predictors/features  $X_1, X_2, \dots$  and  $X_p$   $[T_a, T_g, T_i, Q_{sd} \dots]$

- $(x_{11}, x_{21}, \dots, x_{p1})$ ,
- $(x_{12}, x_{22}, \dots, x_{p2})$ ,
- ... and
- $(x_{1n}, x_{2n}, \dots, x_{pn})$ .

# Nonlinear Estimation

For possible values  $\theta_1, \theta_2, \dots, \theta_q$  of the parameters, the residual sum of squares function

$$S(\theta_1, \theta_2, \dots, \theta_q) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n [y_i - f(x_{1i}, x_{2i}, \dots, x_{pi} | \theta_1, \theta_2, \dots, \theta_q)]^2$$

$$\hat{y}_i = f(x_{1i}, x_{2i}, \dots, x_{pi} | \theta_1, \theta_2, \dots, \theta_q)$$

# Nonlinear Estimation

---

$$S(\theta_1, \theta_2, \dots, \theta_q) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n \left[ y_i - \underset{\bar{f}}{f}(x_{1i}, x_{2i}, \dots, x_{pi} | \theta_1, \theta_2, \dots, \theta_q) \right]^2$$

The least squares estimates of  $\underline{\theta}_1, \theta_2, \dots, \underline{\theta}_q$  are values which minimize  $S(\underline{\theta}_1, \theta_2, \dots, \theta_q)$ .

# Nonlinear Estimation

$$S(\theta_1, \theta_2, \dots, \theta_q) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \sum_{i=1}^n \left[ y_i - f(x_{1i}, x_{2i}, \dots, x_{pi} | \theta_1, \theta_2, \dots, \theta_q) \right]^2$$

To find the least squares estimate we need to determine when all the derivatives  $S(\theta_1, \theta_2, \dots, \theta_q)$  with respect to each parameter  $\theta_1, \theta_2, \dots$  and  $\theta_q$  are equal to zero.

This will involve, terms with partial derivatives of the non-linear function  $f$ .

$$\left( \frac{\delta f(\dots)}{\delta \theta_1} \right), \left( \frac{\delta f(\dots)}{\delta \theta_2} \right), \dots, \frac{\delta f(\dots)}{\delta \theta_q}$$

Gen. -

# Nonlinear Estimation

---

$$\frac{\delta f(\dots)}{\delta \theta_1}, \frac{\delta f(\dots)}{\delta \theta_2}, \dots, \frac{\delta f(\dots)}{\delta \theta_q}$$

Closed form analytical solutions are not possible.

It is usually necessary to develop an iterative technique for solving them

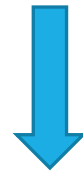
# Recall..

$f(x; \theta)$        $sse = g(\theta)$

$$x(k + 1) = A_{\theta}(k)x(k) + B_{\theta}(k)u(k)$$

$$y(k) = C_{\theta}(k)x(k) + D_{\theta}(k)u(k)$$

$sse = \sum (y_i - \hat{y}_i)^2$



$$\hat{y}(k) = f(\hat{x}(k), u(k), \hat{\theta}_1, \dots, \hat{\theta}_q)$$

# How can we compute the sum of squared error for state-space models ?

---

$$\rightarrow \underline{x}(k+1) = \underline{A}_\theta \underline{x}(k) + \underline{B}_\theta u(k)$$

$$\underline{y}(k) = \underline{C}_\theta \underline{x}(k) + D_\theta u(k)$$

$$\sum_{i=0}^n (y_i - \hat{y}_i)^2$$

(n)??

Consider the LTI model



# sum of squared error for state-space models

Given  $\underline{x}(0) = \underline{x}_0$ , and  $\underline{u}(k)$ ,  $k = \underline{0}, \dots, \underline{N-1}$

$$\rightarrow \underline{y}(0) = C_\theta \underline{x}(0) + D_\theta \underline{u}(0)$$

$$C_\theta \underline{x}(1) = A_\theta \underline{x}(0) + B_\theta \underline{u}(0)$$

$$\underline{y}(1) = C_\theta \underline{x}(1) + D_\theta \underline{u}(1)$$

$$\underline{x}(2) = A_\theta \underline{x}(1) + B_\theta \underline{u}(1)$$

$$\rightarrow \underline{y}(1) = C_\theta A_\theta \underline{x}(0) + C_\theta B_\theta \underline{u}(0) + D_\theta \underline{u}(1)$$

$$\underline{x}(2) = A_\theta A_\theta \underline{x}(0) + A_\theta B_\theta \underline{u}(0) + B_\theta \underline{u}(1)$$

$$\underline{y}(2) = C_\theta \underline{x}(2) + D_\theta \underline{u}(2)$$

$$\underline{y}(2) = C_\theta A_\theta A_\theta \underline{x}(0) + C_\theta A_\theta B_\theta \underline{u}(0) + C_\theta B_\theta \underline{u}(1) + D_\theta \underline{u}(2)$$

# sum of squared error for state-space models

$$\begin{matrix}
 \hat{y} \\
 \downarrow \\
 \begin{pmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{pmatrix}
 \end{matrix}
 =
 \begin{matrix}
 \downarrow \\
 \mathbf{O}
 \end{matrix}
 \begin{matrix}
 \downarrow \\
 \underbrace{x(0)}_{x_0}
 \end{matrix}
 +
 \begin{matrix}
 \downarrow \\
 \mathbf{T}
 \end{matrix}
 \begin{matrix}
 u_0 - u_{N-1} \\
 \downarrow \\
 \begin{pmatrix} u(0) \\ u(1) \\ \vdots \\ u(N-1) \end{pmatrix}
 \end{matrix}
 \end{matrix}
 \quad
 \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

For a given estimate of  $\theta$ , this is the  $\hat{y}$  vector

$$S(\theta_1, \theta_2, \dots, \theta_q) = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

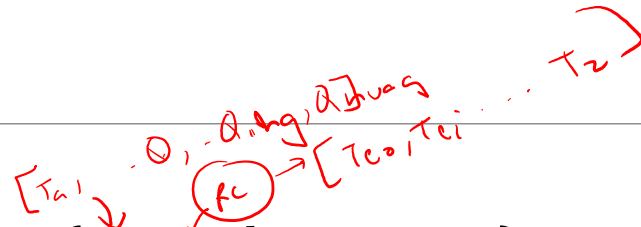
# sum of squared error for state-space models

---

$$\mathcal{O} = \begin{pmatrix} C_\theta \\ C_\theta A_\theta \\ \vdots \\ C_\theta A_\theta^{N-1} \end{pmatrix} \quad \mathcal{J} = \begin{pmatrix} D_\theta & 0 & \dots & \\ C_\theta B_\theta & D_\theta & 0 & \dots \\ \vdots & \vdots & \vdots & \\ C_\theta A_\theta^{N-2} B_\theta & C_\theta A_\theta^{N-3} B_\theta & \dots & C_\theta B_\theta & D_\theta \end{pmatrix}$$

# Nonlinear Estimation

Let  $\mathcal{Z}^N$  be the given data-set  $\{\mathbf{u}_k, \mathbf{x}_0, k = 1, \dots, N\}$



$$\hat{\boldsymbol{\theta}}_N = \hat{\boldsymbol{\theta}}_N(\mathcal{Z}^N) = \arg \min_{\boldsymbol{\theta} \in \Theta} S_N(\boldsymbol{\theta}, \mathcal{Z}^N)$$

$S_N(\boldsymbol{\theta}, \mathcal{Z}^N)$  is the squared error i.e.  $S_N(\boldsymbol{\theta}, \mathcal{Z}^N) = \sum_{k=1}^N \underline{\mathbf{e}_k(\boldsymbol{\theta})} \underline{\mathbf{e}_k^T(\boldsymbol{\theta})}$

$$\mathbf{e}_k(\boldsymbol{\theta}) = \mathbf{y}_k - \hat{\mathbf{y}}_k(\boldsymbol{\theta}) \rightarrow (\mathbf{x}_0, \mathbf{u}_0 - \mathbf{u}_N)$$

Measured  $\uparrow$   $\mathbf{y}_k$        $\hat{\mathbf{y}}_k(\boldsymbol{\theta})$   $\uparrow$  Predicted (for a particular value of  $\boldsymbol{\theta}$ )

# Non-linear least squares

---

We will cover the following methods:

- 1) Steepest descent (or Gradient descent) and Newton's method,
- 2) Gauss Newton and Linearization, and
- 3) Levenberg-Marquardt's procedure.

1. In each case a iterative procedure is used to find the least squares estimators :  $\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_q$
2. That is an initial estimates,  $\theta_1^0, \theta_2^0, \dots, \theta_q^0$  ,for these values are determined.  $((SSE) \downarrow)$
3. Iteratively find better estimates,  $\theta_1^i, \theta_2^i, \dots, \theta_q^i$  that hopefully converge to the least squares estimates,

# Steepest Descent

---

- The steepest descent method focuses on determining the values of  $\theta_1, \theta_2, \dots, \theta_q$  that minimize the sum of squares function,  $S(\theta_1, \theta_2, \dots, \theta_q)$ .
- The basic idea is to determine from an initial point,  $\theta_1^0, \theta_2^0, \dots, \theta_q^0$  and the tangent plane to  $S(\theta_1, \theta_2, \dots, \theta_q)$  at this point, the vector along which the function  $S(\theta_1, \theta_2, \dots, \theta_q)$  will be decreasing at the fastest rate.
- The method of steepest descent then moves from this initial point along the direction of steepest descent until the value of  $S(\theta_1, \theta_2, \dots, \theta_q)$  stops decreasing.

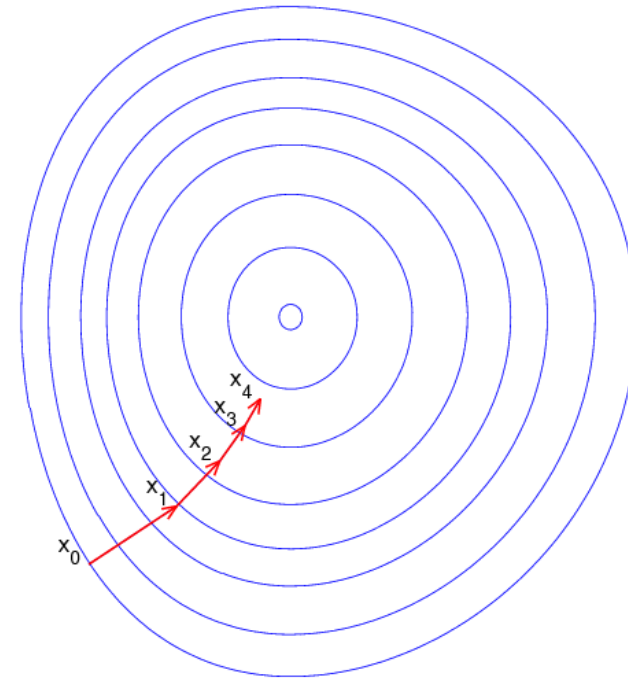
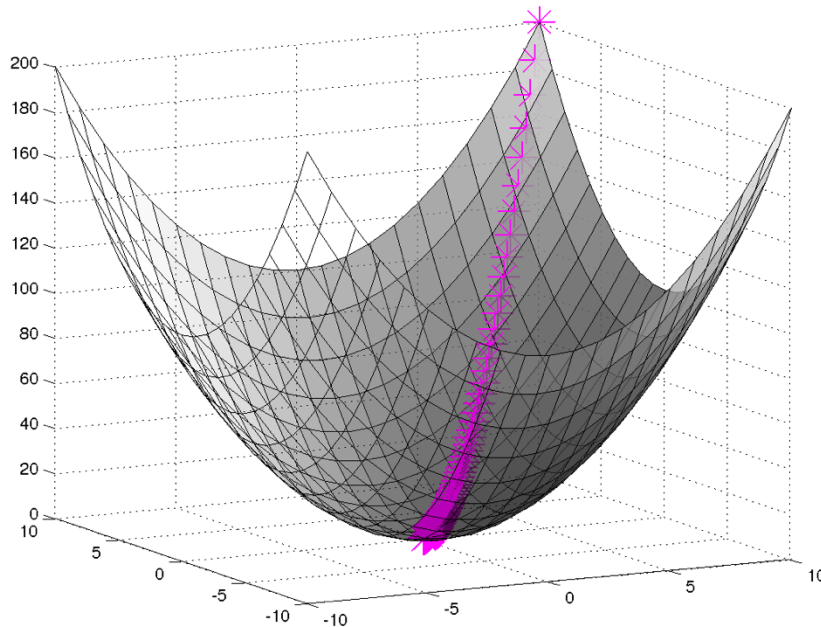
# Steepest Descent

---

- It uses this point,  $\theta_1^1, \theta_2^1, \dots, \theta_q^1$  as the next approximation to the value that minimizes  $S(\theta_1, \theta_2, \dots, \theta_q)$ .
- The procedure then continues until the successive approximations arrive at a point where the sum of squares function,  $S(\theta_1, \theta_2, \dots, \theta_q)$  is minimized.
- At that point, the tangent plane to  $S(\theta_1, \theta_2, \dots, \theta_q)$  will be horizontal and there will be no direction of steepest descent.

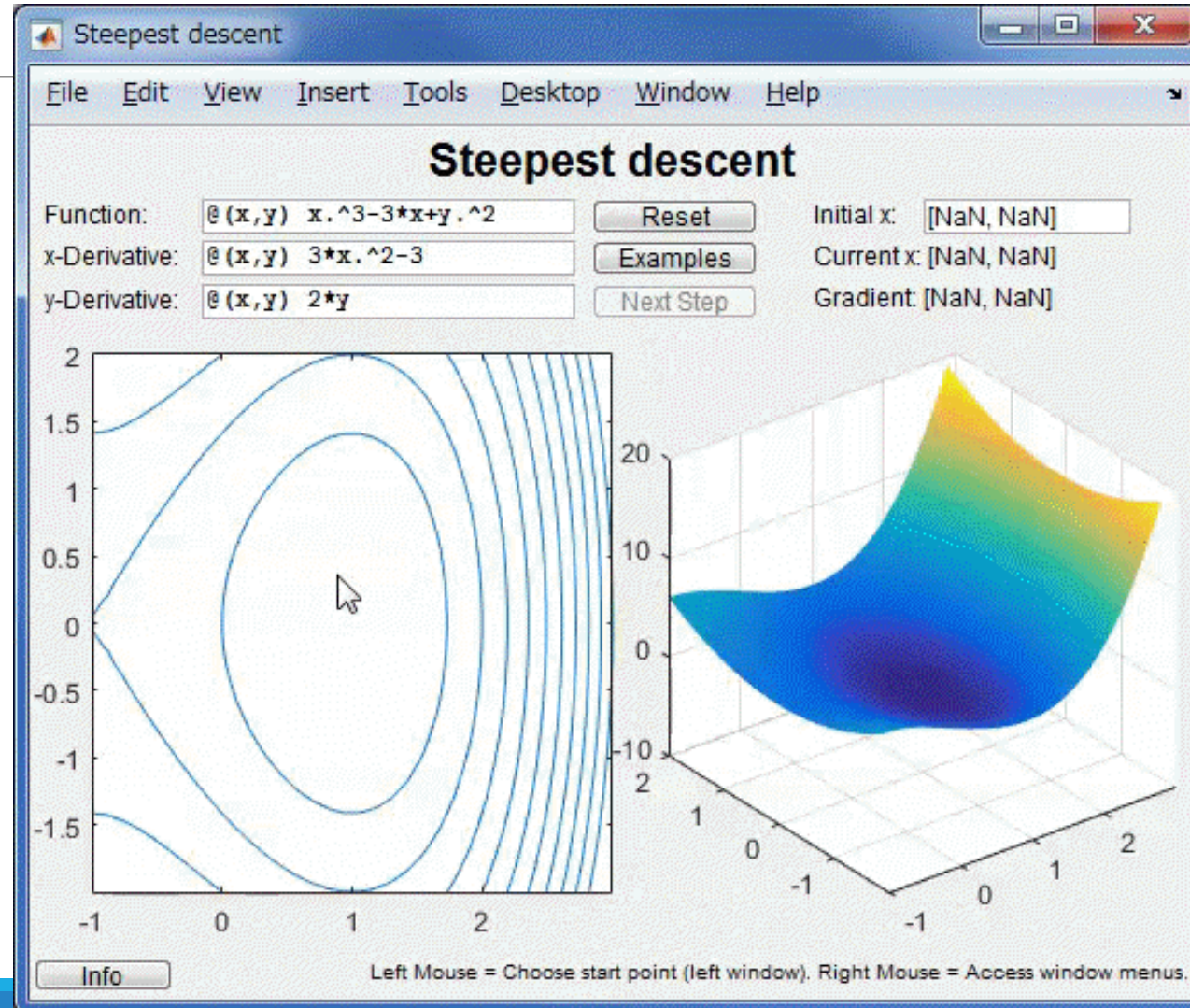
# Steepest Descent

To find a local minimum of a function using steepest descent, one takes steps proportional to the *negative* of the gradient of the function at the current point.





# Steepest Descent



# Steepest Descent

---

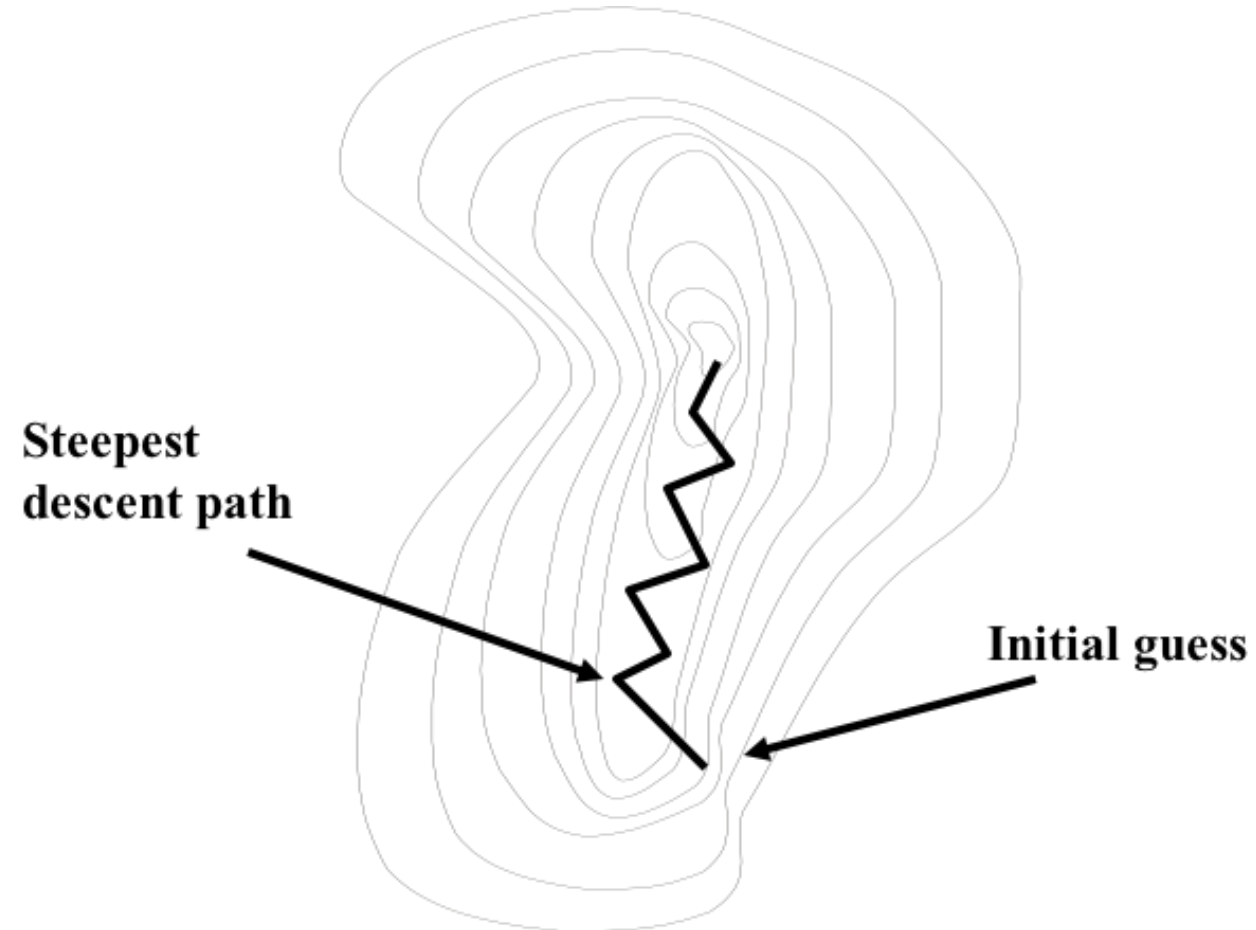
Initialize  $k=0$ , choose  $\theta_0$

While  $k < k_{\max}$

$$\theta_k = \theta_{k-1} - \overbrace{\nabla F(\theta_{k-1})}^{\text{Gradient}}$$

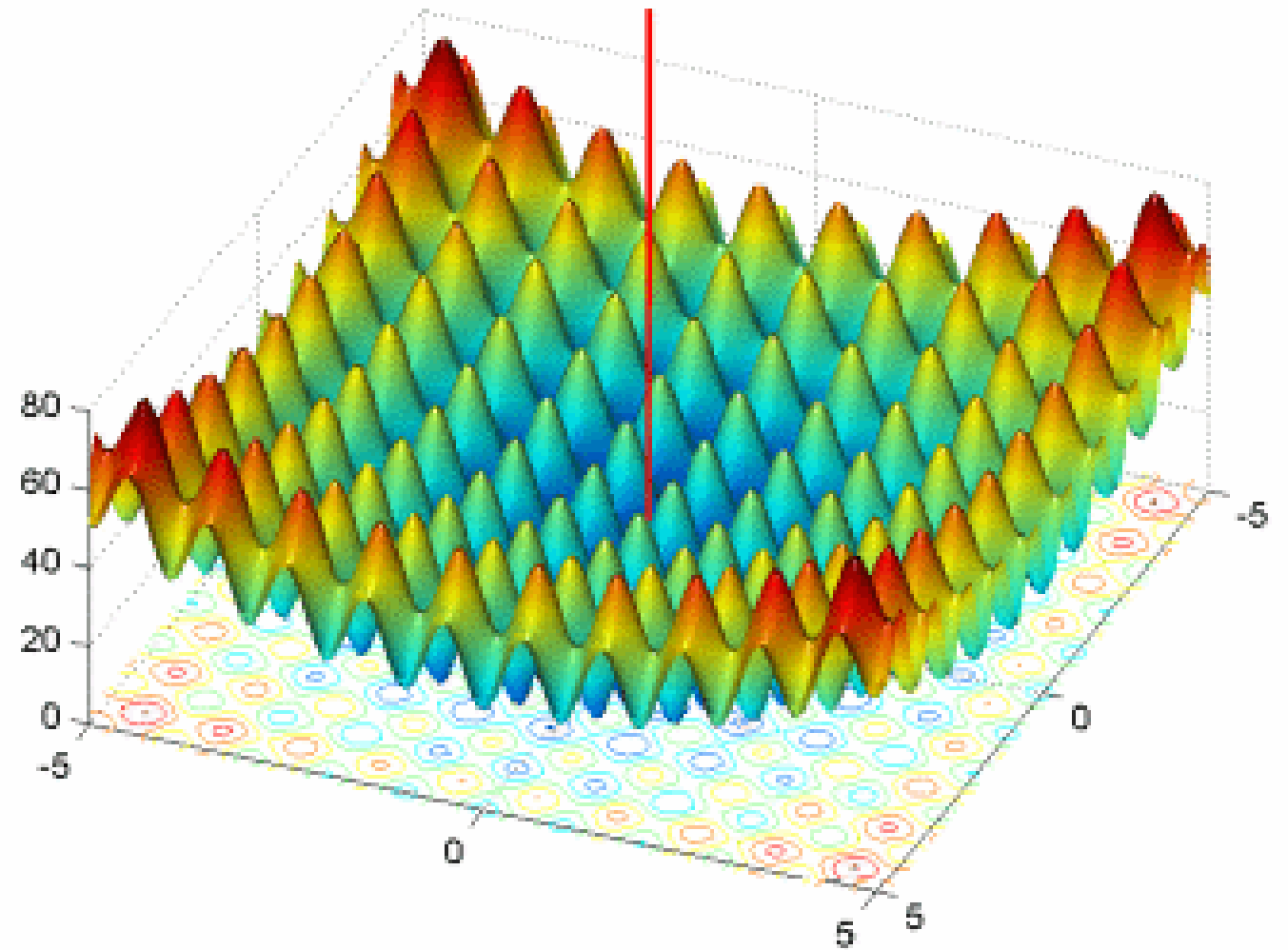
# Steepest Descent

---



# Steepest Descent

---



Gradient descent is a *local* optimization method

# Steepest Descent

---

- While, theoretically, the steepest descent method will converge, it may do so in practice with agonizing slowness after some rapid initial progress.
- **Slow convergence** is particularly likely when the  $S(\theta_1, \theta_2, \dots, \theta_q)$  contours are highly curved and it happens when the path of **steepest descent zigzags** slowly up a narrow ridge, each iteration bringing only a slight reduction in  $S(\theta_1, \theta_2, \dots, \theta_q)$ .
- A further disadvantage of the steepest descent method is that it is **not scale invariant**.
- The steepest descent method is, on the whole, slightly less favored than the linearization method (described later) but will work satisfactorily for many nonlinear problems

# Recall: Least squares in general

---

Most optimization problem can be formulated as a nonlinear least squares problem

$$x^* = \arg \min_x \frac{1}{2} \sum_{i=1}^m (f_i(x))^2$$

$$x^* = \arg \min_x \frac{1}{2} f(x)^T f(x)$$

Sorry for being lazy, we have been denoting Error using  $e$ , and parameter  $\theta$ , and I just switched the notation to  $f$ , and  $f(x)$

Where  $f_i: R^n \mapsto R$ ,  $i=1, \dots, m$  are given functions, and  $m \geq n$

# Newton's Method

---

Quadratic approximation

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2$$

What's the minimum solution of the quadratic approximation

$$\Delta x = -\frac{f'(x)}{f''(x)}$$

# Newton's Method

---

High dimensional case:

$$F(x + \Delta x) \approx F(x) + \nabla F(x)\Delta x + \frac{1}{2}\Delta x^T H(x)\Delta x$$

What's the optimal direction?

$$\Delta x \approx -H(x)^{-1}\nabla F(x)$$



# Terminology

---

The *gradient*  $\nabla f$  of a multivariable function is a vector consisting of the function's partial derivatives:

$$\nabla f(x_1, x_2) = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right)$$

The *Hessian matrix*  $H(f)$  of a function  $f(x)$  is the square matrix of second-order partial derivatives of  $f(x)$ :

$$H(f(x_1, x_2)) = \begin{pmatrix} \frac{\partial f}{\partial x_1^2} & \frac{\partial f}{\partial x_1 \partial x_2} \\ \frac{\partial f}{\partial x_1 \partial x_2} & \frac{\partial f}{\partial x_2^2} \end{pmatrix}$$

# Newton's Method

---

Initialize  $k=0$ , choose  $x_0$

While  $k < k_{\max}$

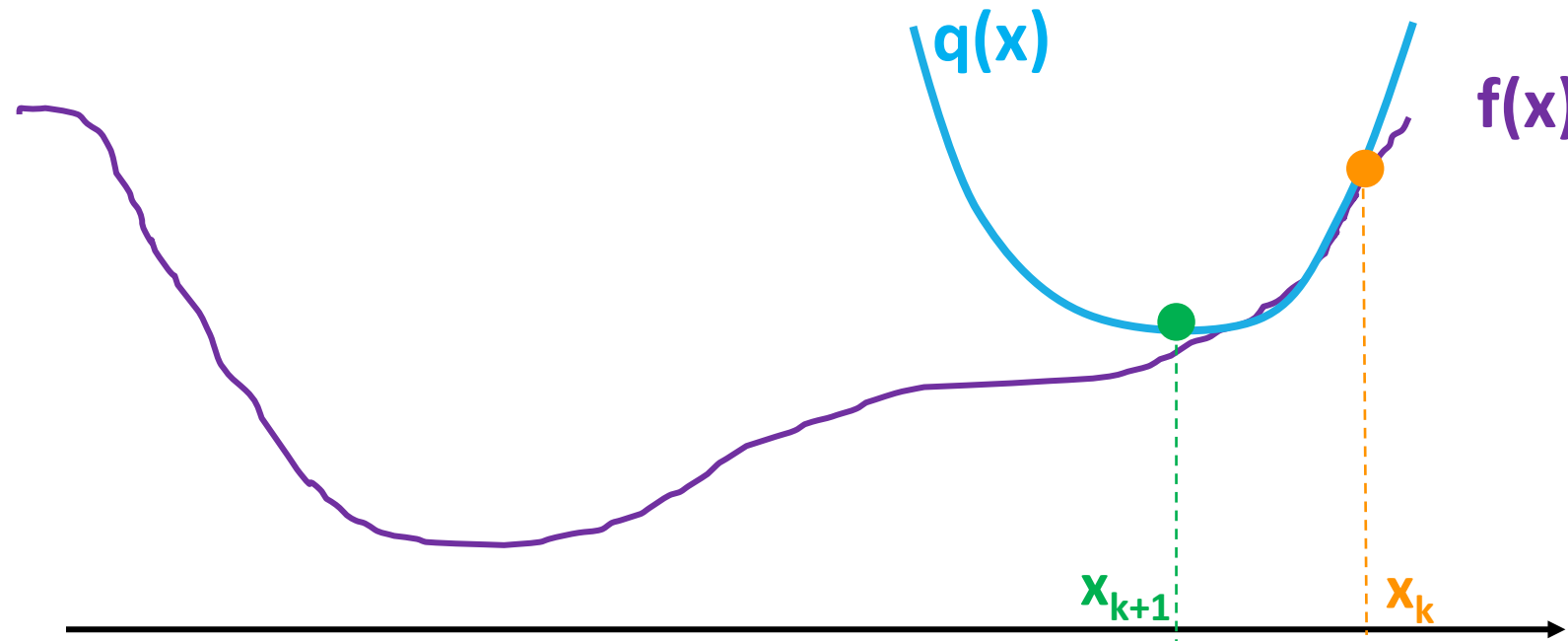
$$x_k = x_{k-1} - \lambda H(x)^{-1} \nabla F(x_{k-1})$$

# Newton's Method

$$\min_x f(x)$$

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

$$x_{k+1} = x_k - H^{-1} \cdot \nabla f$$

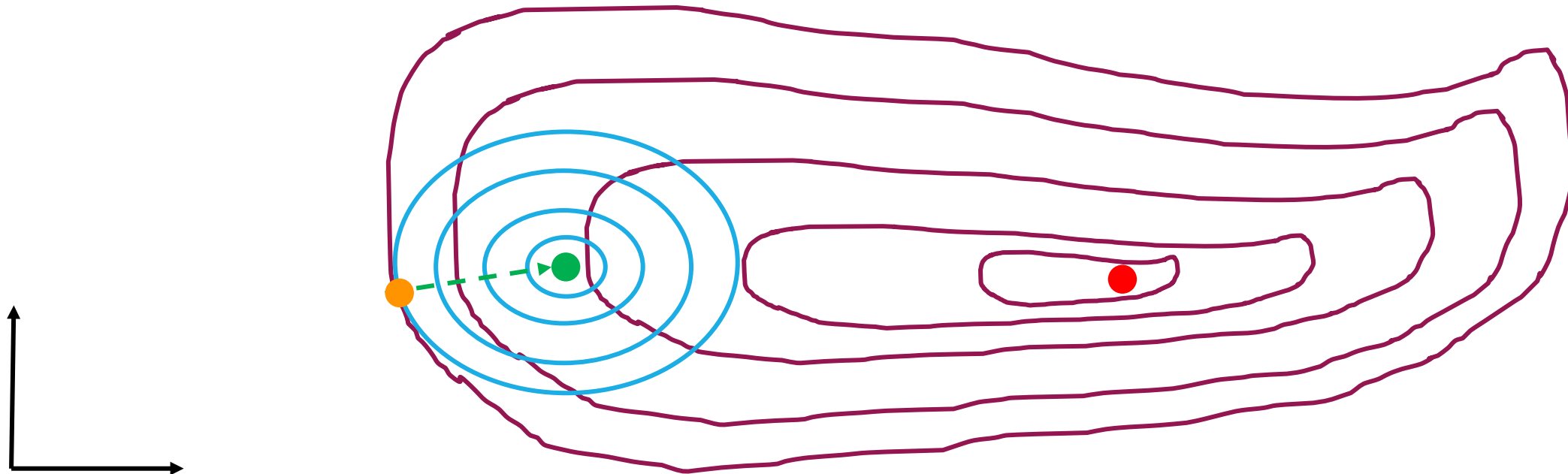


# Newton's Method

$$\min_x f(x)$$

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

$$x_{k+1} = x_k - H^{-1} \cdot \nabla f$$



# Newton's Method

---

Let  $f(x): \mathcal{R}^n \rightarrow \mathcal{R}$  be sufficiently smooth

Taylor's approximation: For close to point 'a'  $f(x) \approx f(a) + g^T(x - a) + \frac{1}{2} \underbrace{(x - a)^T H(x - a)}_{x^T Hx - 2a^T Hx + a^T Ha} + h.o.t.$

$$g = \nabla f(a) \quad H = \nabla^2 f(a)$$

$$q(x) = \frac{1}{2} x^T Hx + b^T x + c \quad \text{where} \quad b = g - HTa$$

$$\nabla q = 0 \Rightarrow Hx + b = 0 \Rightarrow x = -H^{-1}b = -H^{-1}g + a = a - H^{-1}g$$

$$x = a - H^{-1}g \implies x_{k+1} = x_k - H^{-1} \cdot \nabla f$$

# Newton's Method

---

$$\nabla q = 0 \Rightarrow Hx + b$$

*For minima*

$$\nabla^2 q > 0$$

$$\nabla^2 q = H$$

*Minima if H is PSD*

1) Initialize:  $x_0$

2) Iterate:  $x_{k+1} = x_k - H^{-1} \cdot g$

$$g = \nabla f(x_k) \quad H = \nabla^2 f(x_k)$$

1) H may fail to be PSD

2) H may not be invertible.

3) Difficult to compute H in practice through numerical methods

# Recall: Non-linear least squares

---

$$f(x) = \sum_{j=1}^N (r_j(x))^2 = \|r(x)\|_2^2$$

The  $j$ -th component of the vector  $r(x)$  is the residual

$$r_j(x) = y_j - \hat{y}_j$$

$$r(x) = (r_1(x), r_2(x), \dots, r_N(x))^T$$

# Non-linear least squares

---

The *Jacobian*  $J(x)$  is a matrix of all  $\nabla r_j(x)$ :

$$J(x) = \left[ \frac{\partial r_j}{\partial x_i} \right]_{j=1, \dots, N; i=1, \dots, n} = \begin{bmatrix} \nabla r_1(x)^T \\ \nabla r_2(x)^T \\ \vdots \\ \nabla r_N(x)^T \end{bmatrix}$$



# Non-linear least squares

---

$$\nabla f(x) = \sum_{j=1}^N r_j(x) \nabla r_j(x) = J(x)^T r(x)$$

$$\nabla^2 f(x) = \sum_{j=1}^N \nabla r_j(x) \nabla r_j(x)^T + \sum_{j=1}^N r_j(x) \nabla^2 r_j(x)$$

$$= J(x)^T J(x) + \sum_{j=1}^N r_j(x) \nabla^2 r_j(x)$$

# Gauss-Newton Method

---

Use the approximation  $\nabla^2 f_k \approx J_k^T J_k$

$J_k$  must have full rank

Requires accurate initial guess

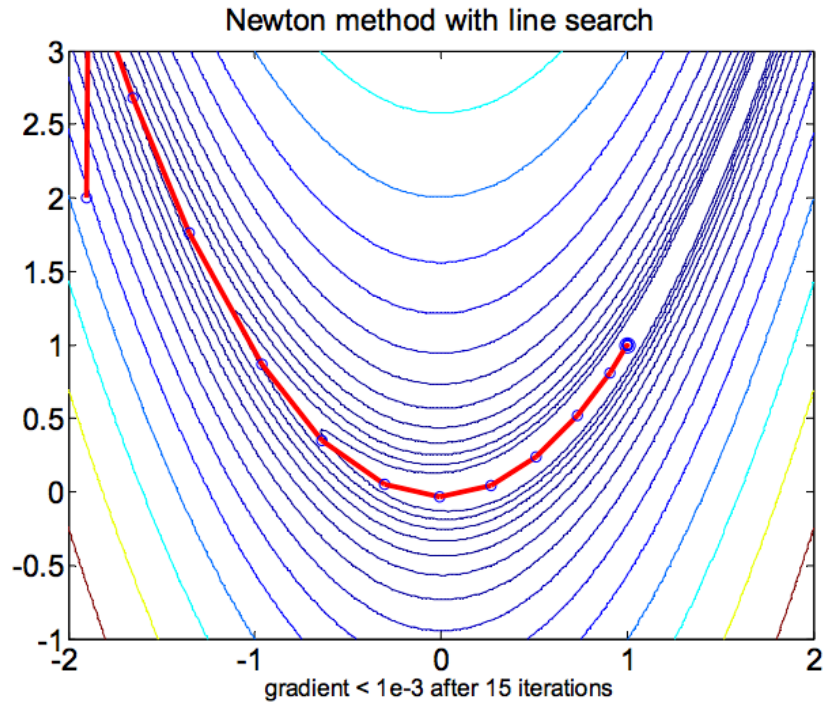
Fast convergence close to solution

$$J(x)^T J(x) + \sum_{j=1}^N r_j(x) \nabla^2 r_j(x)$$

Residuals are small when close to the optimal

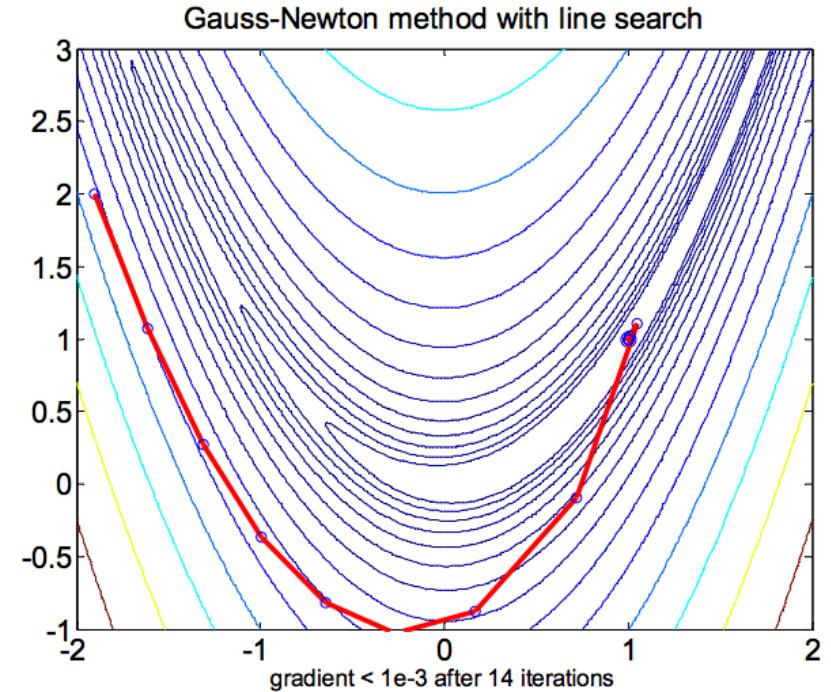
# Comparison

## Newton



- requires computing Hessian (i.e.  $n^2$  second derivatives)
- exact solution if quadratic

## Gauss-Newton



- approximates Hessian by Jacobian product
- requires only  $n$  first derivatives

## Newton's method cannot use negative curvature

---

- We can progress if we use a positive definite approximation of the Hessian matrix of  $f(x)$ .  
$$x_{k+1} = x_k - H^{-1} \cdot g$$
- One possibility is to approximate  $H$  by the identity matrix  $I$  (always PD)
  - This will be the same as steepest descent:  $x_{k+1} = x_k - \Delta g$
  - Too slow, + convergence issues
- Instead use  $\tilde{H} = H_k + \lambda I$ 
  - High value of  $\lambda$  == steepest (gradient) descent.
  - Low value == Newton or Gauss Newton method

# Levenberg-Marquardt Method

---

- Mixture of Gauss-Newton and Gradient descent.
- Acts like Gauss-Newton when close to the minimum (quadratic region)
- Gradient descent when improvement is difficult.
- Depends on a parameter  $\lambda$  which
  1. Controls the mixture of Gauss-Newton and Gradient Descent
  2. Controls the step-length.

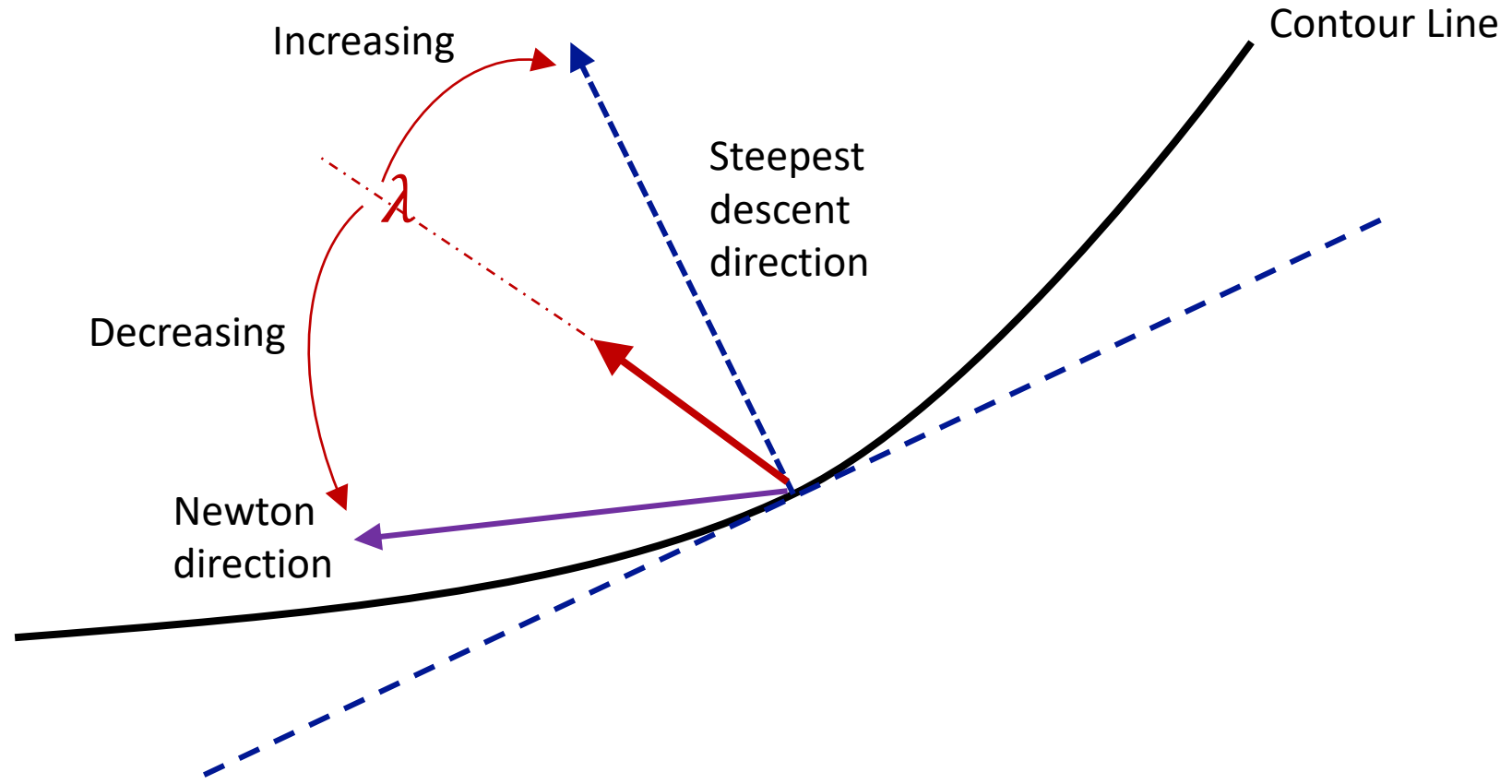


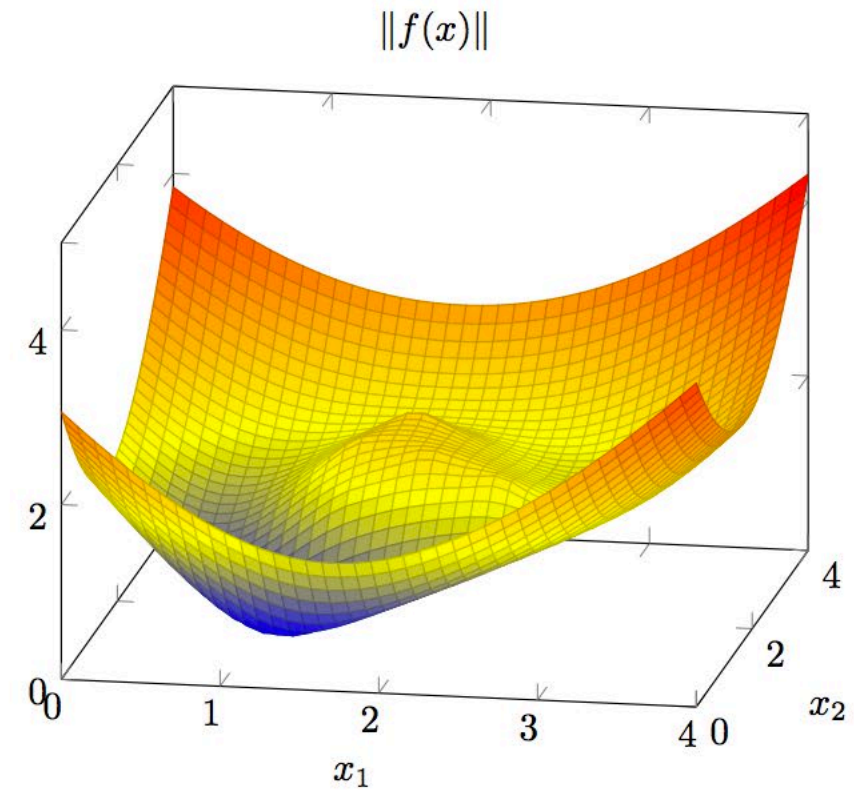
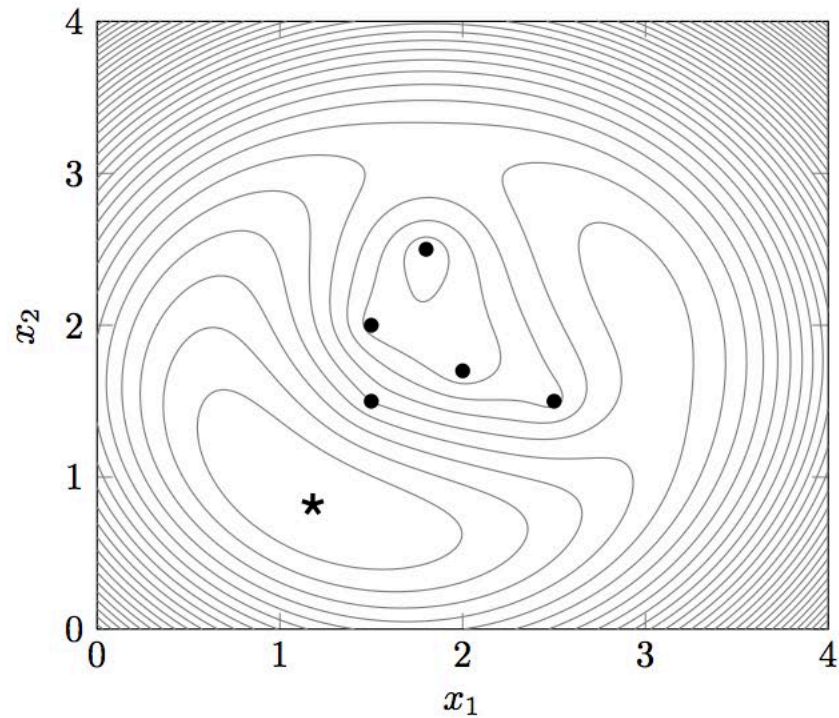
Illustration of Levenberg-Marquardt gradient descent

# Levenberg-Marquardt Method

---

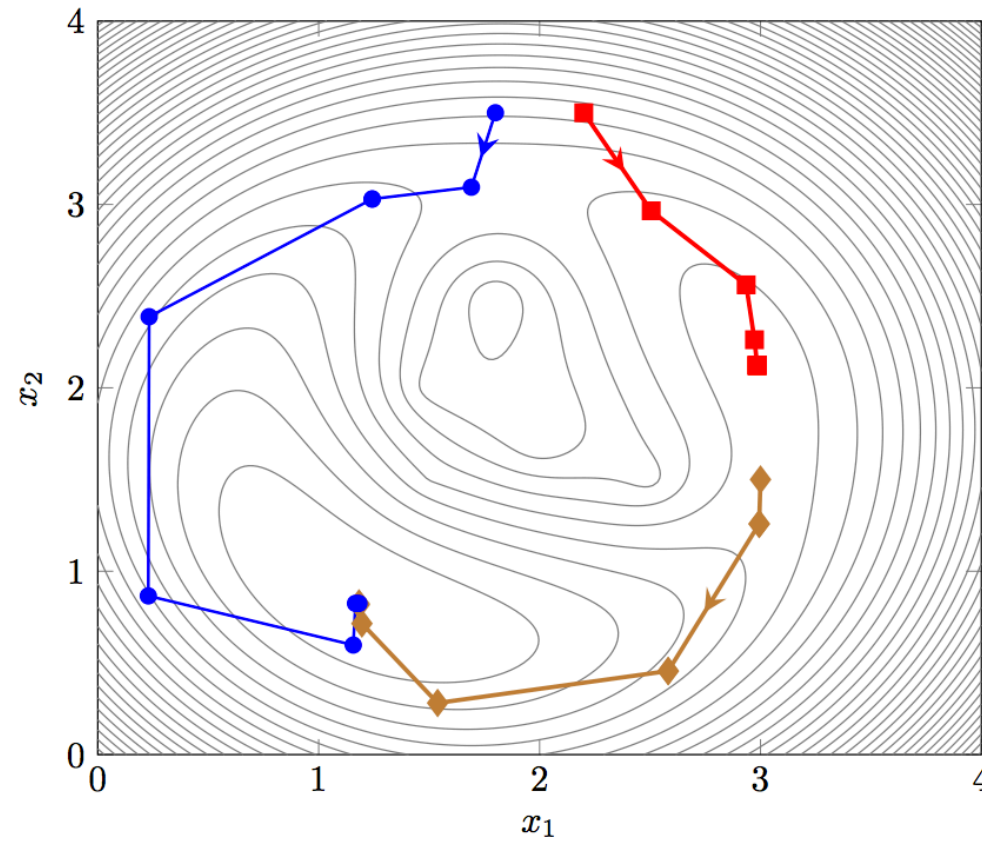
- 1) Adapt the value of  $\lambda$  during the optimization.
- 2) If the iteration was successful ( $F(x_{k+1}) < F(x_k)$ )
  - a) Decrease  $\lambda$  and try to use as much curvature information as possible.
- 3) If the previous iteration was unsuccessful ( $F(x_{k+1}) > F(x_k)$ )
  - a) Increase  $\lambda$  and use only basic gradient information.
- 4) Trust Region Algorithm

# Levenberg-Marquardt Method

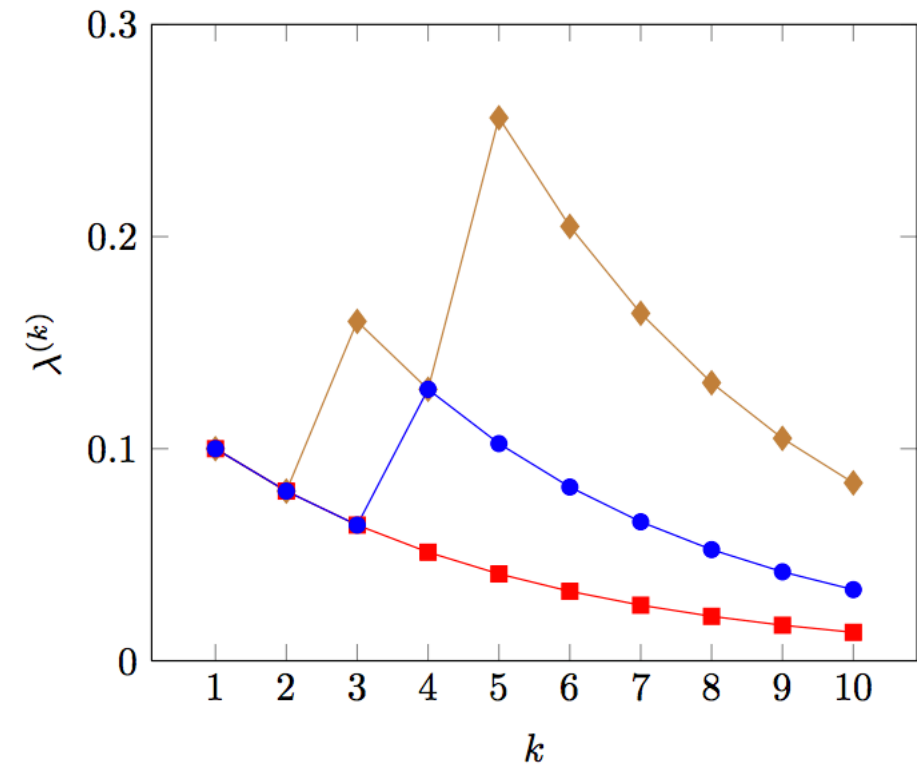
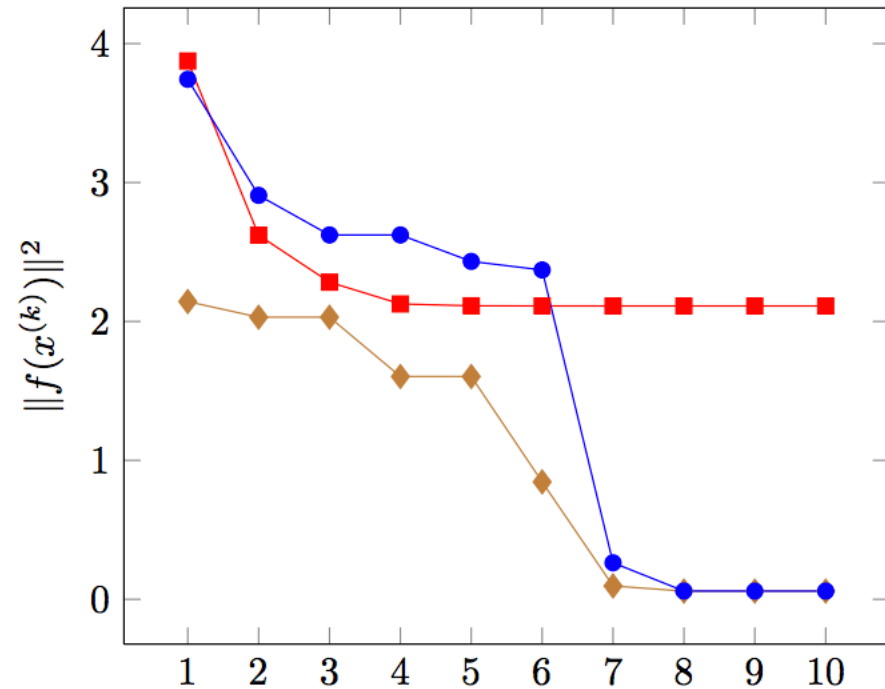




# Levenberg-Marquardt from 3 initial points



# Levenberg-Marquardt from 3 initial points



# Stopping Criteria

---

**Criterion 1:** reach the number of iteration specified by the user

$$K > k_{\max}$$

# Stopping Criteria

---

**Criterion 1:** reach the number of iteration specified by the user

$$K > k_{\max}$$

**Criterion 2:** when the current function value is smaller than a user-specified threshold

$$F(x_k) < \sigma_{\text{user}}$$

# Stopping Criteria

---

**Criterion 1:** reach the number of iteration specified by the user

$$K > k_{\max}$$

**Criterion 2:** when the current function value is smaller than a user-specified threshold

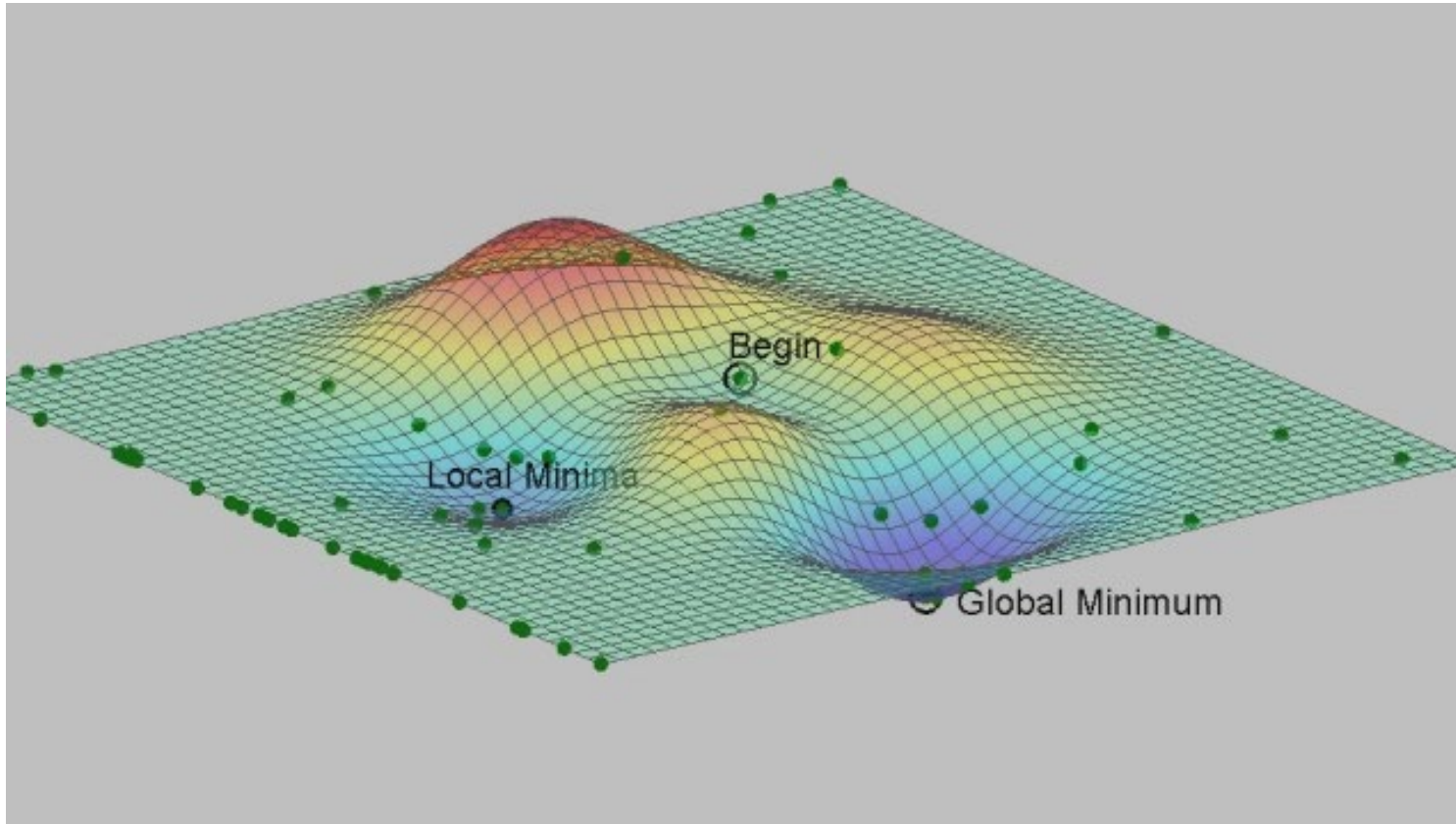
$$F(x_k) < \sigma_{\text{user}}$$

**Criterion 3:** when the change of function value is smaller than a user specified threshold

$$||F(x_k) - F(x_{k-1})|| < \epsilon_{\text{user}}$$

# Multi-start search

---



- Several points as initial guesses for regression and the regression is performed for each point.
- 1) Choose randomly..
  - 2) Choose within some neighborhood of nominal values.

# NLLS in Matlab

---

## **nlinfit**

---

Nonlinear regression

## **lsqnonlin**

---

Solve nonlinear least-squares (nonlinear data-fitting) problems

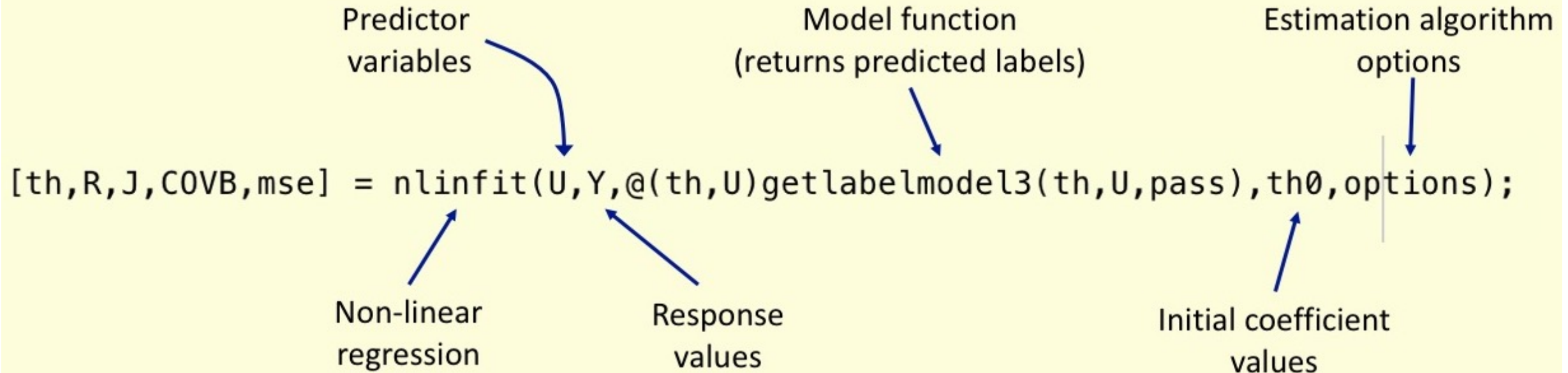
## **lsqcurvefit**

---

Solve nonlinear curve-fitting (data-fitting) problems in least-squares sense

# Example: nlinfit

---





# Example: nlinfit

Predictor variables

Model function  
(returns predicted labels)

Estimation algorithm options

```
[th,R,J,COVB,mse] = nlinfit(U,Y,@(th,U)getlabelmodel3(th,U,pass),th0,options);
```

$$\mathbf{u}_b^T = [ \dot{Q}_b \quad T_a \quad T_g \quad \dot{Q}_{sol,c} \quad \dot{Q}_{sol,e} \quad \dot{Q}_{rad,c} \quad \dot{Q}_{rad,e} \quad \dot{Q}_{sol,trans} \quad \dot{Q}_{conv} ]$$

# Example: nlinfit

Predictor variables

Model function  
(returns predicted labels)

Estimation algorithm options

```
[th,R,J,COVB,mse] = nlinfit(U,Y,@(th,U)getlabelmodel3(th,U,pass),th0,options);
```

$$\begin{pmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{pmatrix} = \mathbf{O} x(0) + \mathbf{J} \begin{pmatrix} u(0) \\ u(1) \\ \vdots \\ u(N-1) \end{pmatrix} \quad \mathbf{O} = \begin{pmatrix} C_\theta \\ C_\theta A_\theta \\ \vdots \\ C_\theta A_\theta^{N-1} \end{pmatrix} \quad \mathbf{J} = \begin{pmatrix} D_\theta & 0 & \dots & \dots \\ C_\theta B_\theta & D_\theta & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots \\ C_\theta A_\theta^{N-2} B_\theta & C_\theta A_\theta^{N-3} B_\theta & \dots & C_\theta B_\theta & D_\theta \end{pmatrix}$$

# Example: nlinfit

$$\theta_1 = [C_{e1} \ C_{i1} \ C_{c1} \ C_{g1} \ R_{e1} \ R_{e2} \ R_{i1} \ R_{i2} \ R_{c1} \ R_{c2} \ R_{g1} \ R_{g1} \ R_{g2} \ C_{e2} \ C_{i2} \ C_{c2} \ C_{g2} \ R_{e3} \ R_{i3} \ R_{c3} \ R_{g3} ]$$

```
[th,R,J,COVB,mse] = nlinfit(U,Y,@(th,U)getlabelmodel3(th,U,pass),th0,options);
```

Non-linear  
regression

Response  
values

Initial coefficient  
values