# Optimization Methods

**Categorization of Optimization Problems**

- **Continuous Optimization**
- **Discrete Optimization**
- **Combinatorial Optimization**
- **Variational Optimization**

**Common Optimization Concepts in Computer Vision**

- **Energy Minimization**
- **Graphs**
- **Markov Random Fields**

**Several general approaches to optimization are as follows:**

- **Analytical methods**
- **Graphical methods**
- **Experimental methods**
- **Numerical methods**

**Several branches of mathematical programming have evolved, as follows:**

- **Linear programming**
- **Integer programming**
- **Quadratic programming**
- **Nonlinear programming**
- **Dynamic programming**

**1. THE OPTIMIZATION PROBLEM**
**1.1 Introduction**
**1.2 The Basic Optimization Problem**

**2. BASIC PRINCIPLES**
**2.1 Introduction**
**2.2 Gradient Information**

**3. GENERAL PROPERTIES OF ALGORITHMS**
**3.5 Descent Functions**
**3.6 Global Convergence**
**3.7 Rates of Convergence**

**4. ONE-DIMENSIONAL OPTIMIZATION**
**4.3 Fibonacci Search**
**4.4 Golden-Section Search2**
**4.5 Quadratic Interpolation Method**

**5. BASIC MULTIDIMENSIONAL GRADIENT METHODS**
**5.1 Introduction**
**5.2 Steepest-Descent Method**
**5.3 Newton Method**
**5.4 Gauss-Newton Method**

**CONJUGATE-DIRECTION METHODS**
**6.1 Introduction**
**6.2 Conjugate Directions**
**6.3 Basic Conjugate-Directions Method**
**6.4 Conjugate-Gradient Method**
**6.5 Minimization of Nonquadratic Functions**

**QUASI-NEWTON METHODS**
**7.1 Introduction**
**7.2 The Basic Quasi-Newton Approach**

**MINIMAX METHODS**
**8.1 Introduction**
**8.2 Problem Formulation**
**8.3 Minimax Algorithms**

**APPLICATIONS OF UNCONSTRAINED OPTIMIZATION**
**9.1 Introduction**
**9.2 Point-Pattern Matching**

**FUNDAMENTALS OF CONSTRAINED OPTIMIZATION**
**10.1 Introduction**
**10.2 Constraints**

**LINEAR PROGRAMMING PART I: THE SIMPLEX METHOD**
**11.1 Introduction**
**11.2 General Properties**
**11.3 Simplex Method**

**LINEAR PROGRAMMING PART I: THE SIMPLEX METHOD**
**11.1 Introduction**
**11.2 General Properties**
**11.3 Simplex Method**

**QUADRATIC AND CONVEX PROGRAMMING**
**13.1 Introduction**
**13.2 Convex QP Problems with Equality Constraints**
**13.3 Active-Set Methods for Strictly Convex QP Problems**

**GENERAL NONLINEAR OPTIMIZATION PROBLEMS**
**15.1 Introduction**
**15.2 Sequential Quadratic Programming Methods**

# Problem specification

Suppose we have a cost function (or objective function)

$$f(\mathbf{x}) : \mathbb{R}^N \longrightarrow \mathbb{R}$$

Our aim is to find values of the parameters (decision variables) $\mathbf{x}$ that minimize this function

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x})$$

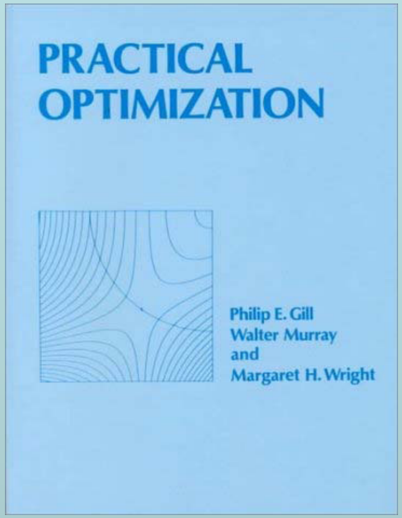
Subject to the following constraints:

- equality: $c_i(\mathbf{x}) = 0$
- nonequality: $c_j(\mathbf{x}) \geq 0$

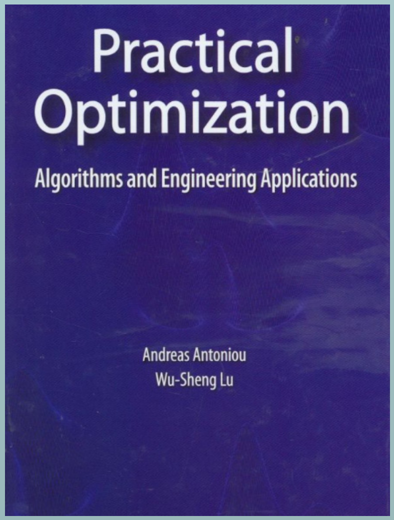
If we seek a maximum of $f(\mathbf{x})$ (profit function) it is equivalent to seeking a minimum of $-f(\mathbf{x})$

# Books to read

- **Practical Optimization**
  - Philip E. Gill, Walter Murray, and Margaret H. Wright, Academic Press,
    1981




- **Practical Optimization: Algorithms and Engineering Applications**
  - Andreas Antoniou and Wu-Sheng Lu
    2007



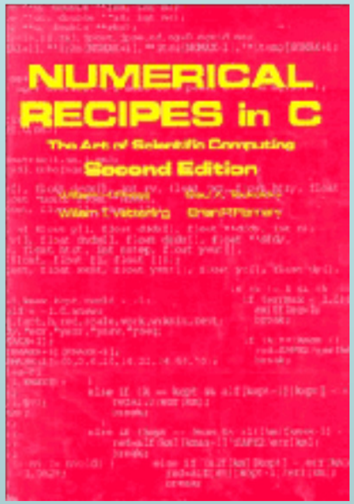

- Both cover unconstrained and constrained optimization. Very clear and comprehensive.

# Further reading and web resources

- **Numerical Recipes in C (or C++) : The Art of Scientific Computing**
  - William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling
  - Good chapter on optimization
  - Available on line at

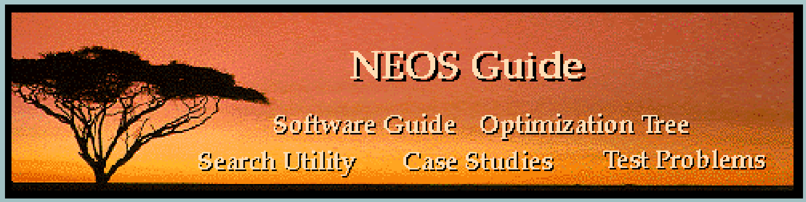
  (1992 ed.) www.nrbook.com/a/bookcpdf.php

  (2007 ed.) www.nrbook.com




- NEOS Guide

  www-fp.mcs.anl.gov/OTC/Guide/




- This powerpoint presentation

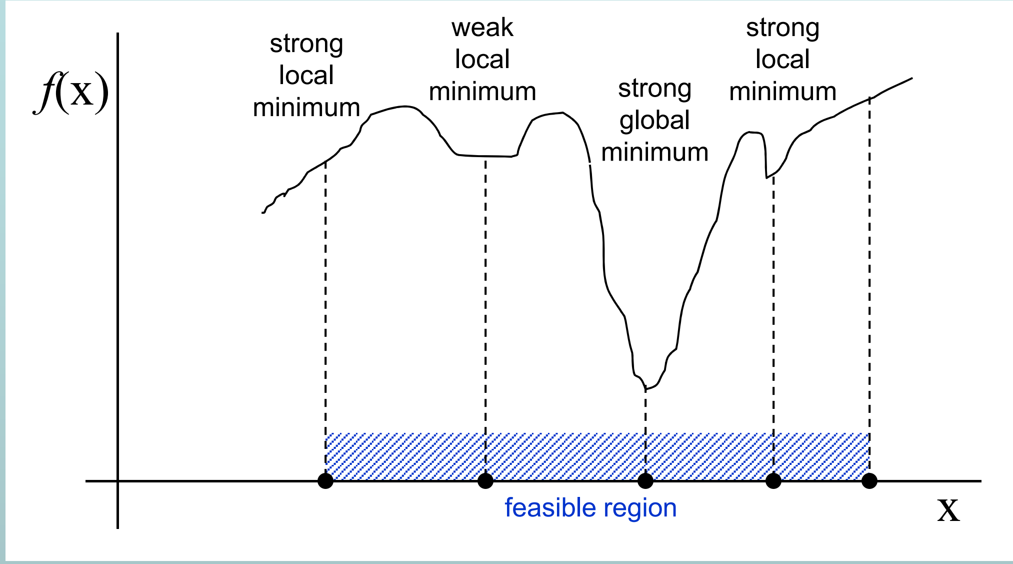
  www.utia.cas.cz

## Introductory Items in OPTIMIZATION

- Category of Optimization methods
- Constrained vs Unconstrained
- Feasible Region

- Gradient and Taylor Series Expansion
- Necessary and Sufficient Conditions
- Saddle Point
- Convex/concave functions
- 1-D search – Dichotomous, Fibonacci – Golden Section, DSC;
- Steepest Descent; Newton; Gauss-Newton
- Conjugate Gradient;
- Quasi-Newton; Minimax;
- Lagrange Multiplier; Simplex; Prinal-Dual, Quadratic programming; Semi-definite;

# Types of minima

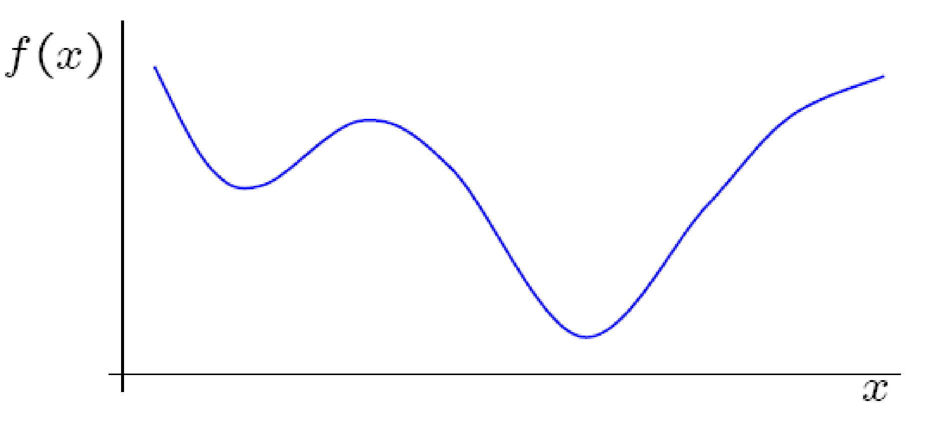



- which of the minima is found depends on the starting point
- such minima often occur in real applications

# Unconstrained univariate optimization

Assume we can start close to the global minimum

$$\min_x f(x)$$




How to determine the minimum?

- Search methods (Dichotomous, Fibonacci, Golden-Section)
- Approximation methods
  1. Polynomial interpolation
  2. Newton method
- Combination of both (alg. of Davies, Swann, and Campey)

# Search methods

- Start with the interval ("bracket") $[x_L, x_U]$ such that the minimum $x^*$ lies inside.
- Evaluate $f(x)$ at two point inside the bracket.
- Reduce the bracket.
- Repeat the process.

- Can be applied to any function and differentiability is not essential.

| XL | XU | X1 | Criteria |
|---|---|---|---|
| 0 | 1 | 1/2 | FA > FB |

| XL | XU | RANGE |
|---|---|---|

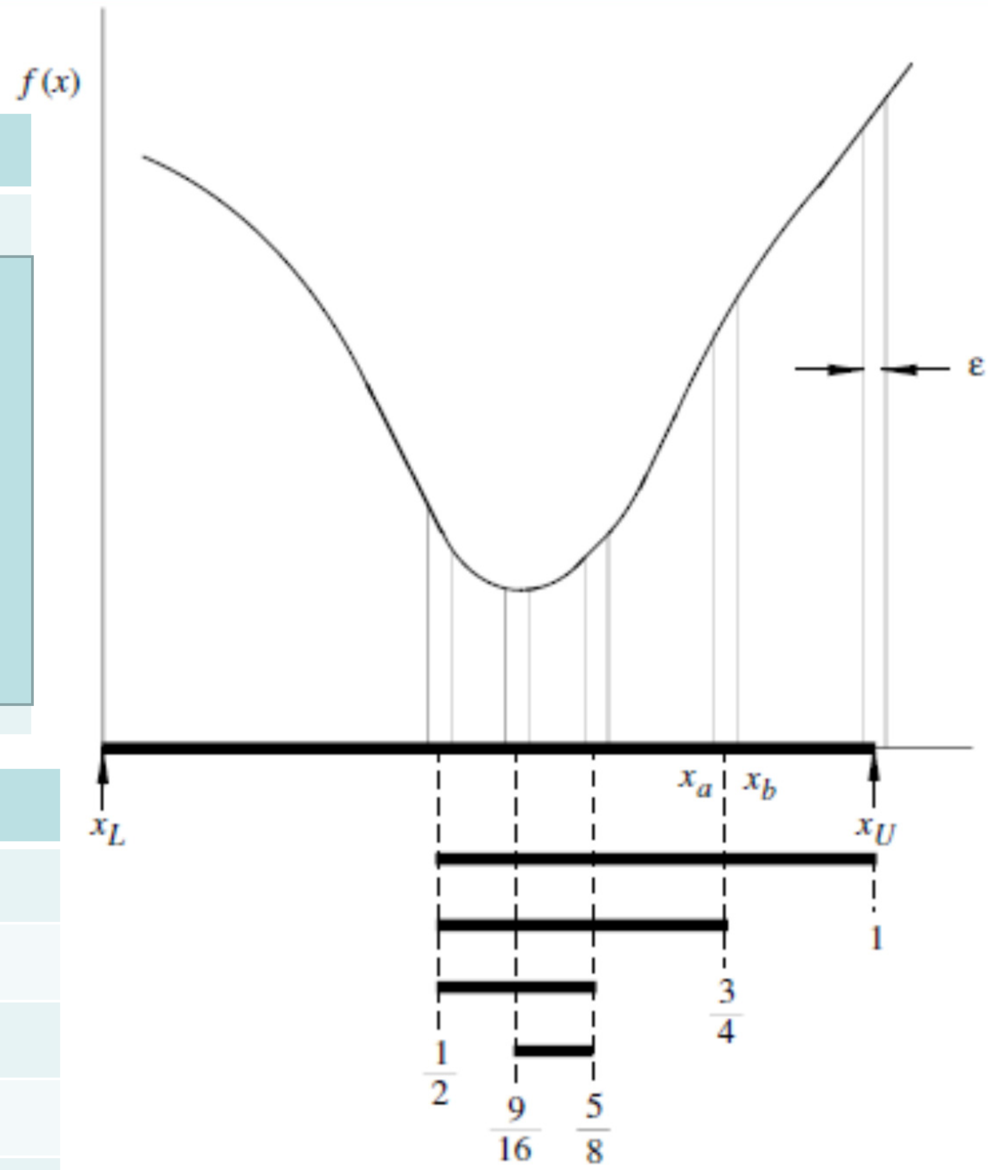
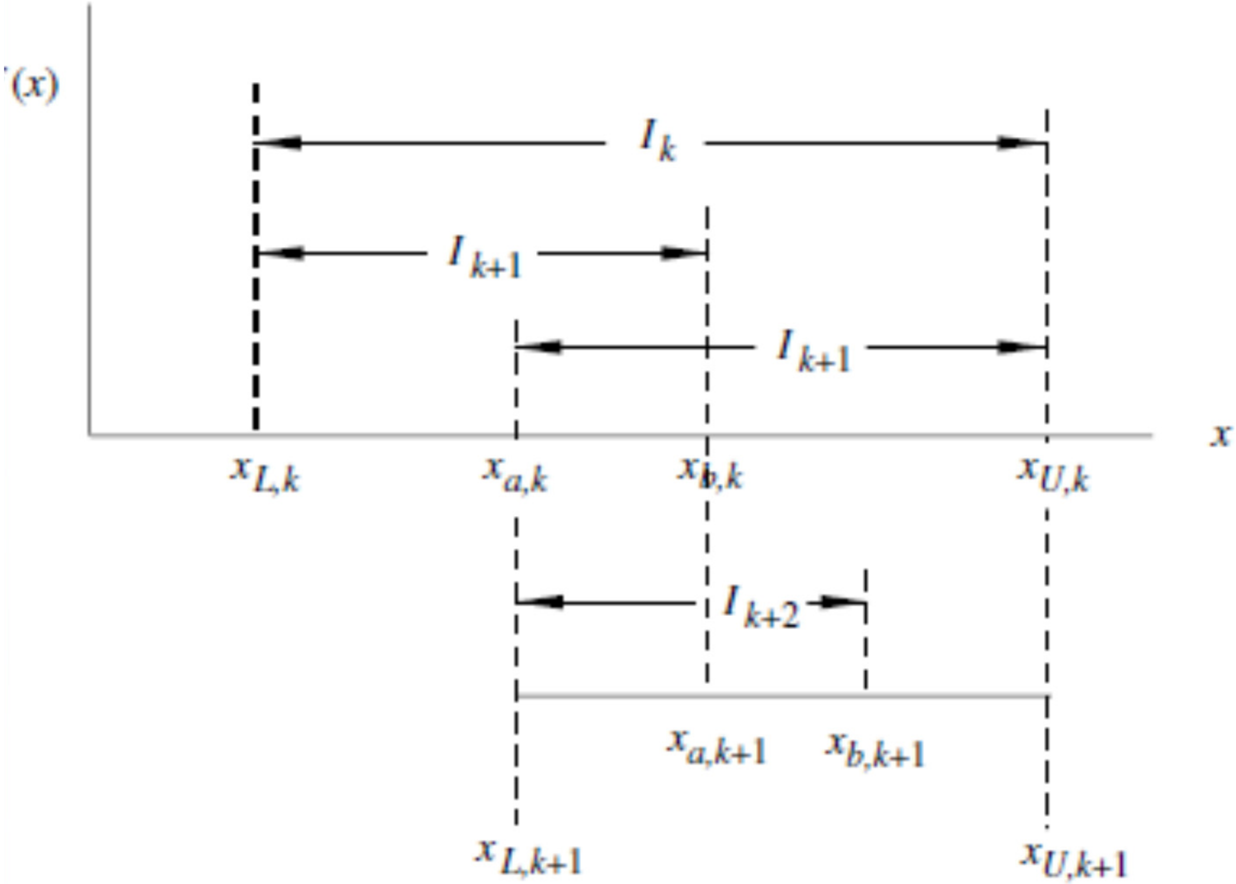



*Figure 4.2.* Construction for dichotomous search.

$$I_{k+2} = \frac{F_{n-k-1}}{F_{n-k}} I_{k+1}$$




If $f_{a,k} > f_{b,k}$, then $x^*$ is in interval $[x_{a,k},\ x_{U,k}]$ and so the new bounds of $x^*$ can be updated as

$$x_{L,k+1} = x_{a,k} \tag{4.7}$$

$$x_{U,k+1} = x_{U,k} \tag{4.8}$$

$$x_{a,k+1} = x_{b,k}$$
$$x_{b,k+1} = x_{L,k+1} + I_{k+2}$$

$$f_{a,k+1} = f_{b,k}$$
$$f_{b,k+1} = f(x_{b,k+1})$$

**Algorithm 4.1 Fibonacci search**

**Step 1**

Input $x_{L,1}$, $x_{U,1}$, and $n$.

**Step 2**

Compute $F_1$, $F_2$, $\ldots$, $F_n$ using Eq. (4.4).

**Step 3**

Assign $I_1 = x_{U,1} - x_{L,1}$ and compute

$$I_2 = \frac{F_{n-1}}{F_n} I_1 \quad \text{(see Eq. (4.6))}$$

$$x_{a,1} = x_{U,1} - I_2, \quad x_{b,1} = x_{L,1} + I_2$$

$$f_{a,1} = f(x_{a,1}), \quad f_{b,1} = f(x_{b,1})$$

Set $k = 1$.

**Step 4**

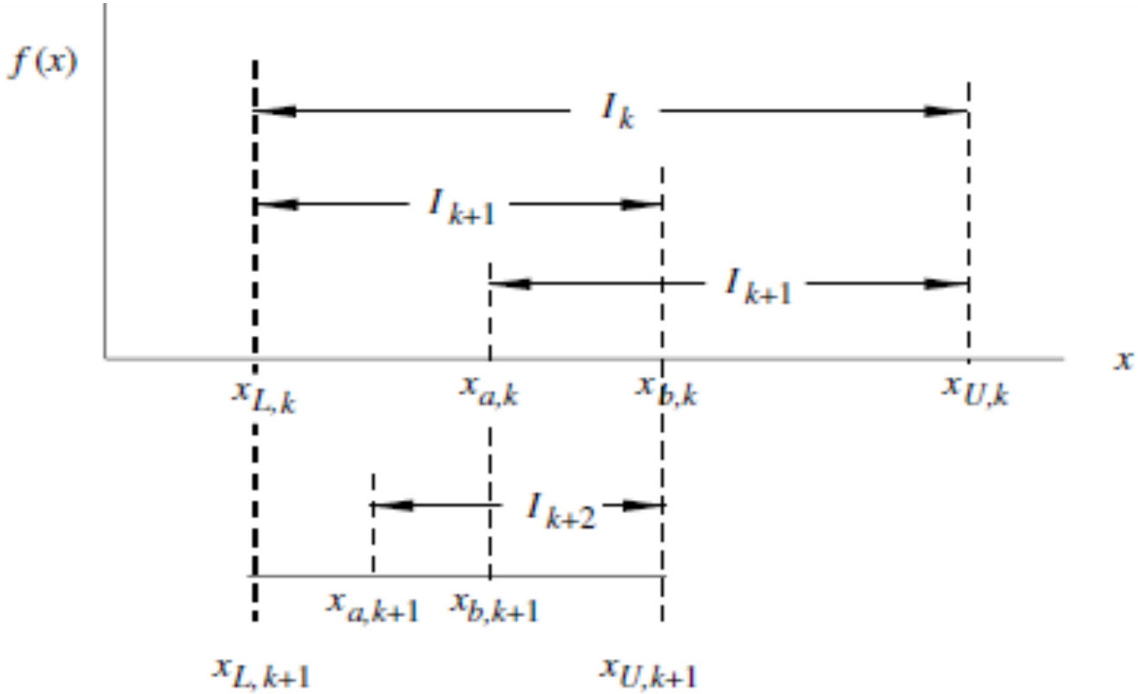
Compute $I_{k+2}$ using Eq. (4.6).

If $f_{a,k} \geq f_{b,k}$, then update $x_{L,k+1}$, $x_{U,k+1}$, $x_{a,k+1}$, $x_{b,k+1}$, $f_{a,k+1}$, and $f_{b,k+1}$ using Eqs. (4.7) to (4.12). Otherwise, if $f_{a,k} < f_{b,k}$, update information using Eqs. (4.13) to (4.18).

**Step 5**

If $k = n - 2$ or $x_{a,k+1} > x_{b,k+1}$, output $x^* = x_{a,k+1}$ and $f^* = f(x^*)$, and stop. Otherwise, set $k = k + 1$ and repeat from Step 4.

The condition $x_{a,k+1} > x_{b,k+1}$ implies that $x_{a,k+1} \approx x_{b,k+1}$ within



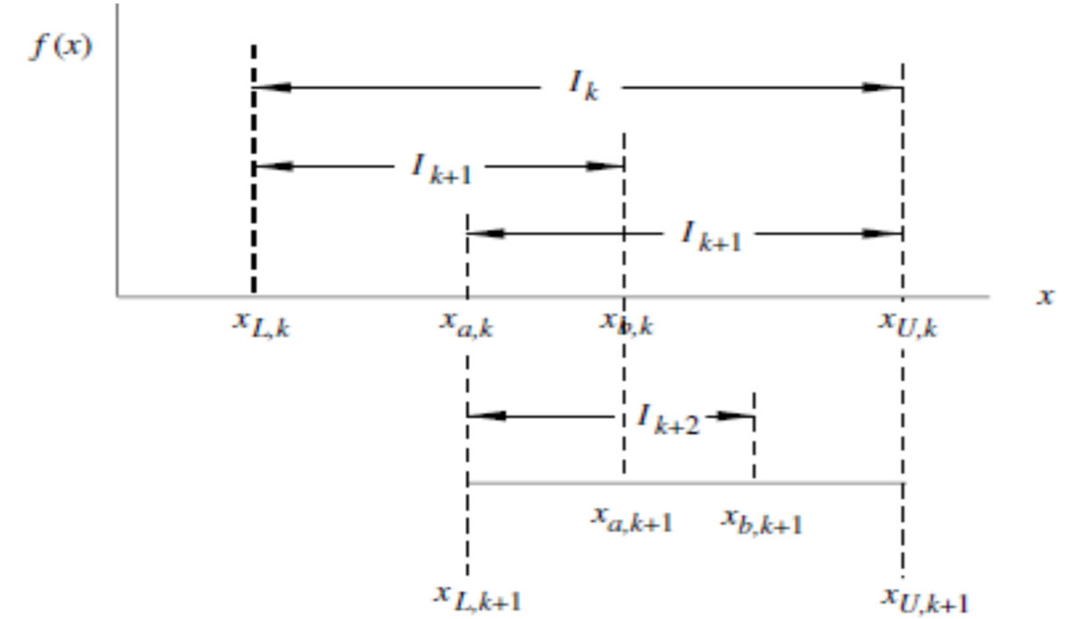

*ure 4.6.* Assignments in $k$th iteration of the fibonacci search if $f_{a,k} < f_{b,k}$

$$I_{k+2} = \frac{F_{n-k-1}}{F_{n-k}} I_{k+1}$$

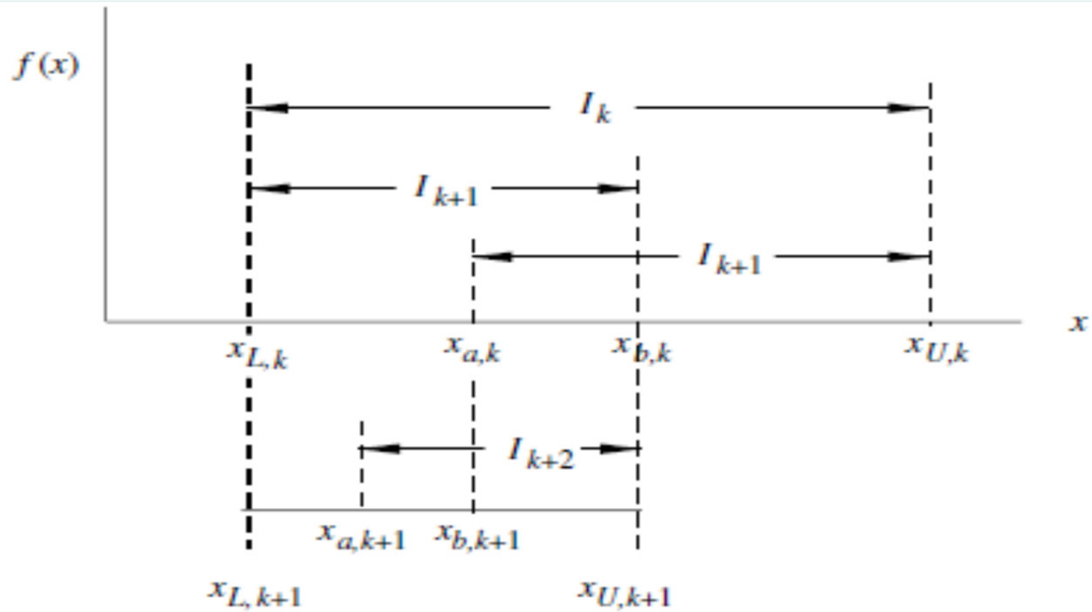
if $f_{a,k} < f_{b,k}$, then $x^*$ is in interval $[x_{L,k},\ x_{b,k}]$.

$$x_{L,k+1} = x_{L,k}$$
$$x_{U,k+1} = x_{b,k}$$
$$x_{a,k+1} = x_{U,k+1} - I_{k+2}$$
$$x_{b,k+1} = x_{a,k}$$
$$f_{b,k+1} = f_{a,k}$$

$$f_{a,k+1} = f(x_{a,k+1})$$




igure 4.5. Assignments in $k$th iteration of the Fibonacci search if $f_{a,k} > f_{b,k}$.




ure 4.6. Assignments in $k$th iteration of the fibonacci search if $f_{a,k} < f_{b,k}$

**Algorithm 4.2 Golden-section search**

**Step 1**

Input $x_{L,1}$, $x_{U,1}$, and $\varepsilon$.

**Step 2**

Assign $I_1 = x_{U,1} - x_{L,1}$, $K = 1.618034$ and compute

$$I_2 = I_1/K$$
$$x_{a,1} = x_{U,1} - I_2, \quad x_{b,1} = x_{L,1} + I_2$$
$$f_{a,1} = f(x_{a,1}), \quad f_{b,1} = f(x_{b,1})$$

Set $k = 1$.

**Step 3**

Compute

$$I_{k+2} = I_{k+1}/K$$

If $f_{a,k} \geq f_{b,k}$, then update $x_{L,k+1}$, $x_{U,k+1}$, $x_{a,k+1}$, $x_{b,k+1}$, $f_{a,k+1}$, and $f_{b,k+1}$ using Eqs. (4.7) to (4.12). Otherwise, if $f_{a,k} < f_{b,k}$, update information using Eqs. (4.13) to (4.18).

**Step 4**

If $I_k < \varepsilon$ or $x_{a,k+1} > x_{b,k+1}$, then do:

If $f_{a,k+1} > f_{b,k+1}$, compute

$$x^* = \tfrac{1}{2}(x_{b,k+1} + x_{U,k+1})$$

If $f_{a,k+1} = f_{b,k+1}$, compute

$$x^* = \tfrac{1}{2}(x_{a,k+1} + x_{b,k+1})$$

If $f_{a,k+1} < f_{b,k+1}$, compute

$$x^* = \tfrac{1}{2}(x_{L,k+1} + x_{a,k+1})$$

Compute $f^* = f(x^*)$.

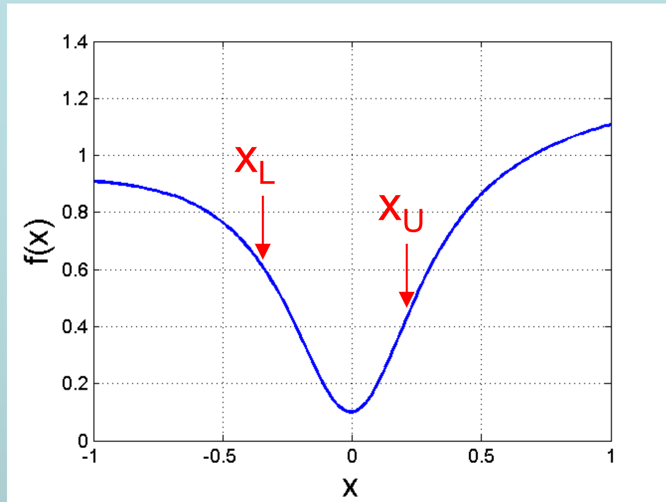
Output $x^*$ and $f^*$, and stop.

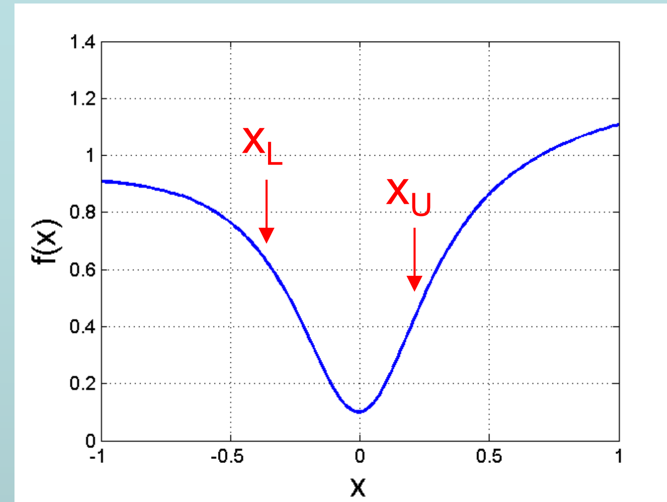
**Step 5**

Set $k = k + 1$ and repeat from Step 3.

# Optimization Methods

# Search methods



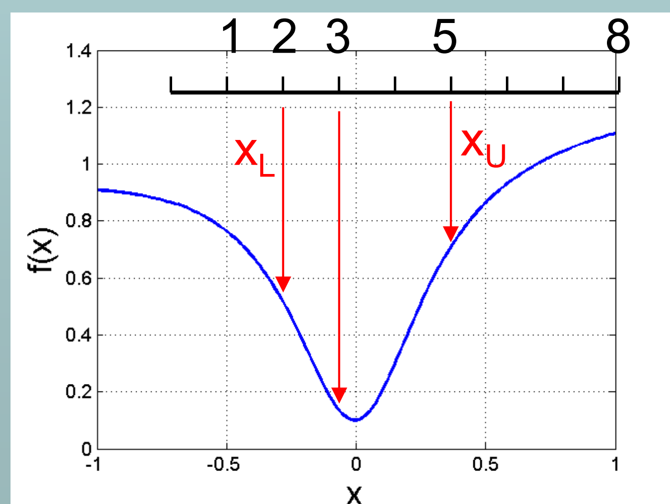

Dichotomous

Fibonacci:

1 1 2 3 5 8 …

$I_{k+5}$ $I_{k+4}$ $I_{k+3}$ $I_{k+2}$ $I_{k+1}$ $I_k$

$$I_k = I_{k+1} + I_{k+2}$$




Golden-Section Search divides intervals by $K = 1.6180$

$$\frac{I_k}{I_{k+1}} = K$$

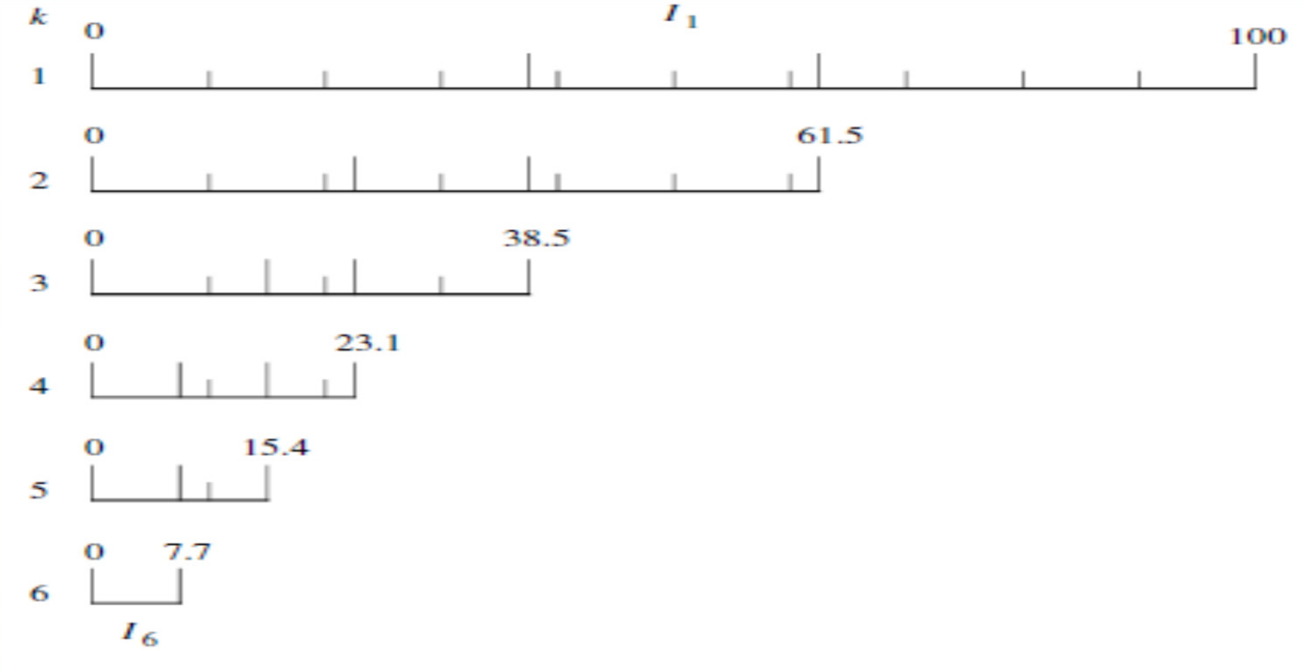



*Figure 4.4.* Fibonacci search for $n = 6$.

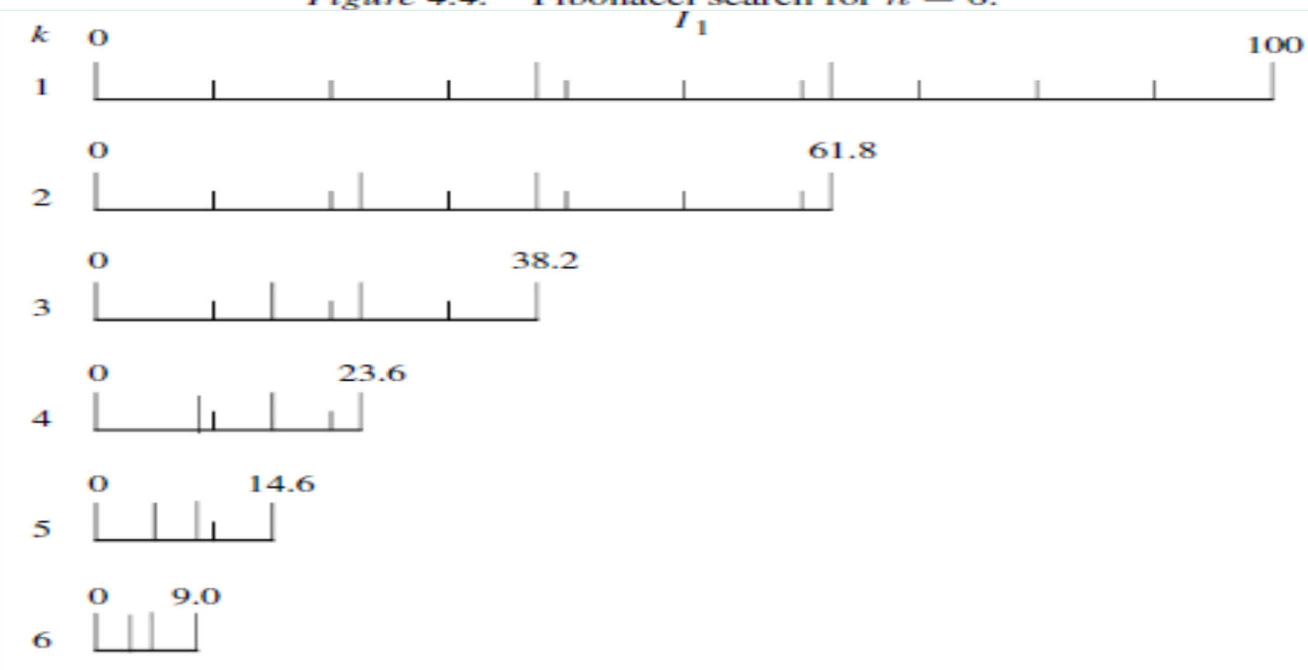



*Figure 4.8.* Golden section search.

# 1-D search

$$I_k = (\tfrac{1}{2})^k I_0$$

$$I_n = \frac{I_1}{F_n}$$

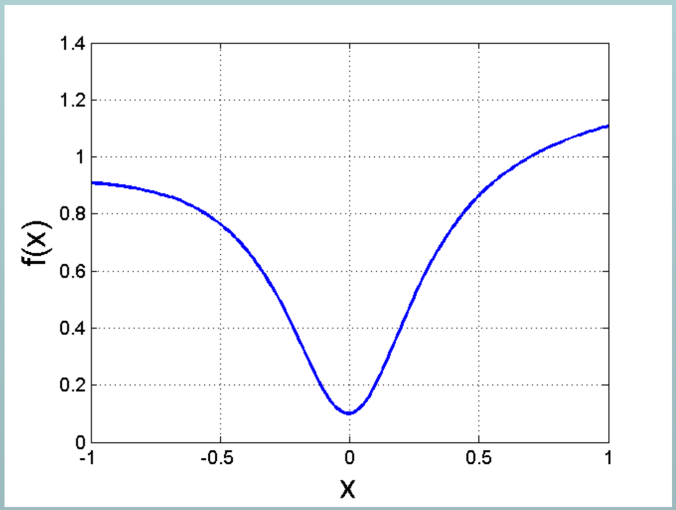
$$\frac{I_k}{I_{k+1}} = \frac{I_{k+1}}{I_{k+2}} = \frac{I_{k+2}}{I_{k+3}} = \cdots = K$$

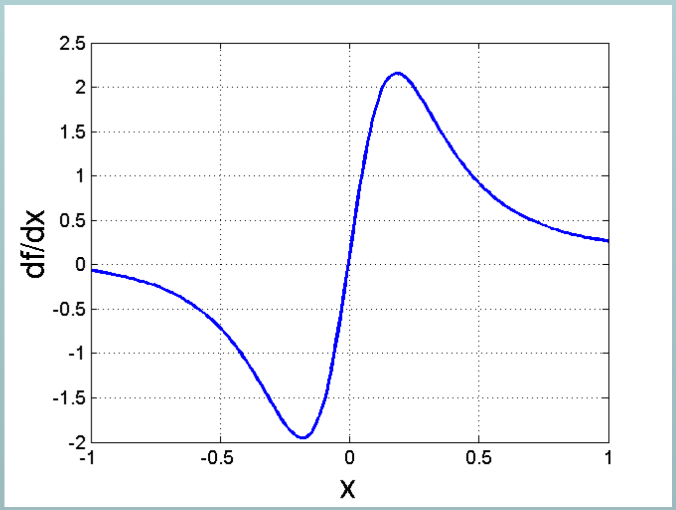
$$K = \frac{1 \pm \sqrt{5}}{2}$$

# 1D function

As an example consider the function

$$f(x) = 0.1 + 0.1x + x^2/(0.1 + x^2)$$

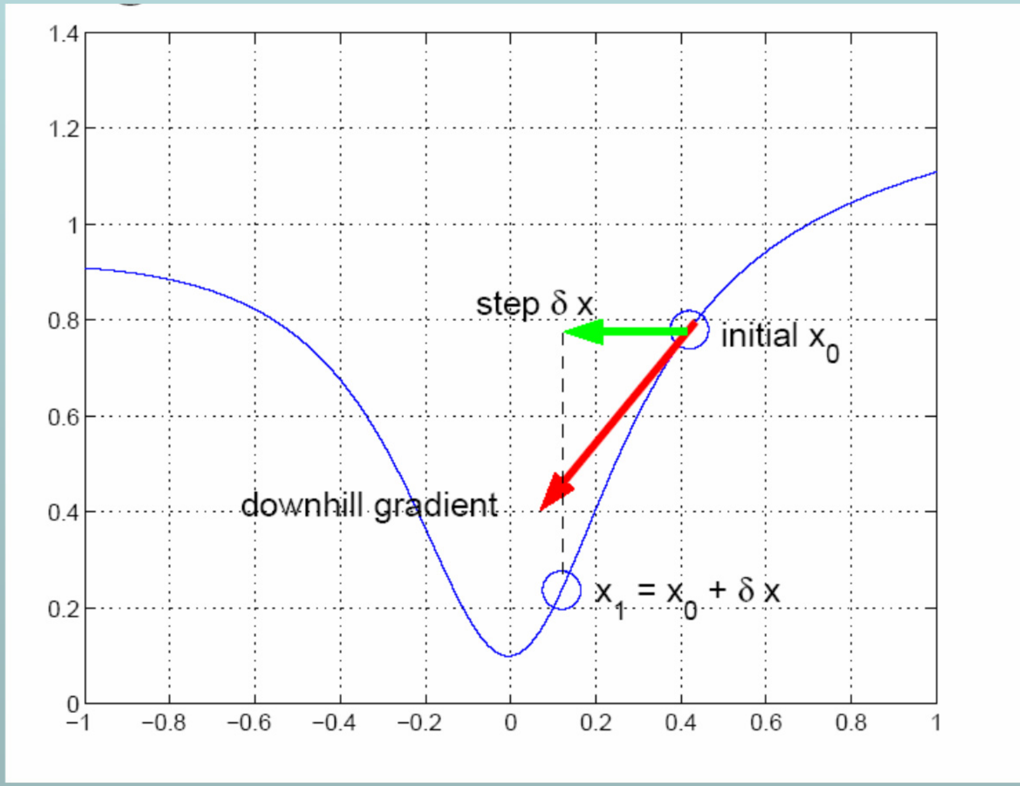



(assume we do not know the actual function expression from now on)

# Gradient descent

Given a starting location, $x_0$, examine df/dx and move in the *downhill* direction to generate a new estimate, $x_1 = x_0 + \delta x$

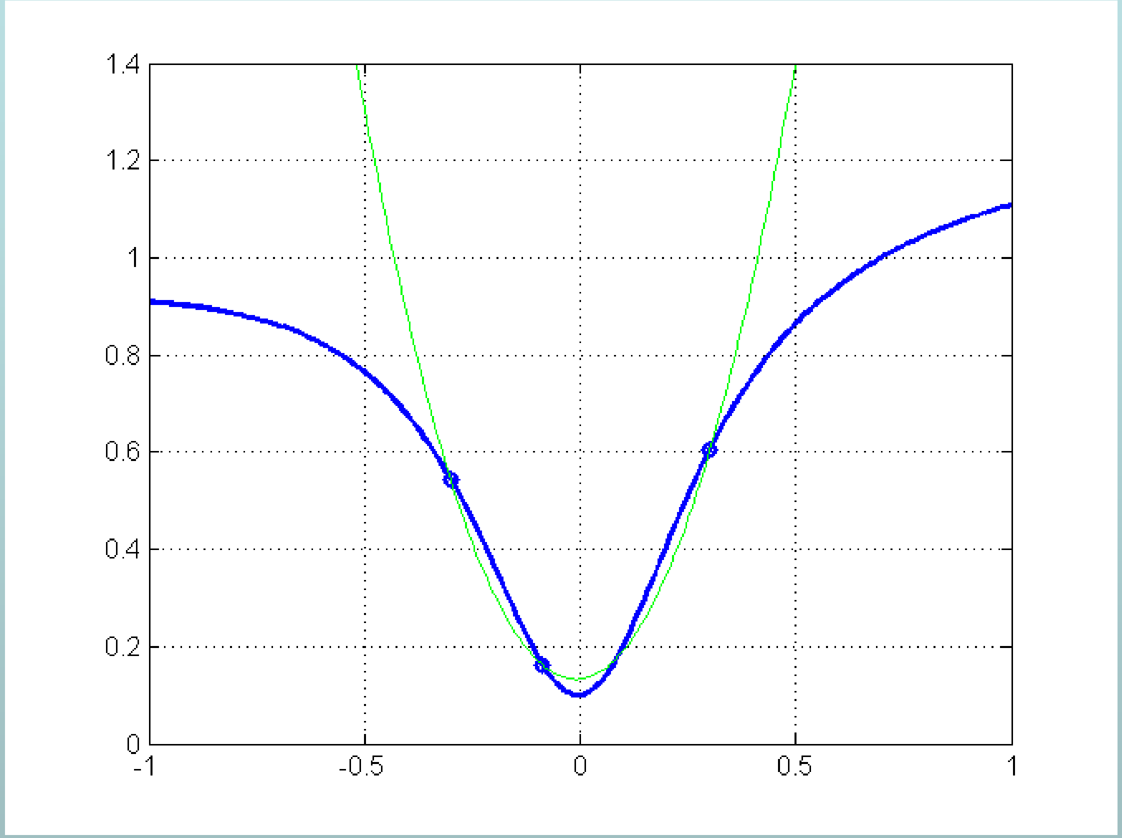



How to determine the step size $\delta x$?

# Polynomial interpolation

- Bracket the minimum.
- Fit a quadratic or cubic polynomial which interpolates $f(x)$ at some points in the interval.
- Jump to the (easily obtained) minimum of the polynomial.
- Throw away the worst point and repeat the process.

# Polynomial interpolation




- Quadratic interpolation using 3 points, 2 iterations
- Other methods to interpolate?
  - 2 points and one gradient
  - Cubic interpolation

# Newton method

Fit a quadratic approximation to $f(x)$ using both gradient and curvature information at $x$.

- Expand $f(x)$ locally using a Taylor series.

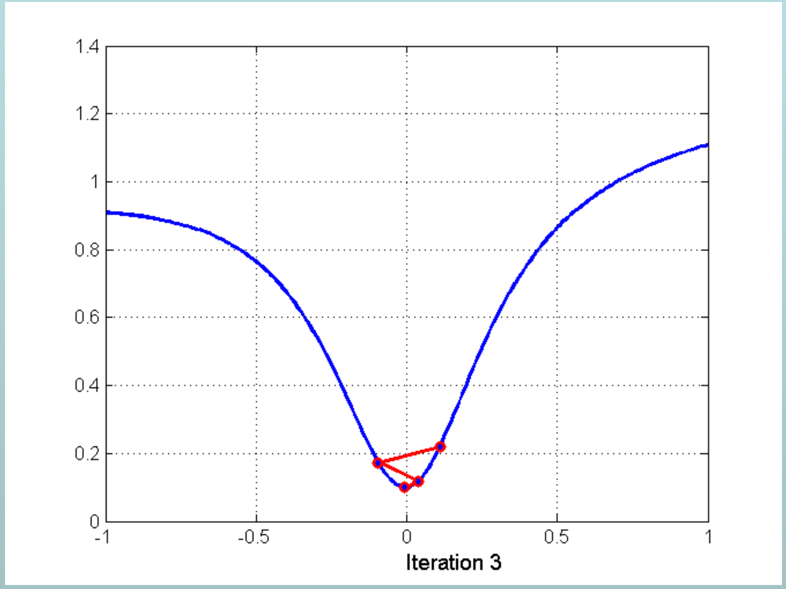
$$f(x+\delta x) = f(x) + f'(x)\delta x + \frac{1}{2} f''(x)\delta x^2 + o(\delta x^2)$$

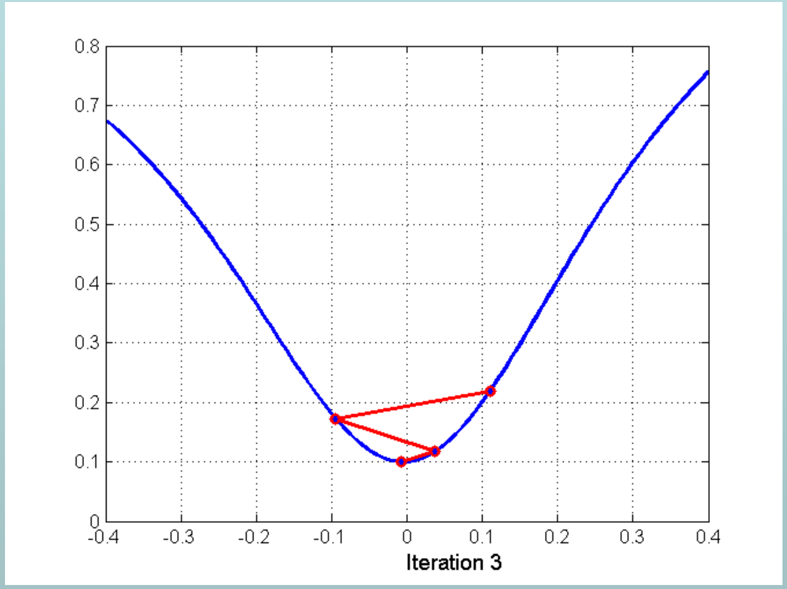
- Find the $\delta x$ which minimizes this local quadratic approximation.

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

- Update $x$. $$x_{n+1} = x_n - \delta x = x_n - \frac{f'(x)}{f''(x)}$$

# Newton method

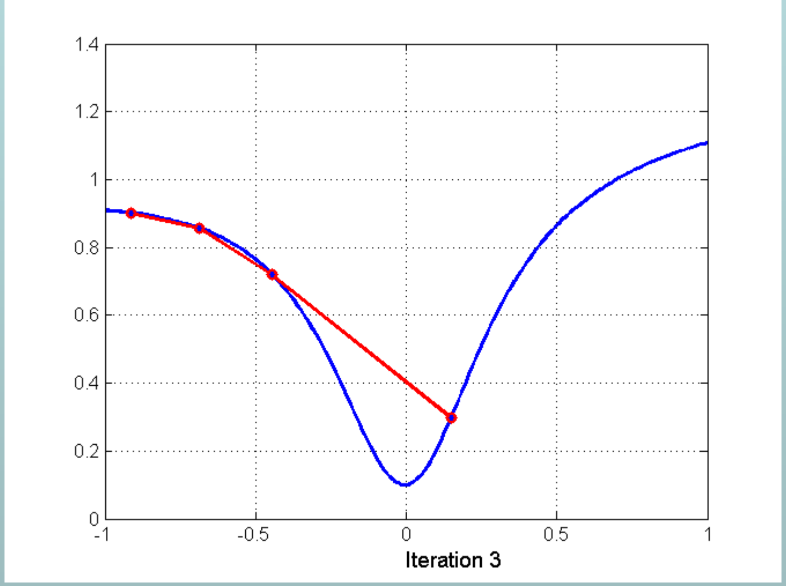


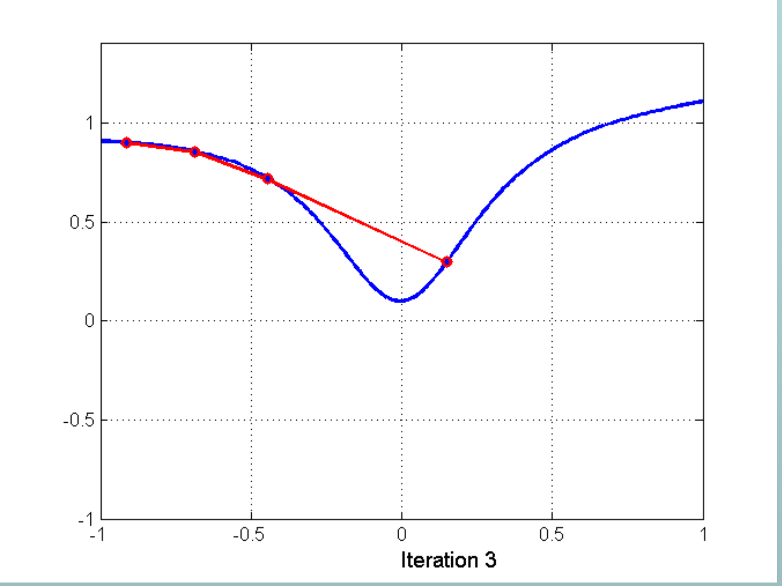




- avoids the need to bracket the root
- quadratic convergence (decimal accuracy doubles at every iteration)

# Newton method

- Global convergence of Newton's method is poor.
- Often fails if the starting point is too far from the minimum.

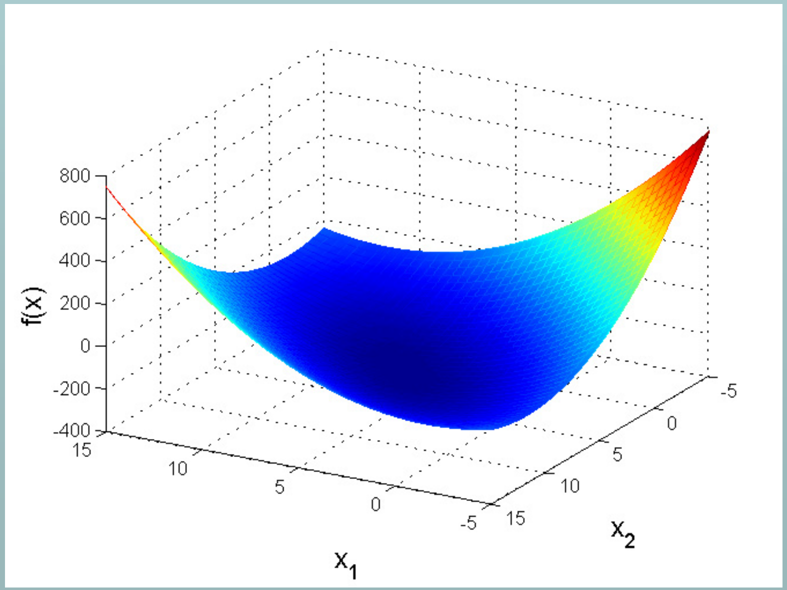


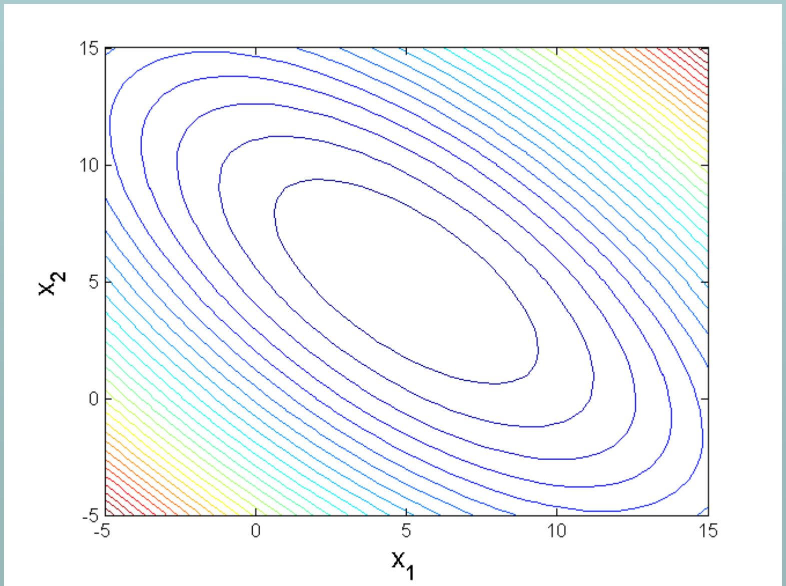




- in practice, must be used with a globalization strategy which reduces the step length until function decrease is assured

# Extension to N (multivariate) dimensions

- How big N can be?
  - problem sizes can vary from a handful of parameters to many thousands
- We will consider examples for N=2, so that cost function surfaces can be visualized.



# An Optimization Algorithm

- Start at $\mathbf{x}_0$, $k = 0$.

1. Compute a search direction $\mathbf{p}_k$
2. Compute a step length $\alpha_k$, such that $f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) < f(\mathbf{x}_k)$
3. Update $\mathbf{x}_k = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
4. Check for convergence (stopping criteria)
   e.g. $\mathrm{d}f/\mathrm{d}\mathbf{x} = \mathbf{0}$

$k = k+1$

Reduces optimization in N dimensions to a series of (1D) line minimizations

# Taylor expansion

A function may be approximated locally by its Taylor series expansion about a point $\mathbf{x}^*$

$$\boxed{f(\mathbf{x}^* + \mathbf{x}) \approx f(\mathbf{x}^*) + \nabla f^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \mathbf{H} \mathbf{x}}$$

where the gradient $\nabla f(\mathbf{x}^*)$ is the vector

$$\nabla f(\mathbf{x}^*) = \left[\frac{\partial f}{x_1} \ldots \frac{\partial f}{x_N}\right]^T$$

and the Hessian $\mathbf{H}(\mathbf{x}^*)$ is the symmetric matrix

$$\mathbf{H}(\mathbf{x}^*) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \ldots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \ldots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix}$$

# Summary of Eqns. Studies so far

$$\mathbf{g}(\mathbf{x}) = [\frac{\partial f}{\partial x_1} \quad \frac{\partial f}{\partial x_2} \quad \ldots \quad \frac{\partial f}{\partial x_n}]^T = \nabla f(\mathbf{x})$$

$$\nabla = [\frac{\partial}{\partial x_1} \quad \frac{\partial}{\partial x_2} \quad \ldots \quad \frac{\partial}{\partial x_n}]^T$$

the *Hessian*[1] of $f(\mathbf{x})$ is defined as

$$\mathbf{H}(\mathbf{x}) = \nabla \mathbf{g}^T = \nabla\{\nabla^T f(\mathbf{x})\}$$

$$\mathbf{H}(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \ldots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \ldots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \ldots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix}$$

$$f(\mathbf{x}+\boldsymbol{\delta}) = f(\mathbf{x}) + \mathbf{g}(\mathbf{x})^T\boldsymbol{\delta} + \tfrac{1}{2}\boldsymbol{\delta}^T\mathbf{H}(\mathbf{x})\boldsymbol{\delta} + o(\|\boldsymbol{\delta}\|^2)$$

$$f(\mathbf{x}+\boldsymbol{\delta}) \approx f(\mathbf{x}) + \mathbf{g}(\mathbf{x})^T\boldsymbol{\delta}$$

$$f(\mathbf{x}+\boldsymbol{\delta}) \approx f(\mathbf{x}) + \mathbf{g}(\mathbf{x})^T\boldsymbol{\delta} + \tfrac{1}{2}\boldsymbol{\delta}^T\mathbf{H}(\mathbf{x})\boldsymbol{\delta}$$

$$f(\mathbf{x}+\boldsymbol{\delta}) = f(\mathbf{x}) + \mathbf{g}(\mathbf{x})^T\boldsymbol{\delta} + \tfrac{1}{2}\boldsymbol{\delta}^T\mathbf{H}(\mathbf{x}+\alpha\boldsymbol{\delta})\boldsymbol{\delta}$$

**Theorem 2.1** *First-order necessary conditions for a minimum*

*(a) If* $f(\mathbf{x}) \in C^1$ *and* $\mathbf{x}^*$ *is a local minimizer, then*

$$\mathbf{g}(\mathbf{x}^*)^T\mathbf{d} \geq 0$$

*for every feasible direction* $\mathbf{d}$ *at* $\mathbf{x}^*$.

*(b) If* $\mathbf{x}^*$ *is located in the interior of* $\mathcal{R}$ *then*

$$\mathbf{g}(\mathbf{x}^*) = 0$$

**Theorem 2.2** ***Second-order necessary conditions for a minimum***

*(a) If $f(\mathbf{x}) \in C^2$ and $\mathbf{x}^*$ is a local minimizer, then for every feasible direction $\mathbf{d}$ at $\mathbf{x}^*$*

*(i) $\mathbf{g}(\mathbf{x}^*)^T\mathbf{d} \geq 0$*

*(ii) If $\mathbf{g}(\mathbf{x}^*)^T\mathbf{d} = 0$, then $\mathbf{d}^T\mathbf{H}(\mathbf{x}^*)\mathbf{d} \geq 0$*

*(b) If $\mathbf{x}^*$ is a local minimizer in the interior of $\mathcal{R}$, then*

*(i) $\mathbf{g}(\mathbf{x}^*) = \mathbf{0}$*

*(ii) $\mathbf{d}^T\mathbf{H}(\mathbf{x})^*\mathbf{d} \geq 0$ for all $\mathbf{d} \neq \mathbf{0}$*

**Theorem 2.4** ***Second-order sufficient conditions for a minimum*** *If $f(\mathbf{x}) \in C^2$ and $\mathbf{x}^*$ is located in the interior of $\mathcal{R}$, then the conditions*

*(a) $\mathbf{g}(\mathbf{x}^*) = \mathbf{0}$*

*(b) $\mathbf{H}(\mathbf{x}^*)$ is positive definite*

*are sufficient for $\mathbf{x}^*$ to be a strong local minimizer.*

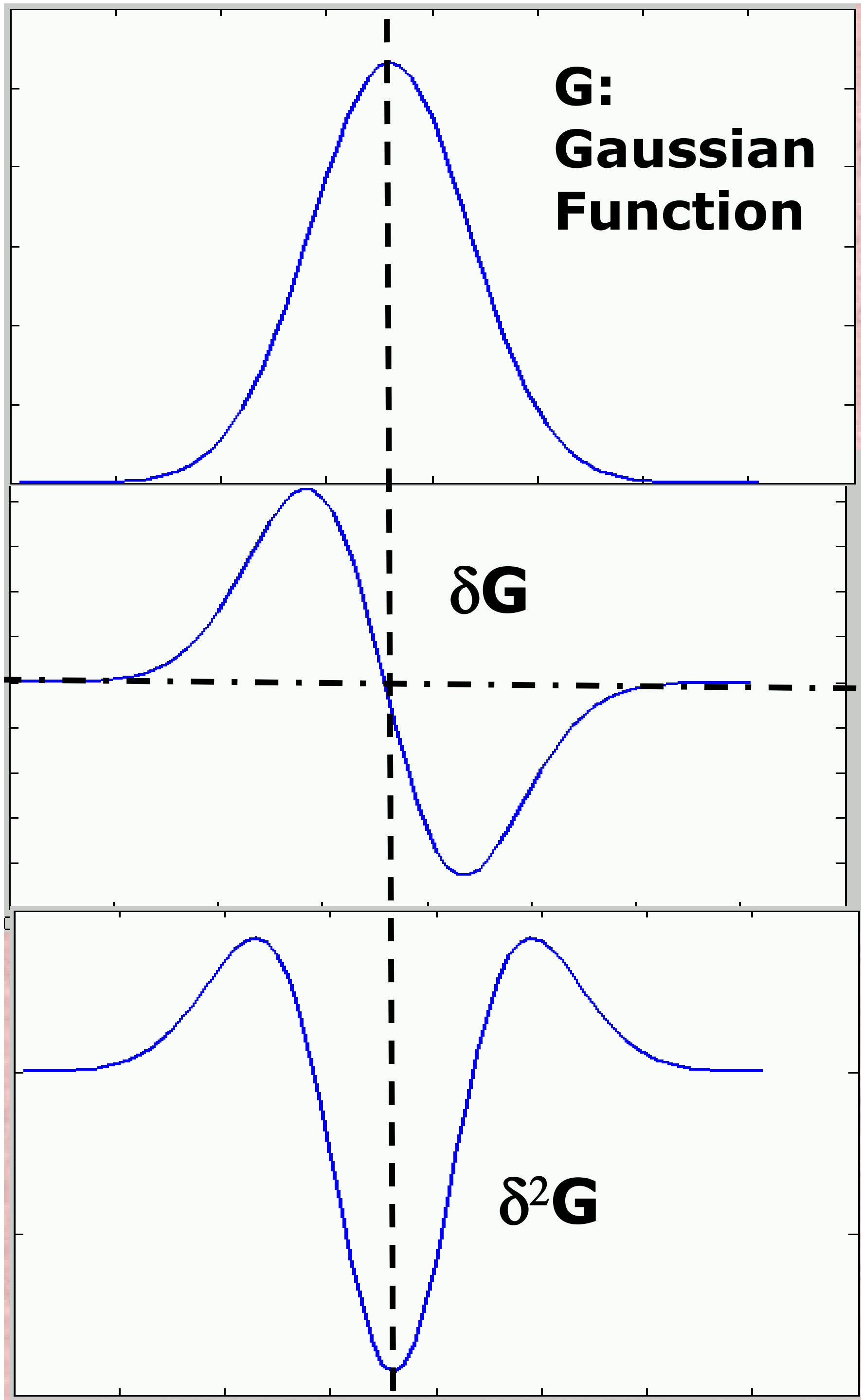
**Definition 2.6** A point $\bar{\mathbf{x}} \in \mathcal{R}$, where $\mathcal{R}$ is the feasible region, is said to be a *saddle point* if

$(a)$ $\mathbf{g}(\bar{\mathbf{x}}) = \mathbf{0}$

$(b)$ point $\bar{\mathbf{x}}$ is neither a maximizer nor a minimizer.

Stationary points can be located and classified as follows:

1. Find the points $\mathbf{x}_i$ at which $\mathbf{g}(\mathbf{x}_i) = \mathbf{0}$.
2. Obtain the Hessian $\mathbf{H}(\mathbf{x}_i)$.
3. Determine the character of $\mathbf{H}(\mathbf{x}_i)$ for each point $\mathbf{x}_i$.




$$\text{Take}: G(x) = \frac{-1}{\sqrt{2\pi\sigma}} e^{-\frac{x^2}{2\sigma^2}}$$

$$\nabla G(x) =$$

$$\frac{x}{\sqrt{2\pi\sigma_2^3}} \exp\left(-\frac{x^2}{2\sigma_2^2}\right)$$

$$\nabla^2 G(x) =$$

$$g(x^*) = \nabla G(x^*) = 0$$

$$\mathbf{g}(\mathbf{x}^*)^T \mathbf{d} \geq 0 \qquad \mathbf{g}(\mathbf{x}^*) = 0$$

$$\mathbf{d}^T \mathbf{H}(\mathbf{x}^*) \mathbf{d} \geq 0$$

# Quadratic functions

$$f(\mathbf{x}) = a + \mathbf{g}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}$$

- The vector $\mathbf{g}$ and the Hessian $\mathbf{H}$ are constant.
- Second order approximation of any function by the Taylor expansion is a quadratic function.

We will assume only quadratic functions for a while.

# Necessary conditions for a minimum

$$f(\mathbf{x}) = a + \mathbf{g}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}$$

Expand $f(\mathbf{x})$ about a stationary point $\mathbf{x}^*$ in direction $\mathbf{p}$

$$\begin{aligned} f(\mathbf{x}^* + \alpha\mathbf{p}) &= f(\mathbf{x}^*) + \mathbf{g}(\mathbf{x}^*)^T\alpha\mathbf{p} + \frac{1}{2}\alpha^2\mathbf{p}^T\mathbf{H}\mathbf{p} \\ &= f(\mathbf{x}^*) + \frac{1}{2}\alpha^2\mathbf{p}^T\mathbf{H}\mathbf{p} \end{aligned}$$

since at a stationary point $\mathbf{g}(\mathbf{x}^*) = 0$

At a stationary point the behavior is determined by $\mathbf{H}$

- $\mathbf{H}$ is a symmetric matrix, and so has orthogonal eigenvectors

$$\mathbf{H}\mathbf{u}_i = \lambda_i \mathbf{u}_i \qquad \|\mathbf{u}_i\| = 1$$

$$\begin{aligned} f(\mathbf{x}^* + \alpha \mathbf{u}_i) &= f(\mathbf{x}^*) + \frac{1}{2}\alpha^2 \mathbf{u}_i^T \mathbf{H} \mathbf{u}_i \\ &= f(\mathbf{x}^*) + \frac{1}{2}\alpha^2 \lambda_i \end{aligned}$$

- As $|\alpha|$ increases, $f(\mathbf{x}^* + \alpha \mathbf{u}_i)$ increases, decreases or is unchanging according to whether $\lambda_i$ is positive, negative or zero

# Examples of quadratic functions

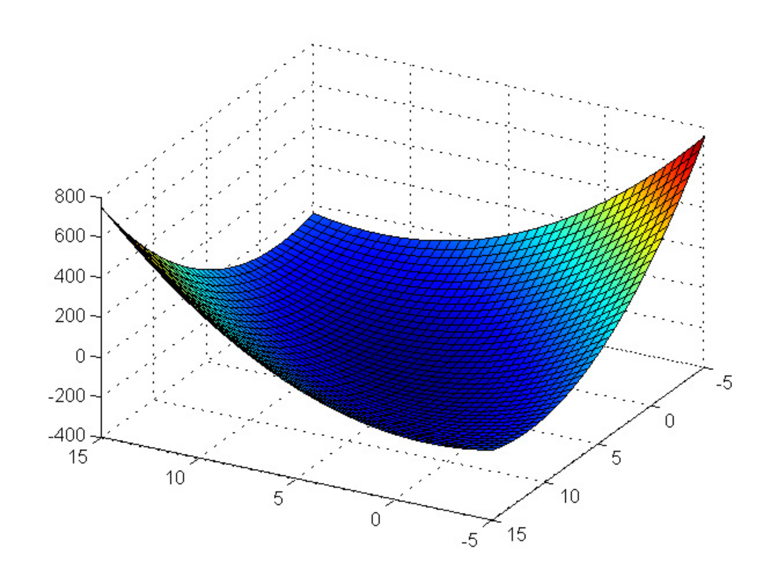
Case 1: both eigenvalues positive

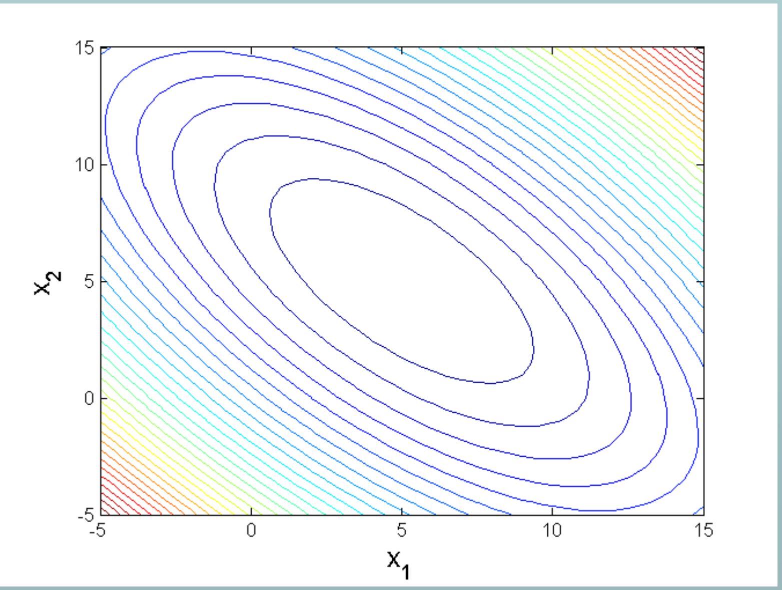
$$f(\mathbf{x}) = a + \mathbf{g}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}$$

with

$$a = 0\,, \qquad \mathbf{g} = \begin{bmatrix} -50 \\ -50 \end{bmatrix}\,, \qquad \mathbf{H} = \begin{bmatrix} 6 & 4 \\ 4 & 6 \end{bmatrix}$$

positive definite




minimum

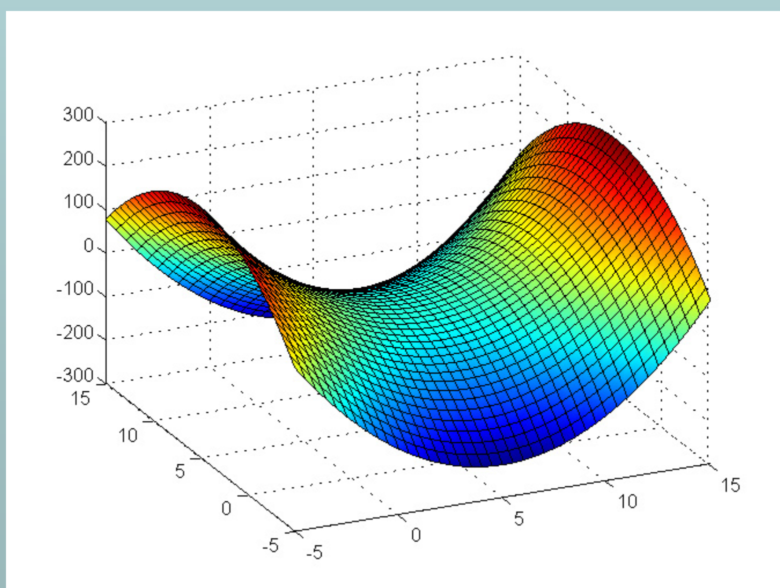
# Examples of quadratic functions

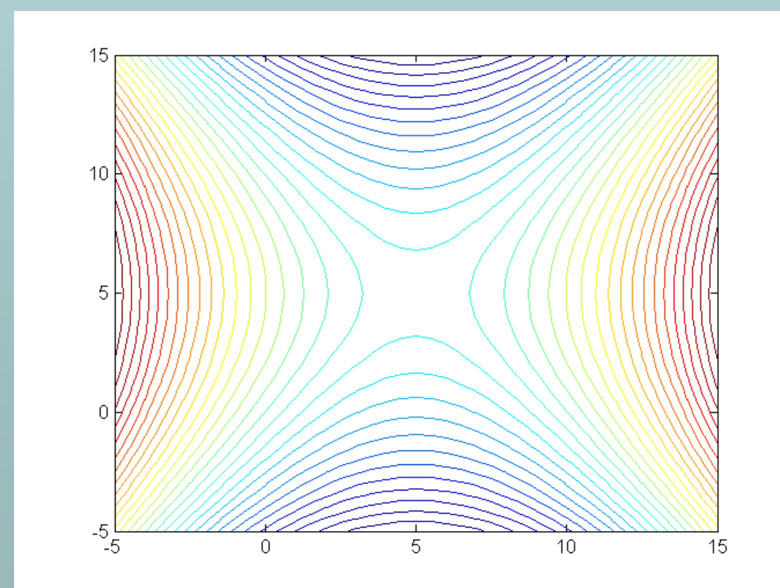
Case 2: eigenvalues have different sign

$$f(\mathbf{x}) = a + \mathbf{g}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}$$

with

$$a = 0\,, \qquad \mathbf{g} = \begin{bmatrix} -30 \\ 20 \end{bmatrix}\,, \qquad \mathbf{H} = \begin{bmatrix} 6 & 0 \\ 0 & -4 \end{bmatrix}$$ indefinite




saddle point

# Examples of quadratic functions

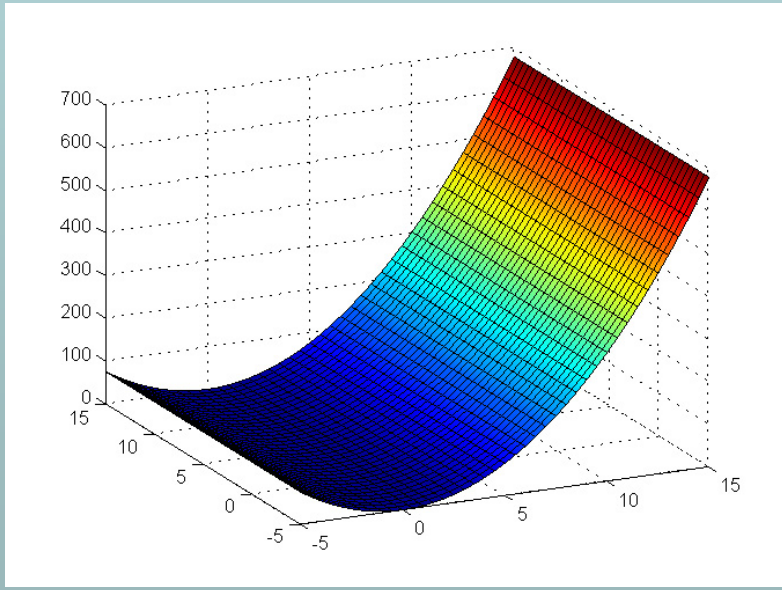
Case 3: one eigenvalues is zero

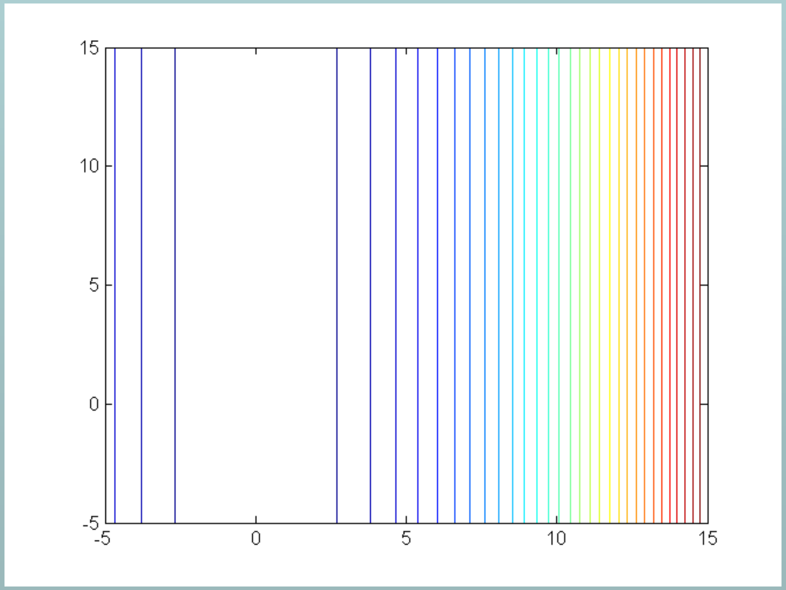
$$f(\mathbf{x}) = a + \mathbf{g}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}$$

with

$$a = 0\,, \qquad \mathbf{g} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}\,, \qquad \mathbf{H} = \begin{bmatrix} 6 & 0 \\ 0 & 0 \end{bmatrix}$$

positive semidefinite




parabolic cylinder

# Optimization for quadratic functions

Assume that $\mathbf{H}$ is positive definite

$$f(\mathbf{x}) = a + \mathbf{g}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}$$

$$\nabla f(\mathbf{x}) = \mathbf{g} + \mathbf{H}\mathbf{x}$$

There is a unique minimum at

$$\mathbf{x}^* = -\mathbf{H}^{-1}\mathbf{g}$$

If N is large, it is not feasible to perform this inversion directly.

# Steepest descent

$$F + \Delta F = f(\mathbf{x} + \delta) \approx f(\mathbf{x}) + \mathbf{g}^T \delta + \frac{1}{2} \delta^T \mathbf{H} \delta$$

- Basic principle is to minimize the N-dimensional function by a series of 1D line-minimizations:

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

- The steepest descent method chooses $\mathbf{p}_k$ to be parallel to the gradient

$$\mathbf{p}_k = -\nabla f(\mathbf{x}_k)$$

- Step-size $\alpha_k$ is chosen to minimize $f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)$.

  For quadratic forms there is a closed form solution:

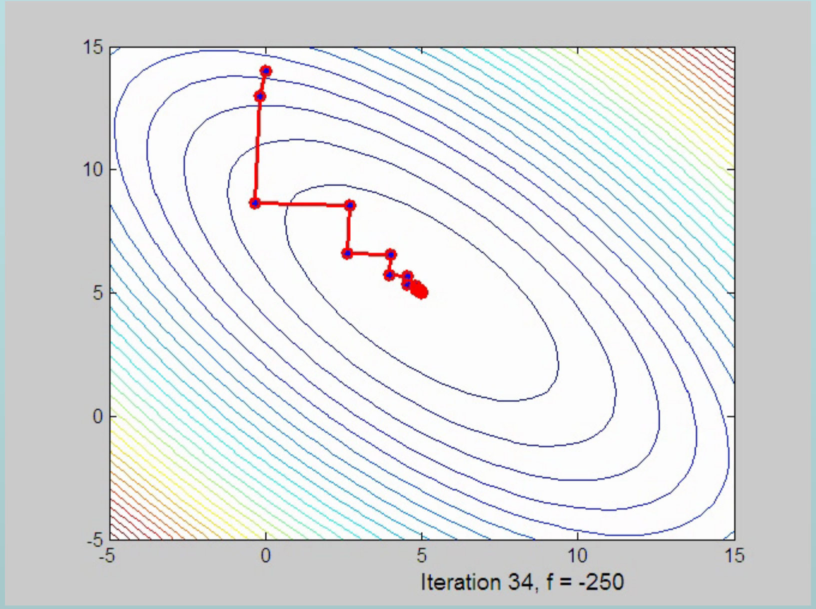
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{g}_k^T \mathbf{H}_k \mathbf{g}_k} \mathbf{g}_k$$

$$\alpha_k = \frac{\mathbf{p}_k^T \mathbf{p}_k}{\mathbf{p}_k^T \mathbf{H} \mathbf{p}_k}$$

Prove it!

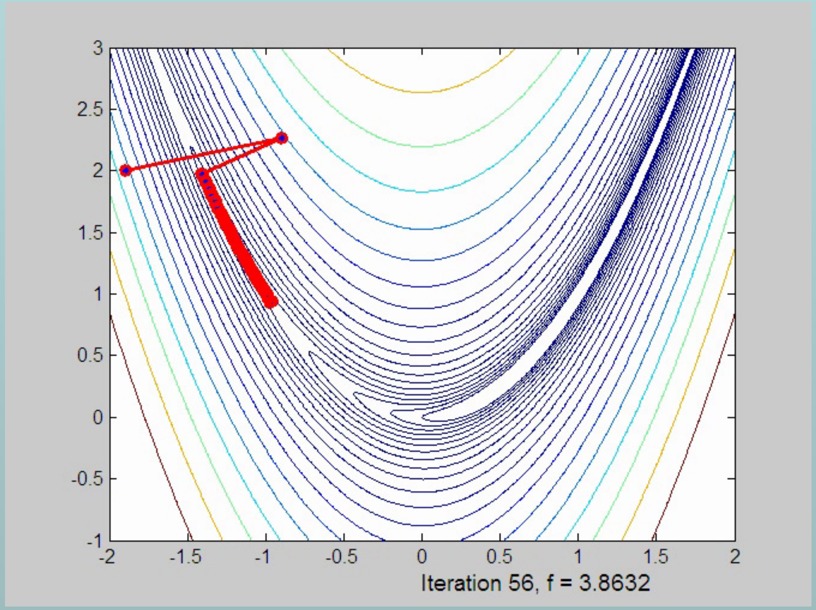
# Steepest descent

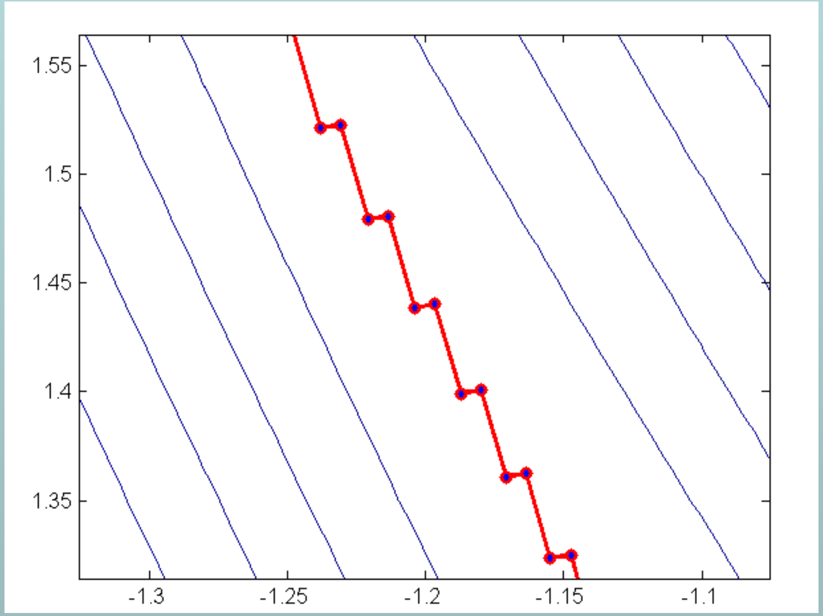



- The gradient is everywhere perpendicular to the contour lines.
- After each line minimization the new gradient is always *orthogonal* to the previous step direction (true of any line minimization).
- Consequently, the iterates tend to zig-zag down the valley in a very inefficient manner

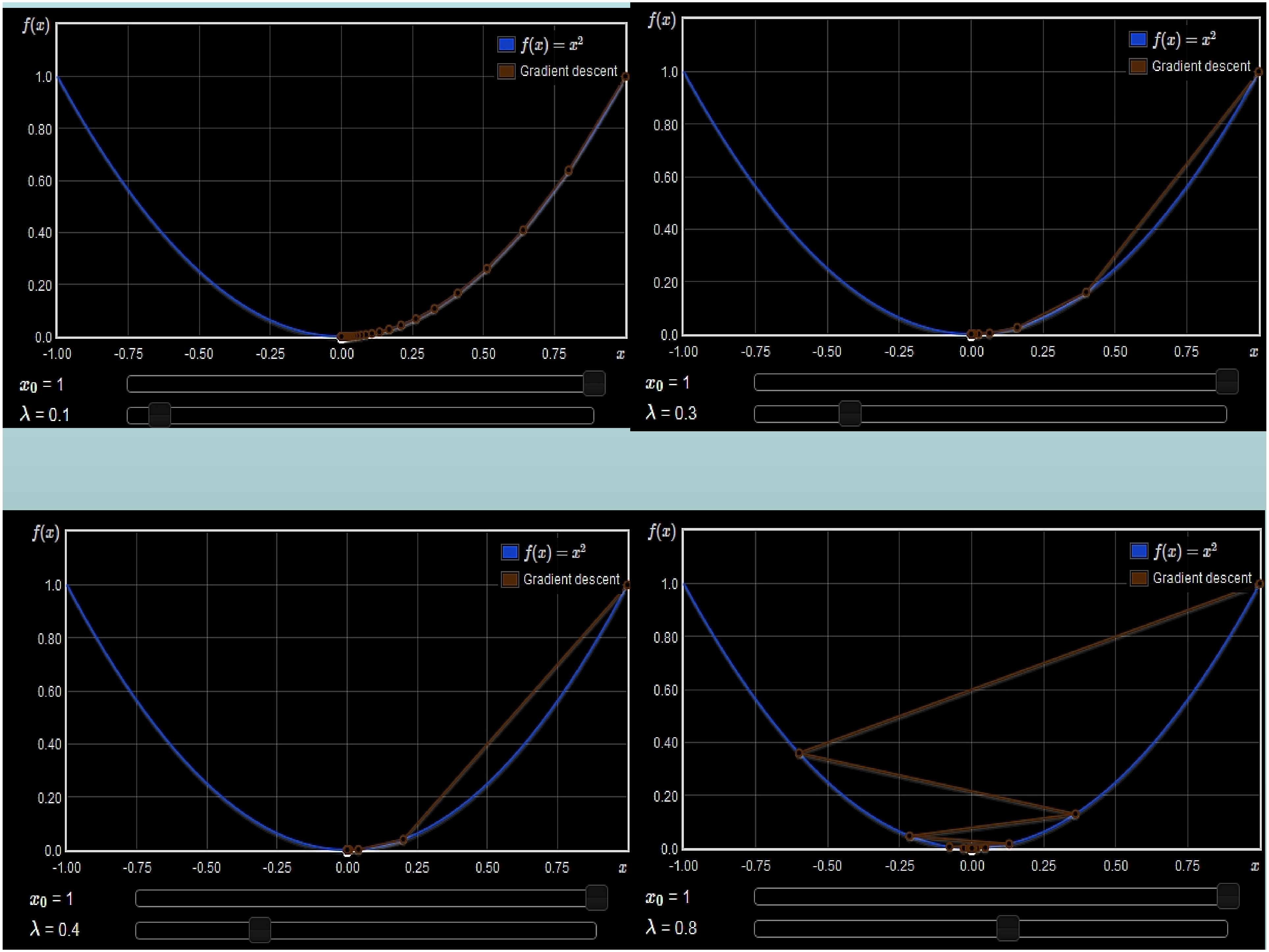
# Steepest descent

- The 1D line minimization must be performed using one of the earlier methods (usually cubic polynomial interpolation)

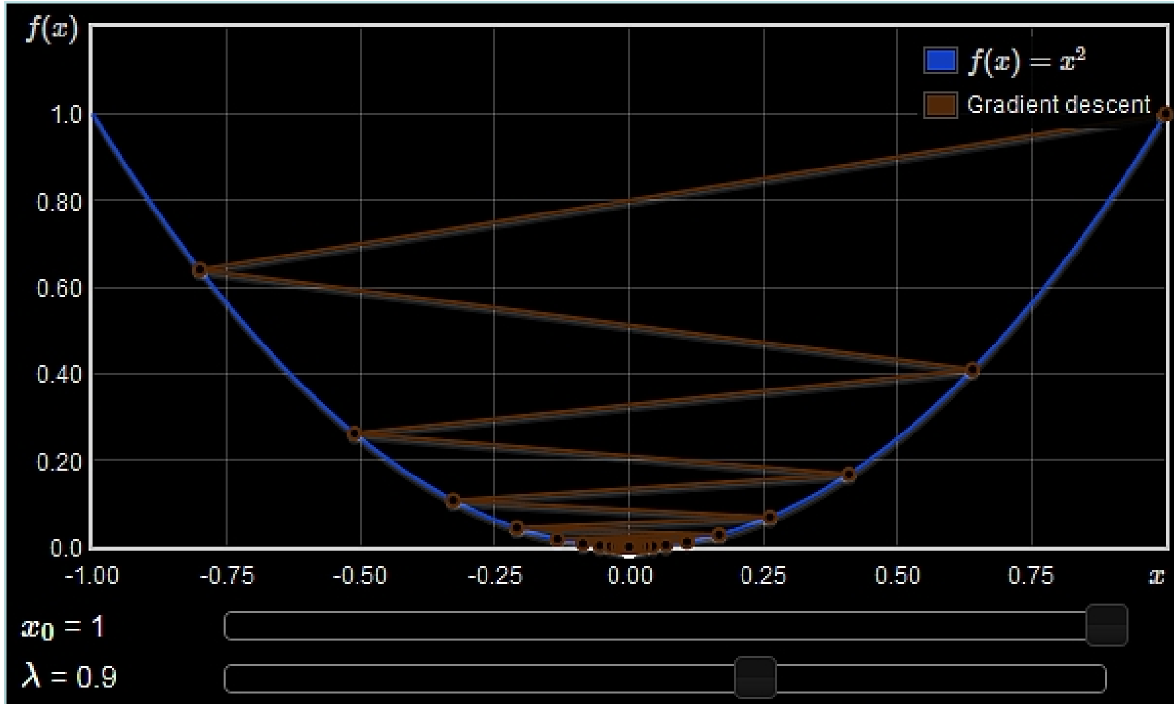


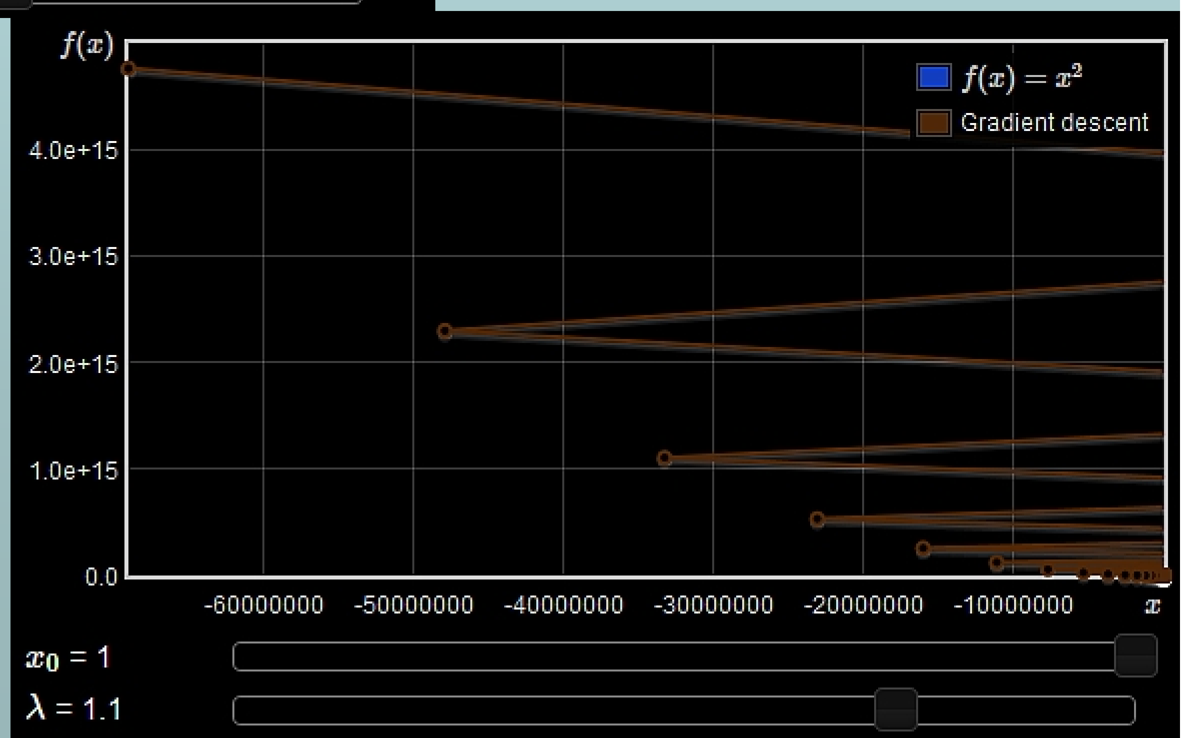

- The zig-zag behaviour is clear in the zoomed view
- The algorithm crawls down the valley



f(x)
f(x) = x^2
Gradient descent
1.0
0.80
0.60
0.40
0.20
0.0
-1.00
-0.75
-0.50
-0.25
0.00
0.25
0.50
0.75
x
x0 = 1
λ = 0.1
λ = 0.3
λ = 0.4
λ = 0.8

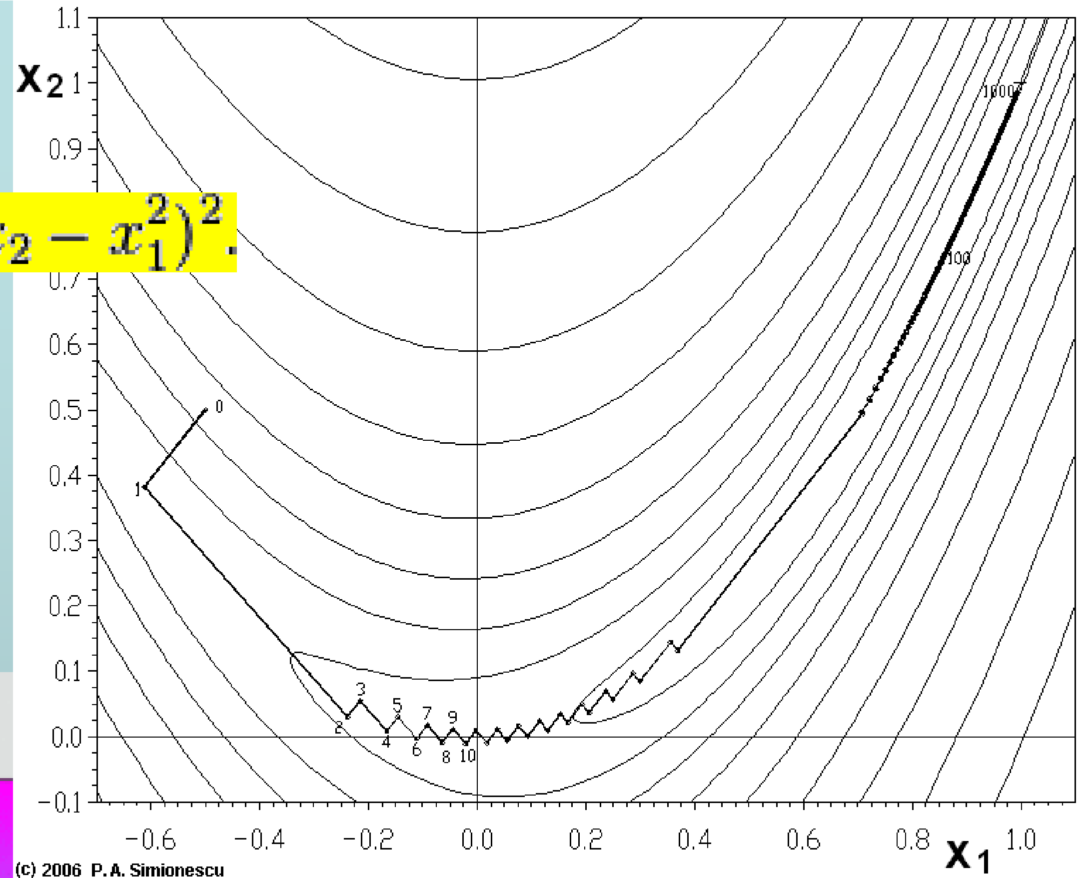


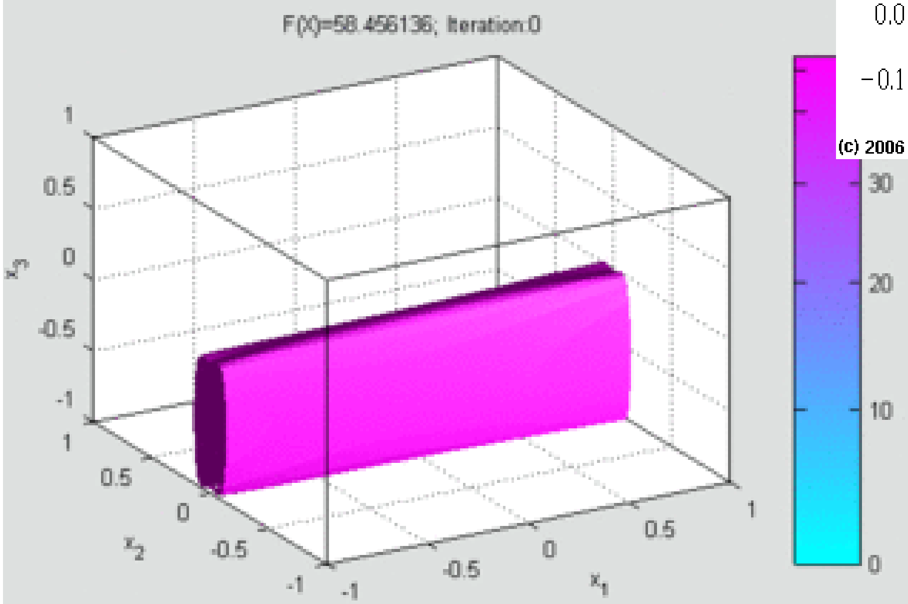
f(x)
f(x) = x²
Gradient descent
1.0
0.80
0.60
0.40
0.20
0.0
-1.00
-0.75
-0.50
-0.25
0.00
0.25
0.50
0.75
x
x₀ = 1
λ = 0.9

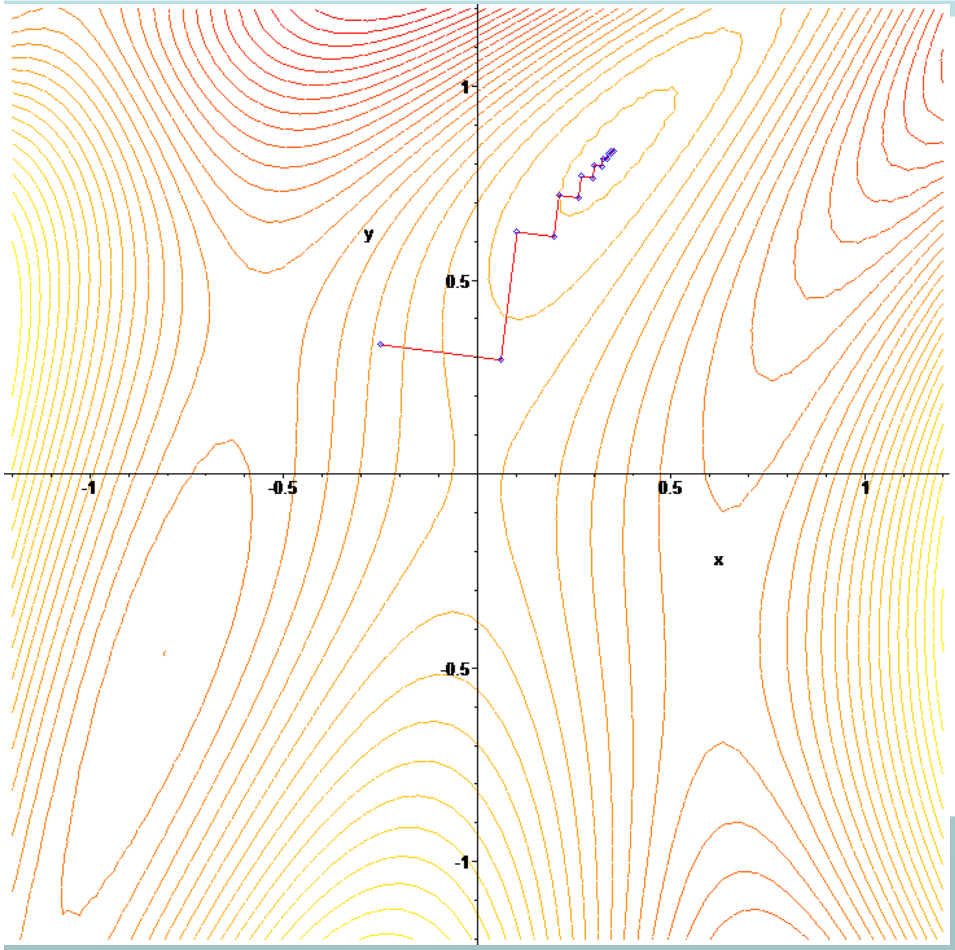
f(x)
f(x) = x²
Gradient descent
4.0e+15
3.0e+15
2.0e+15
1.0e+15
0.0
-60000000
-50000000
-40000000
-30000000
-20000000
-10000000
x
x₀ = 1
λ = 1.1

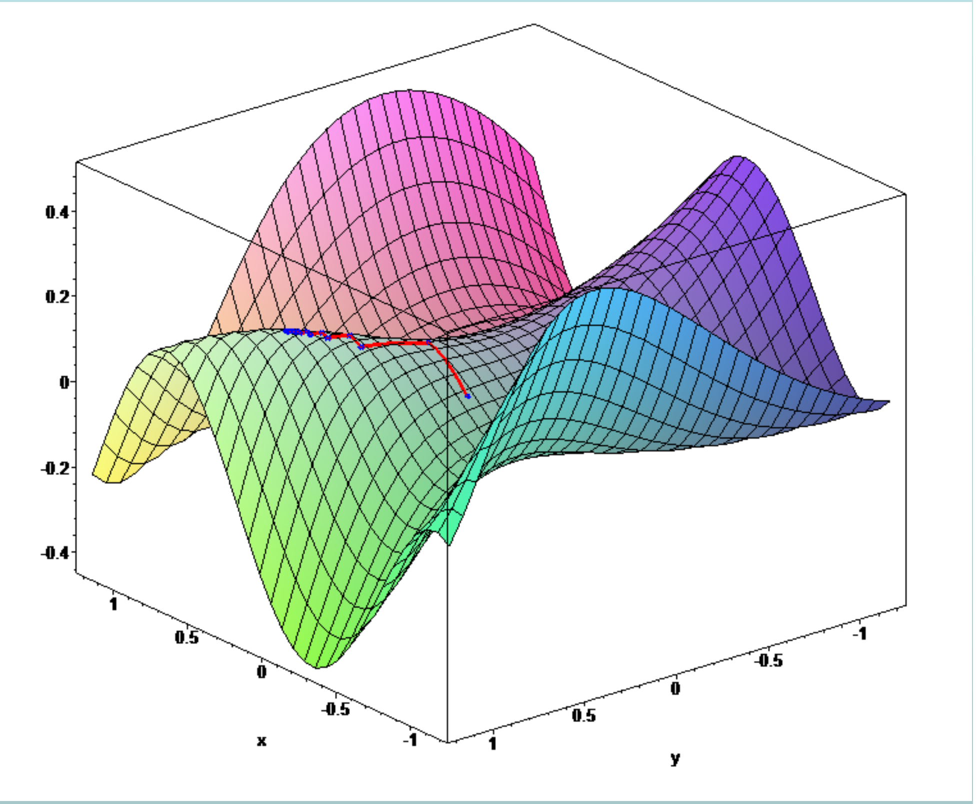
$$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2.$$




$$= \frac{1}{2}G^T(\mathbf{x})G(\mathbf{x}) = \frac{1}{2}\left(\left(3x_1 - \cos(x_2x_3) - \frac{3}{2}\right)^2 + \left(4x_1^2 - 625x_2^2 + 2x_2 - 1\right)^2 + \left(\exp(-x_1x_2) + 20x_3 + \frac{1}{3}(10\pi - 3)\right)^2\right)$$




$$F(x,y) = \sin\left(\frac{1}{2}x^2 - \frac{1}{4}y^2 + 3\right)\cos(2x + 1 - e^y)$$

# Newton method

Expand $f(\mathbf{x})$ by its Taylor series about the point $\mathbf{x}_k$

$$f(\mathbf{x}_k + \delta\mathbf{x}) \approx f(\mathbf{x}_k) + \mathbf{g}_k^T \delta\mathbf{x} + \frac{1}{2}\delta\mathbf{x}^T \mathbf{H}_k \delta\mathbf{x}$$

where the gradient is the vector

$$\mathbf{g}_k = \nabla f(\mathbf{x}_k) = \left[\frac{\partial f}{x_1} \dots \frac{\partial f}{x_N}\right]^T$$

and the Hessian is the symmetric matrix

$$\mathbf{H}_k = \mathbf{H}(\mathbf{x}_k) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix}$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \boldsymbol{\delta}_k = \mathbf{x}_k + \alpha_k \mathbf{d}_k$$

$$\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$$

# Newton method

For a minimum we require that $\nabla f(\mathbf{x}) = \mathbf{0}$, and so

$$\nabla f(\mathbf{x}) = \mathbf{g}_k + \mathbf{H}_k \delta\mathbf{x} = \mathbf{0}$$

with solution $\delta\mathbf{x} = -\mathbf{H}_k^{-1}\mathbf{g}_k$. This gives the iterative update

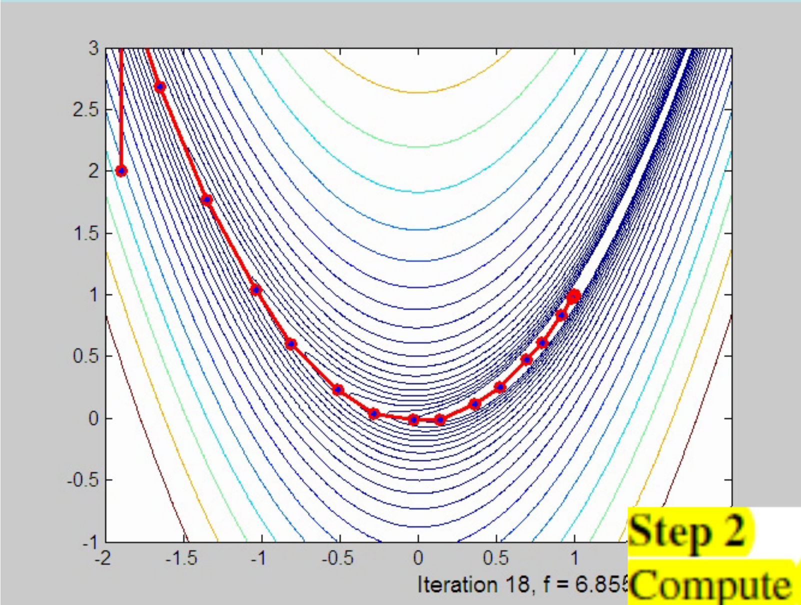
$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}_k^{-1}\mathbf{g}_k$$

- If $f(\mathbf{x})$ is quadratic, then the solution is found in one step.
- The method has quadratic convergence (as in the 1D case).
- The solution $\delta\mathbf{x} = -\mathbf{H}_k^{-1}\mathbf{g}_k$ is guaranteed to be a downhill direction.
- Rather than jump straight to the minimum, it is better to perform a line minimization which ensures global convergence

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{H}_k^{-1}\mathbf{g}_k$$

- If $\mathbf{H}=\mathbf{I}$ then this reduces to steepest descent.

# Newton method - example




**Step 2**

Compute $\mathbf{g}_k$ and $\mathbf{H}_k$.

If $\mathbf{H}_k$ is not positive definite, force it to become positive definite.

**Step 3**

Compute $\mathbf{H}_k^{-1}$ and $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$.

**Step 4**

Find $\alpha_k$, the value of $\alpha$ that minimizes $f(\mathbf{x}_k + \alpha\mathbf{d}_k)$, using a line search.

**Step 5**

Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{d}_k$.

Compute $f_{k+1} = f(\mathbf{x}_{k+1})$.

- The algorithm converges in only 18 iterations compared to the 98 for conjugate gradients.
- However, the method requires computing the Hessian matrix at each iteration – this is not always feasible

# Gauss - Newton method

$$F = \sum_{p=1}^{m} f_p(\mathbf{x})^2 = \mathbf{f}^T\mathbf{f}$$

$$\mathbf{f} = [f_1(\mathbf{x})\ f_2(\mathbf{x})\ \cdots\ f_m(\mathbf{x})]^T$$

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

$$\mathbf{g}_F = 2\mathbf{J}^T\mathbf{f}$$

$$\mathbf{H}_F \approx 2\mathbf{J}^T\mathbf{J}$$

**Step 2**
Compute $f_{pk} = f_p(\mathbf{x}_k)$ for $p = 1, 2, \ldots, m$ and $F_k$.
**Step 3**
Compute $\mathbf{J}_k$, $\mathbf{g}_k = 2\mathbf{J}_k^T\mathbf{f}_k$, and $\mathbf{H}_k = 2\mathbf{J}_k^T\mathbf{J}_k$.
**Step 4**
Compute $\mathbf{L}_k$ and $\hat{\mathbf{D}}_k$ using Algorithm 5.4.
Compute $\mathbf{y}_k = -\mathbf{L}_k\mathbf{g}_k$ and $\mathbf{d}_k = \mathbf{L}_k^T\hat{\mathbf{D}}_k^{-1}\mathbf{y}_k$.
**Step 5**
Find $\alpha_k$, the value of $\alpha$ that minimizes $F(\mathbf{x}_k + \alpha\mathbf{d}_k)$.
**Step 6**
Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{d}_k$.
Compute $f_{p(k+1)}$ for $p = 1, 2, \ldots, m$ and $F_{k+1}$.

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{x}_k - \alpha_k(2\mathbf{J}^T\mathbf{J})^{-1}(2\mathbf{J}^T\mathbf{f}) \\ &= \mathbf{x}_k - \alpha_k(\mathbf{J}^T\mathbf{J})^{-1}(\mathbf{J}^T\mathbf{f}) \end{aligned}$$

| Method | Alpha ($\alpha$) Calculation | Intermediate Updates | Convergence Condition |
|---|---|---|---|
| **Steepest-Descent Method (Method 1)** | Find $\alpha_k$, the value of $\alpha$ that minimizes $f(x_k + \alpha d_k)$, using line search | $x_{k+1} = x_k + \alpha_k d_k$<br>$d_k = -g_k$<br>$f_{k+1} = f(x_{k+1})$ | If $\lvert\lvert \alpha_k d_k \rvert\rvert < \in$, then $x^* = x_{k+1}$ , $f(x^*) = f_{k+1}$<br>Else $k = k + 1$ |
| **Steepest-Descent Method (Method 2)** | Without Using Line Search<br>$\alpha_k \approx \dfrac{g_k^T g_k}{g_k^T H_k g_k}$ | $x_{k+1} = x_k + \alpha_k d_k$<br>$d_k = -g_k$<br>$f_{k+1} = f(x_{k+1})$ | --" "-- |
| **Newton Method** | Find $\alpha_k$, the value of $\alpha$ that minimizes $f(x_k + \alpha d_k)$, using line search | $x_{k+1} = x_k + \alpha_k d_k$<br>$d_k = -H_k^{-1} g_k$<br>$f_{k+1} = f(x_{k+1})$ | --" "-- |
| **Gauss-Newton Method** | Find $\alpha_k$, the value of $\alpha$ that minimizes F$(x_k + \alpha d_k)$, using line search<br>$F = \sum_{p=1}^{m} f_p(x)^2 = f^T f$<br>$f = [f_1(x)\ f_2(x) \ldots f_m(x)]^T$ | $x_{k+1} = x_k + \alpha_k d_k$<br>$d_k = -H_k^{-1} g_k$<br>$g_F = 2J^T f$<br>$H \approx 2J^T J = \boldsymbol{L}^{-1}\boldsymbol{D}(\boldsymbol{L}^{\boldsymbol{T}})^{-1}$<br>$x_{k+1} = x_k - \alpha_k (J^T J)^{-1}(J^T f)$<br>$x_{k+1} = x_k - \alpha_k \boldsymbol{L}^{\boldsymbol{T}}\boldsymbol{D}\boldsymbol{L} g_k$<br>$f_{p(k+1)} = f_p(x_k)$<br>$F_{(k+1)} = F(x_k)$ | If $\lvert F_{k+1} - F_k \rvert < \in$ then $x^* = x_{k+1}$ , F$(x^*) = F_{k+1}$<br>Else $k = k + 1$ |

# Conjugate gradient

- Each $\mathbf{p}_k$ is chosen to be conjugate to all previous search directions with respect to the Hessian $\mathbf{H}$:

$$\mathbf{p}_i^T \mathbf{H} \mathbf{p}_j = 0 \,, \qquad i \neq j$$

- The resulting search directions are mutually linearly independent.

Prove it!

- *Remarkably*, $\mathbf{p}_k$ can be chosen using only knowledge of $\mathbf{p}_{k-1}$, $\nabla f(\mathbf{x}_{k-1})$, and $\nabla f(\mathbf{x}_k)$

$$\mathbf{p}_k = \nabla f_k + \left( \frac{\nabla f_k^\top \nabla f_k}{\nabla f_{k-1}^\top \nabla f_{k-1}} \right) \mathbf{p}_{k-1}$$

# Conjugate-gradient algorithm

**Step 3**

Input $\mathbf{H}_k$, i.e., the Hessian at $\mathbf{x}_k$.

Compute

$$\alpha_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{d}_k^T \mathbf{H}_k \mathbf{d}_k}$$

Set $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$ and calculate $f_{k+1} = f(\mathbf{x}_{k+1})$.

**Step 4**

If $\|\alpha_k \mathbf{d}_k\| < \varepsilon$, output $\mathbf{x}^* = \mathbf{x}_{k+1}$ and $f(\mathbf{x}^*) = f_{k+1}$, and stop.

**Step 5**

Compute $\mathbf{g}_{k+1}$.

Compute

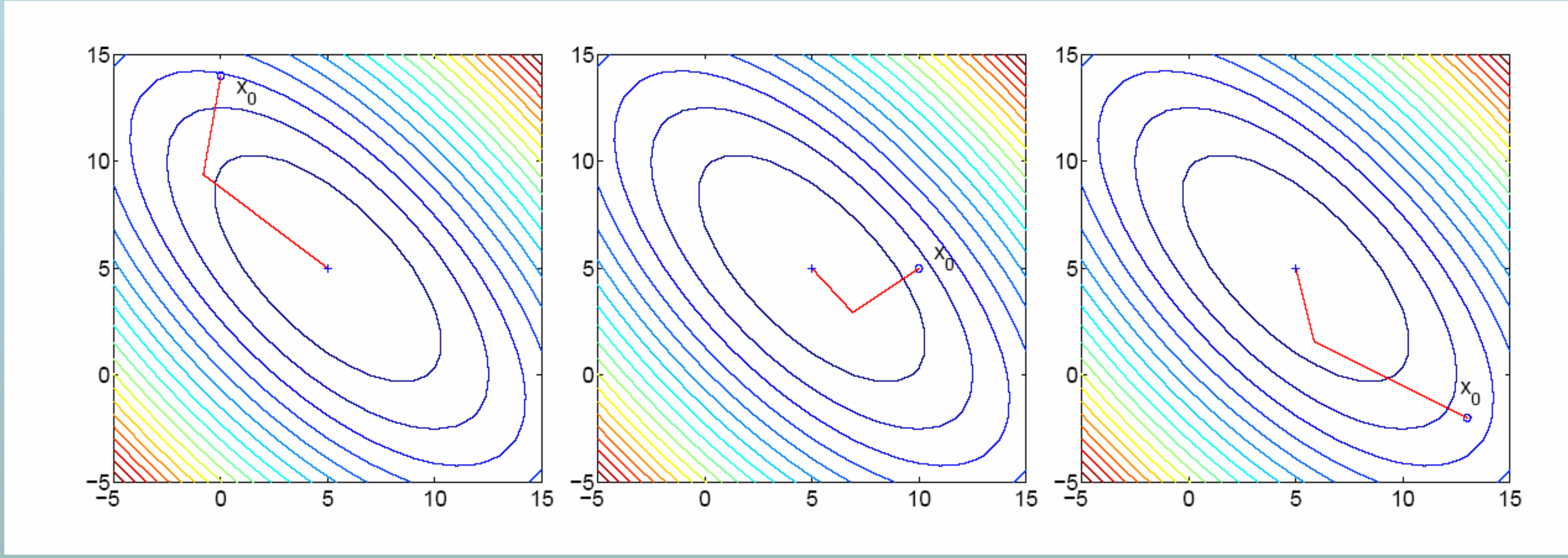
$$\beta_k = \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k}$$

Generate new direction

$$\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{d}_k$$

# Conjugate gradient

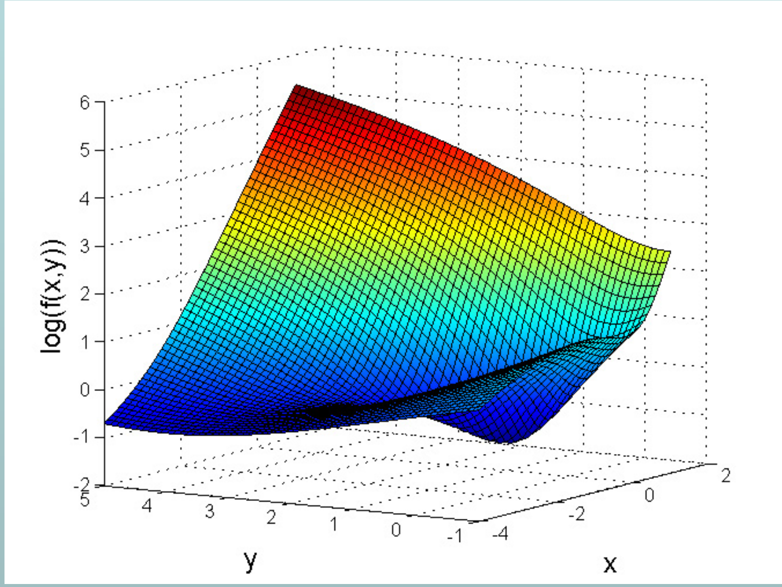
- An N-dimensional quadratic form can be minimized in at most N conjugate descent steps.

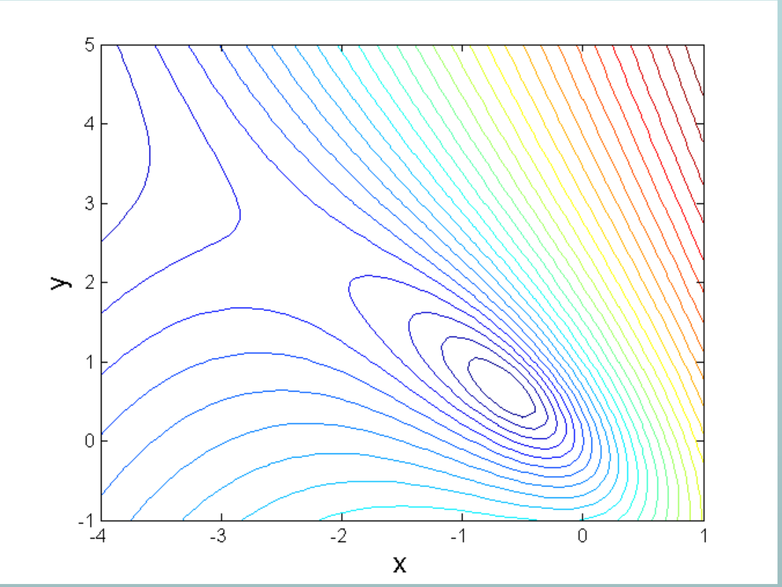



- 3 different starting points.
- Minimum is reached in exactly 2 steps.

# Optimization for General functions

$$f(x, y) = \exp(x)(4x^2 + 2y^2 + 4xy + 2x + 1)$$

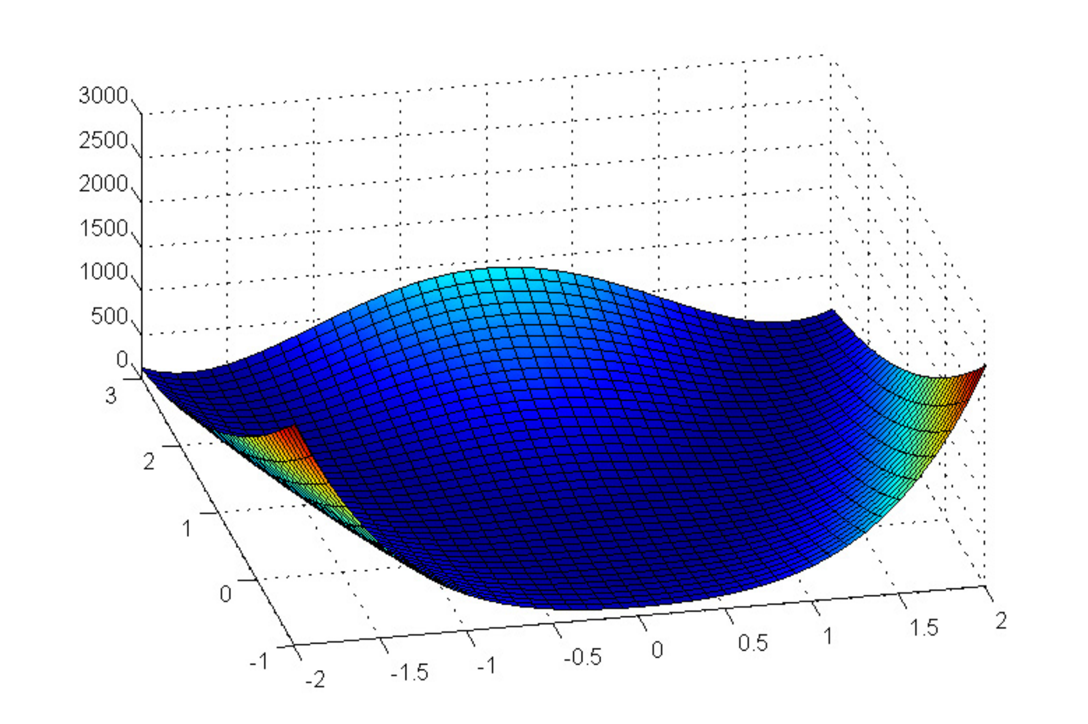


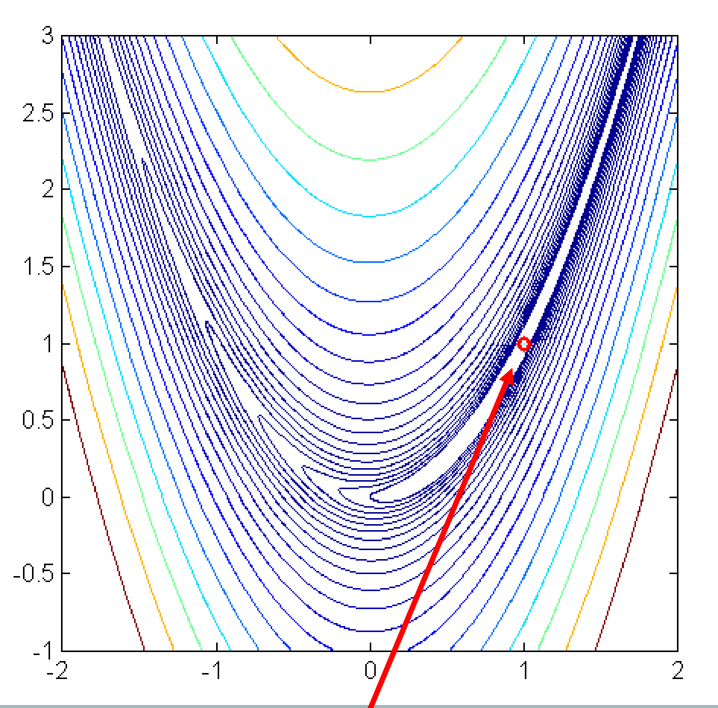




Apply methods developed using quadratic Taylor series expansion

# Rosenbrock's function

$$f(x, y) = 100(y - x^2)^2 + (1 - x)^2$$

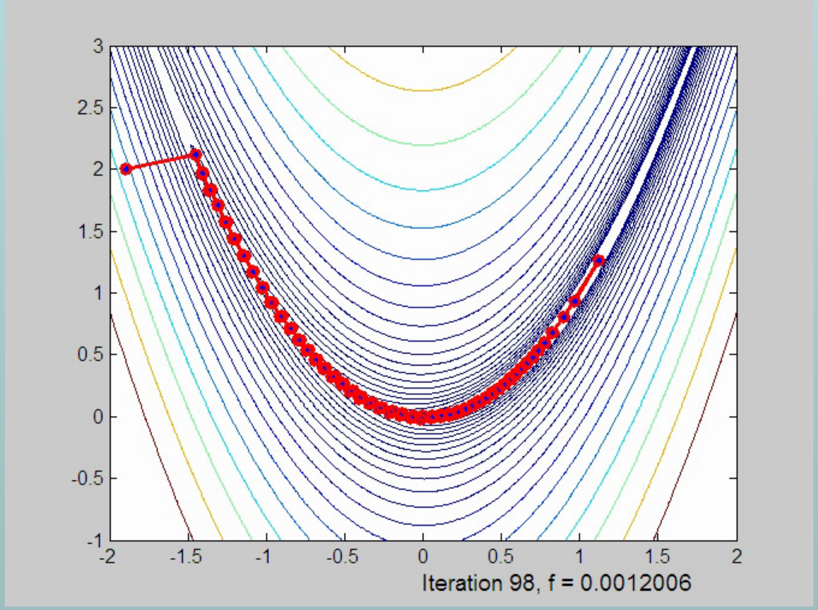



Minimum at [1, 1]

# Conjugate gradient

- Again, an explicit line minimization must be used at every step




- The algorithm converges in 98 iterations
- Far superior to steepest descent

# Quasi-Newton methods

- If the problem size is large and the Hessian matrix is dense then it may be infeasible/inconvenient to compute it directly.
- Quasi-Newton methods avoid this problem by keeping a "rolling estimate" of H(x), updated at each iteration using new gradient information.
- Common schemes are due to Broyden, Goldfarb, Fletcher and Shanno (BFGS), and also Davidson, Fletcher and Powell (DFP).
- The idea is based on the fact that for quadratic functions holds

$$\mathbf{g}_{k+1} - \mathbf{g}_k = \mathbf{H}(\mathbf{x}_{k+1} - \mathbf{x}_k)$$

and by accumulating $\mathbf{g}_k$'s and $\mathbf{x}_k$'s we can calculate $\mathbf{H}$.

# Quasi-Newton BFGS method

- Set $\mathbf{H}_0 = \mathbf{I}$.
- Update according to

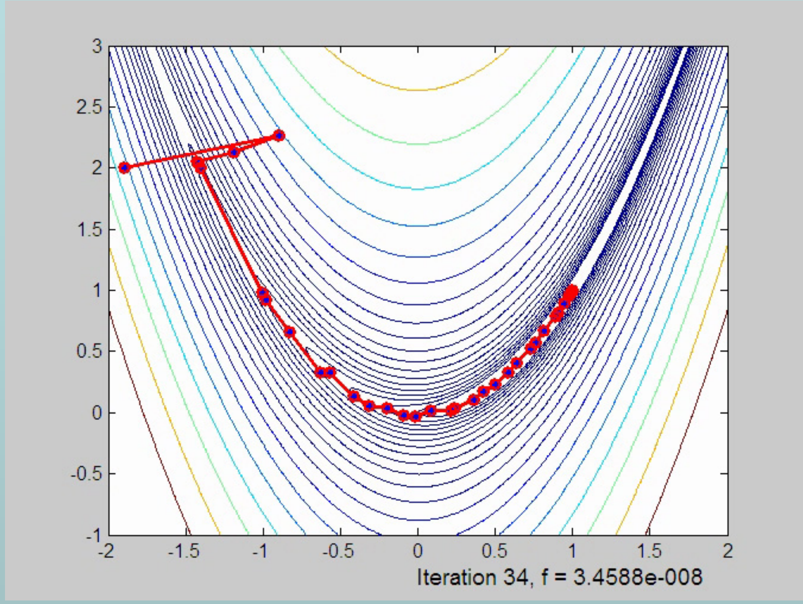
$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{\gamma_k \gamma_k^T}{\gamma_k^T \delta_k} - \frac{\mathbf{H}_k \gamma_k \gamma_k^T \mathbf{H}_k}{\delta_k^T \mathbf{H}_k \delta_k}$$

where

$$\gamma_k = \mathbf{g}_{k+1} - \mathbf{g}_k \qquad \delta_k = \mathbf{x}_{k+1} - \mathbf{x}_k$$

- The matrix inverse can also be computed in this way.
- Directions $\delta_k$'s form a conjugate set.
- $\mathbf{H}_{k+1}$ is positive definite if $\mathbf{H}_k$ is positive definite.
- The estimate $\mathbf{H}_k$ is used to form a local quadratic approximation as before

# BFGS example




- The method converges in 34 iterations, compared to 18 for the full-Newton method

# Optimization Methods for Computer Vision Applications

| Method | Alpha ($\alpha$) Calculation | Intermediate Updates | Convergence Condition |
|---|---|---|---|
| **Steepest-Descent Method (Method 1)** | Find $\alpha_k$, the value of $\alpha$ that minimizes $f(x_k + \alpha d_k)$, using line search | $x_{k+1} = x_k + \alpha_k d_k$<br>$d_k = -g_k$<br>$f_{k+1} = f(x_{k+1})$ | If $\|\|\alpha_k d_k\|\| < \in$, then $x^* = x_{k+1}$ , $f(x^*) = f_{k+1}$<br>Else $k = k + 1$ |
| **Steepest-Descent Method (Method 2)** | Without Using Line Search<br>$\alpha_k \approx \dfrac{g_k^T g_k}{g_k^T H_k g_k}$ | $x_{k+1} = x_k + \alpha_k d_k$<br>$d_k = -g_k$<br>$f_{k+1} = f(x_{k+1})$ | --" "-- |
| **Newton Method** | Find $\alpha_k$, the value of $\alpha$ that minimizes $f(x_k + \alpha d_k)$, using line search | $x_{k+1} = x_k + \alpha_k d_k$<br>$d_k = -H_k^{-1} g_k$<br>$f_{k+1} = f(x_{k+1})$ | --" "-- |
| **Gauss-Newton Method** | Find $\alpha_k$, the value of $\alpha$ that minimizes F$(x_k + \alpha d_k)$, using line search<br>$F = \sum_{p=1}^{m} f_p(x)^2 = f^T f$<br>$f = [f_1(x)\ f_2(x) \ldots f_m(x)]^T$ | $x_{k+1} = x_k + \alpha_k d_k$<br>$d_k = -H_k^{-1} g_k$<br>$g_F = 2J^T f$<br>$H \approx 2J^T J = \boldsymbol{L}^{-1}\boldsymbol{D}(\boldsymbol{L}^{\boldsymbol{T}})^{-1}$<br>$x_{k+1} = x_k - \alpha_k (J^T J)^{-1}(J^T f)$<br>$x_{k+1} = x_k - \alpha_k \boldsymbol{L}^{\boldsymbol{T}}\boldsymbol{D}\boldsymbol{L} g_k$<br>$f_{p(k+1)} = f_p(x_k)$<br>$F_{(k+1)} = F(x_k)$ | If $\|F_{k+1} - F_k\| < \in$ then $x^* = x_{k+1}$ , F$(x^*) = F_{k+1}$<br>Else $k = k + 1$ |

| Method | Alpha ($\alpha$) Calculation | Intermediate Updates | Convergence Condition |
|---|---|---|---|
| **Coordinate Descent** | Find $\alpha_k$, the value of $\alpha$ that minimizes $f(x_k + \alpha d_k)$, using line search | $x_{k+1} = x_k + \alpha_k d_k$<br>$d_k = [0\ 0\ \ldots 0\ d_k 0\ \ldots 0]^T$<br>$f_{k+1} = f(x_{k+1})$ | If $\lvert\lvert \alpha_k d_k \rvert\rvert < \in$, then $x^* = x_{k+1}$ , $f(x^*) = f_{k+1}$<br>Elseif k==n, then $x_1 = x_{k+1}, k = 1$<br>Else $k = k + 1$ |
| **Conjugate Gradient** | Without Using Line Search<br>$\alpha_k = \dfrac{g_k^T g_k}{d_k^T H_k d_k}$ | $d_0 = -g_0$<br>$x_{k+1} = x_k + \alpha_k d_k$<br>$\beta_k = \dfrac{g_{k+1}^T g_{k+1}}{g_k^T g_k}$<br>$d_{k+1} = -g_{k+1} + \beta_k d_k$<br>$f_{k+1} = f(x_{k+1})$ | If $\lvert\lvert \alpha_k d_k \rvert\rvert < \in$, then $x^* = x_{k+1}$ , $f(x^*) = f_{k+1}$<br>Else $k = k + 1$ |
| **Quasi-Newton Method** | Find $\alpha_k$, the value of $\alpha$ that minimizes $f(x_k + \alpha d_k)$, using line search | $x_{k+1} = x_k + \delta_k$<br>$\delta_k = \alpha_k d_k;\ \ d_k = -S_k g_k$<br>$x_{k+1} = x_k - \alpha_k S_k g_k$<br>$Compute\ g_{k+1}\ \ /* = g_k + H\delta_k\ \ */$<br>$S_0 = I_n$<br>$S_{k+1} = S_k + \dfrac{(\delta_k - S_k\gamma_k)(\delta_k - S_k\gamma_k)^T}{\gamma_k^T(\delta_k - S_k\gamma_k)}$<br>$\gamma_k = g_{k+1} - g_k$ | If $\lvert\lvert \delta_k \rvert\rvert < \in$, then $x^* = x_{k+1}$ , $f(x^*) = f_{k+1}$<br>Else $k = k + 1$ |

# Non-linear least squares

- It is very common in applications for a cost function $f(\mathbf{x})$ to be the sum of a large number of squared residuals

$$f(\mathrm{x}) = \sum_{i=1}^{M} r_i^2(\mathrm{x})$$

- If each residual depends non-linearly on the parameters $\mathbf{x}$ then the minimization of $f(\mathbf{x})$ is a non-linear least squares problem.

# Non-linear least squares

$$f(\mathbf{x}) = \sum_{i=1}^{M} r_i^2(\mathbf{x})$$

- The M × N Jacobian of the vector of residuals $r$ is defined as

$$J(\mathbf{x}) = \begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_1}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_M}{\partial x_1} & \cdots & \frac{\partial r_M}{\partial x_N} \end{bmatrix}$$

- Consider

$$\frac{\partial f}{\partial x_k} = \frac{\partial}{\partial x_k} \sum_i r_i^2 = \sum_i 2 r_i \frac{\partial r_i}{\partial x_k}$$

- Hence

$$\nabla f(\mathbf{x}) = 2\mathbf{J}^T \mathbf{r}$$

# Non-linear least squares

- For the Hessian holds

$$\frac{\partial^2 f}{\partial x_k \partial x_l} = \underbrace{2\sum_i \frac{\partial r_i}{\partial x_l}\frac{\partial r_i}{\partial x_k}}_{\mathbf{H}(\mathbf{x}) \approx 2\mathbf{J}^T\mathbf{J}} + 2\sum_i r_i \frac{\partial^2 r_i}{\partial x_k \partial x_l}$$

$$\mathbf{H}(\mathbf{x}) \approx 2\mathbf{J}^T\mathbf{J}$$

Gauss-Newton approximation

- Note that the second-order term in the Hessian is multiplied by the residuals $r_i$.
- In most problems, the residuals will typically be small.
- Also, at the minimum, the residuals will typically be distributed with mean = 0.
- For these reasons, the second-order term is often ignored.
- Hence, explicit computation of the full Hessian can again be avoided.

# Gauss-Newton example

- The minimization of the Rosenbrock function

$$f(x, y) = 100(y - x^2)^2 + (1 - x)^2$$

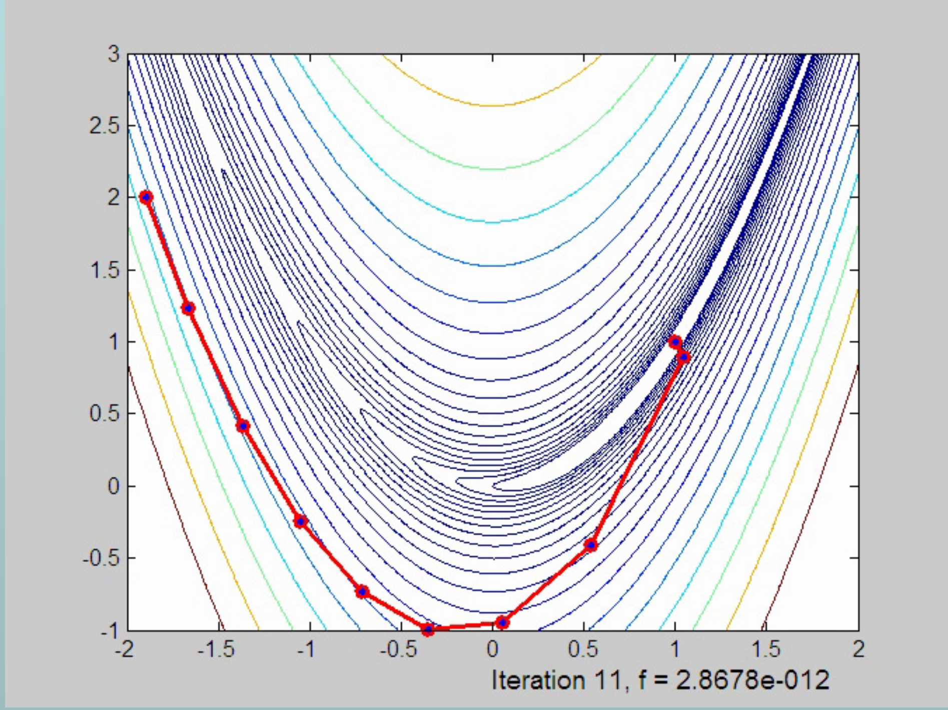
- can be written as a least-squares problem with residual vector

$$\mathbf{r} = \begin{bmatrix} 10(y - x^2) \\ (1 - x) \end{bmatrix}$$

$$\mathbf{J} = \begin{bmatrix} \frac{\partial r_1}{\partial x} & \frac{\partial r_1}{\partial y} \\ \frac{\partial r_2}{\partial x} & \frac{\partial r_2}{\partial y} \end{bmatrix} = \begin{bmatrix} -20x & 10 \\ -1 & 0 \end{bmatrix}$$

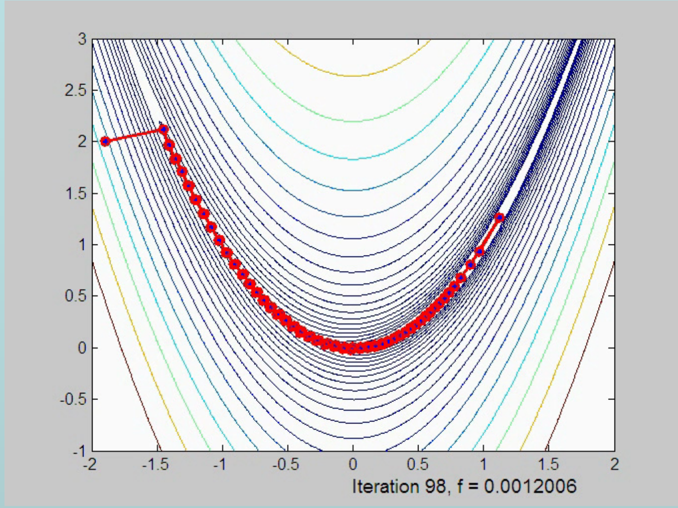
# Gauss-Newton example

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \mathbf{H}_k^{-1} \mathbf{g}_k \qquad \mathbf{H}_k = 2\mathbf{J}_k^T \mathbf{J}$$

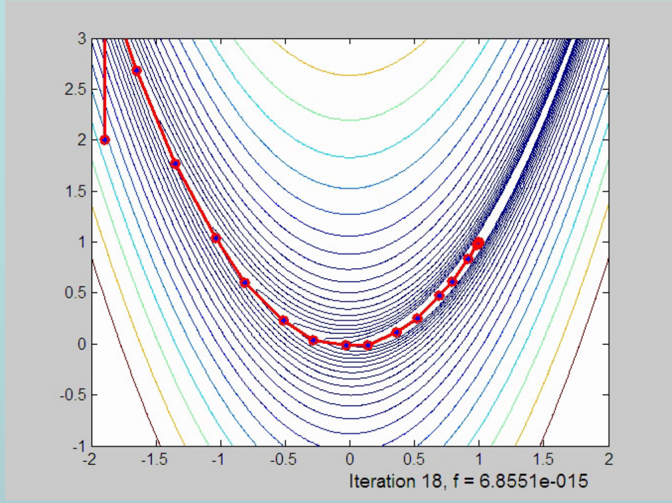



- minimization with the Gauss-Newton approximation with line search takes only 11 iterations

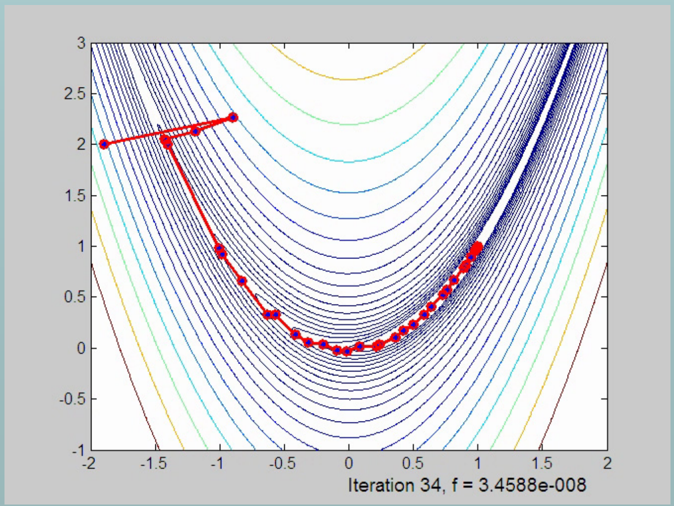
# Comparison



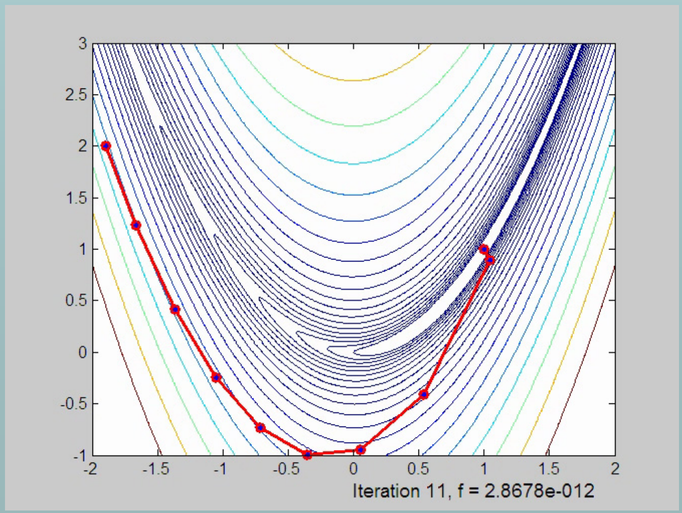

CG




Newton

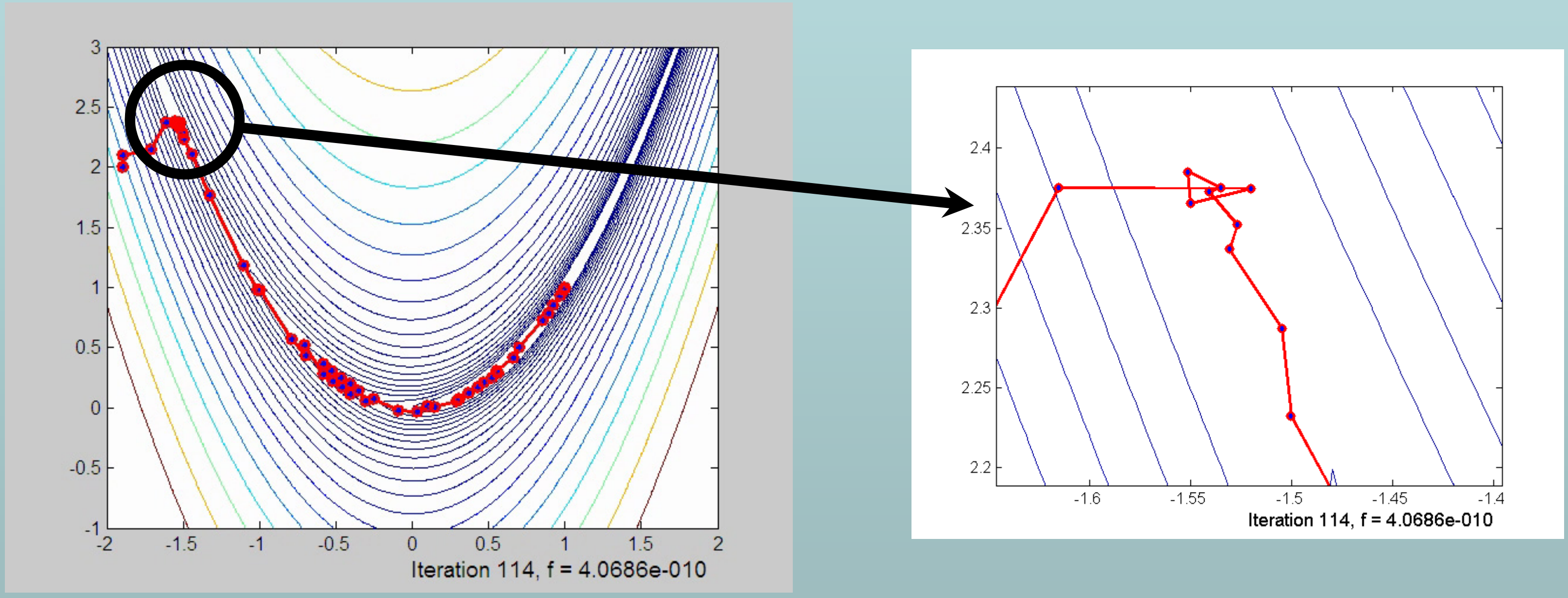



Quasi-Newton




Gauss-Newton

# Simplex



# Constrained Optimization

$$f(\mathbf{x}) : \mathbb{R}^N \longrightarrow \mathbb{R}$$

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x})$$

Subject to:

- Equality constraints: $a_i(\mathbf{x}) = 0 \qquad i = 1, 2, \ldots, p$

- Nonequality constraints: $c_j(\mathbf{x}) \geq 0 \qquad j = 1, 2, \ldots, q$

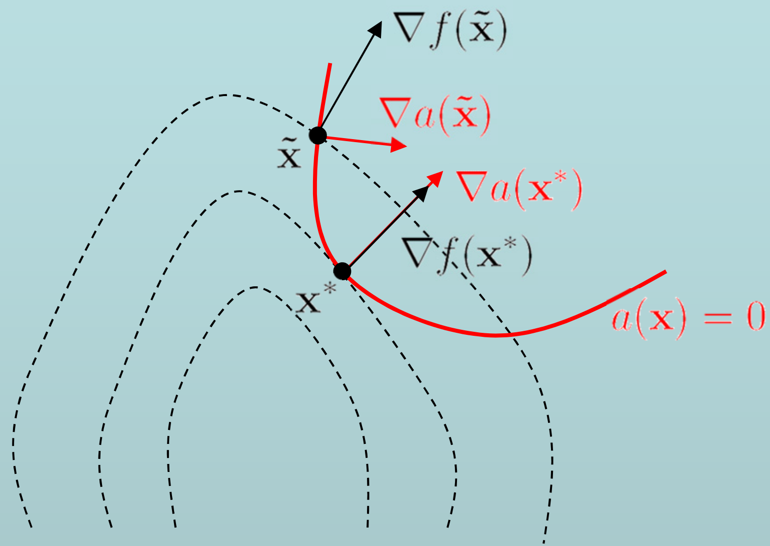
- Constraints define a feasible region, which is nonempty.

- The idea is to convert it to an unconstrained optimization.

# Equality constraints

- Minimize $f(\mathbf{x})$ subject to: $a_i(\mathbf{x}) = 0$ for $i = 1, 2, \ldots, p$
- The gradient of $f(\mathbf{x})$ at a local minimizer is equal to the linear combination of the gradients of $a_i(\mathbf{x})$ with Lagrange multipliers as the coefficients.

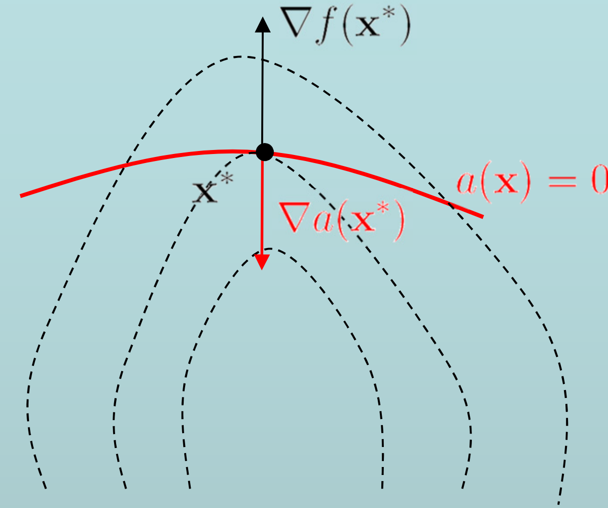
$$\nabla f(\mathbf{x}^*) = \sum_{i=1}^{p} \lambda_i^* \nabla a_i(\mathbf{x}^*)$$




$f_3 > f_2 > f_1$

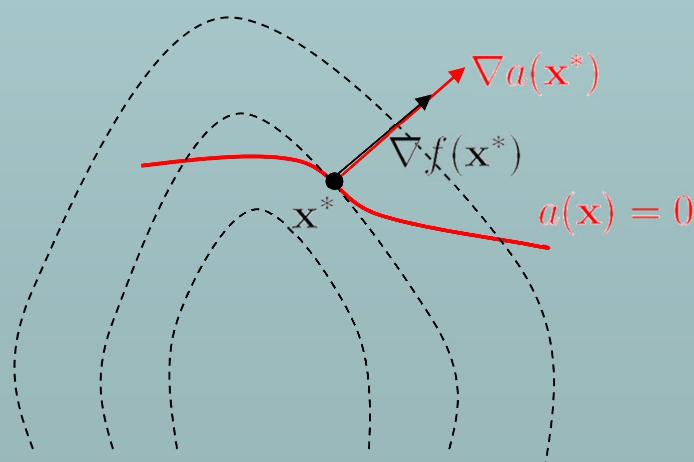
$\tilde{x}$ is not a minimizer

x* is a minimizer, $\lambda^*>0$




$f_3 > f_2 > f_1$

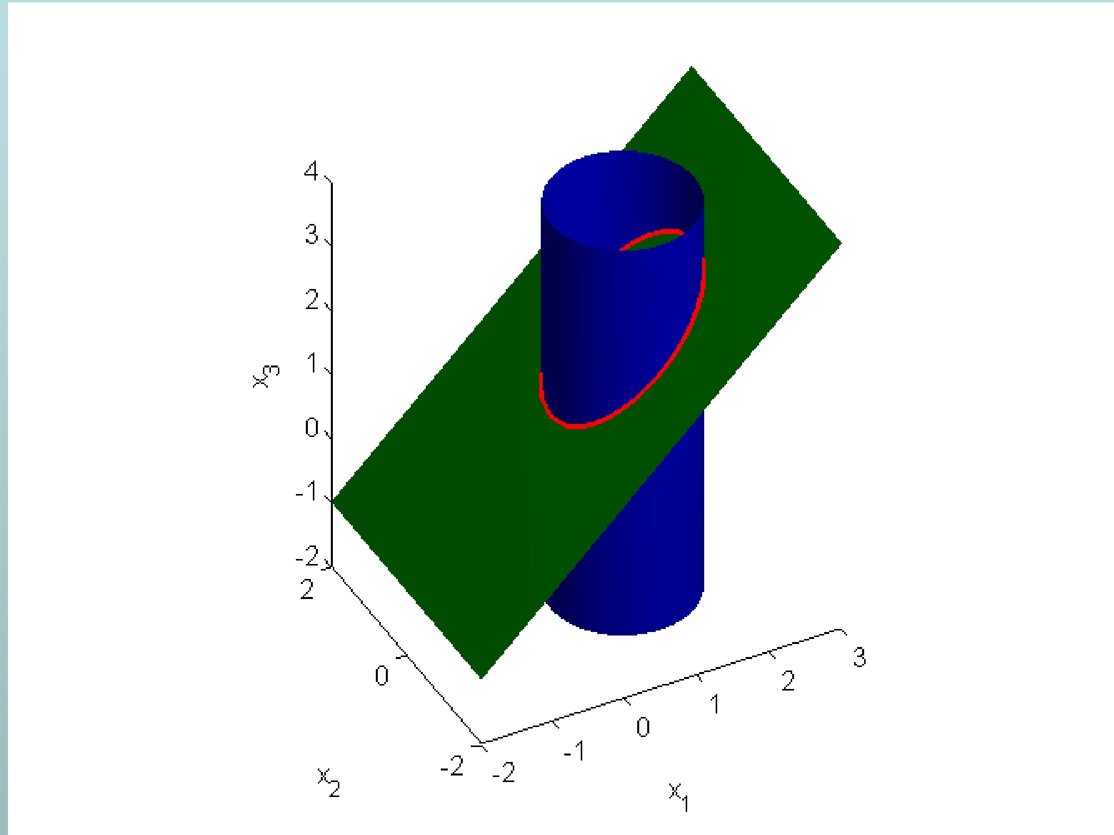
x* is a minimizer, $\lambda^*<0$




$f_3 > f_2 > f_1$

x* is not a minimizer

# 3D Example

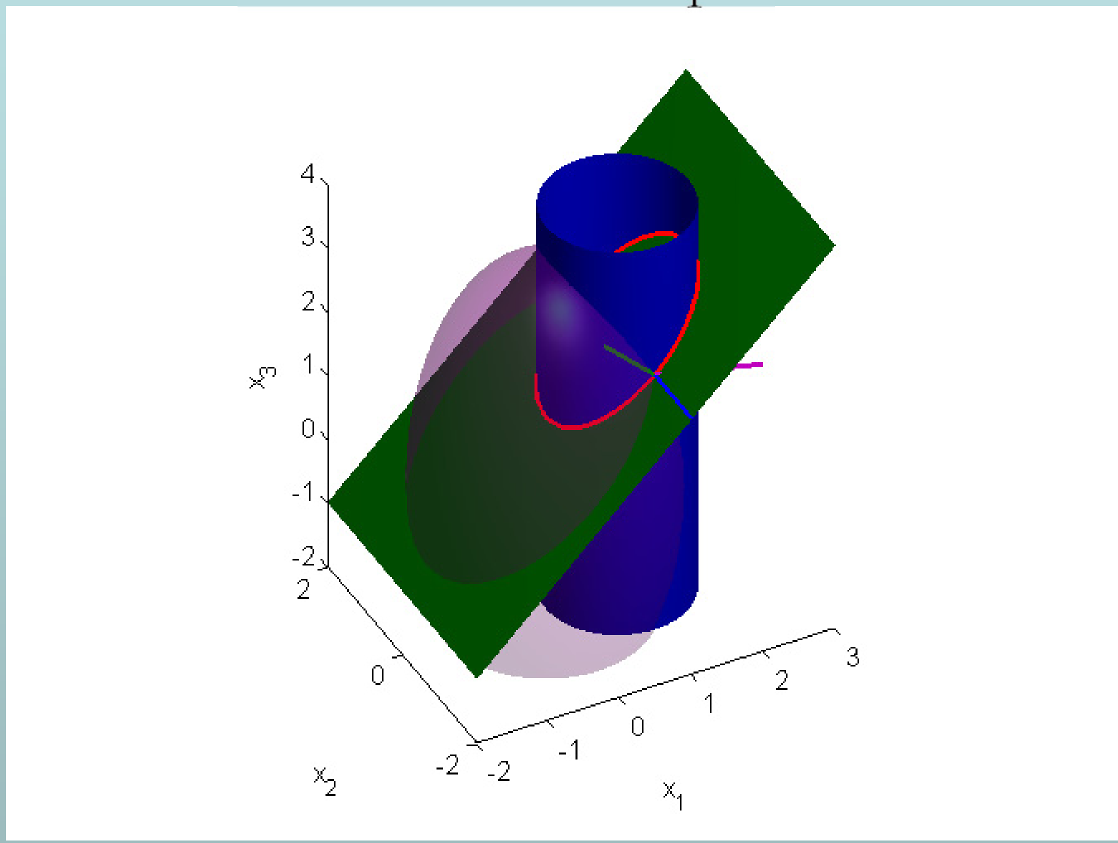



$$\boxed{a_1(\mathbf{x}) = -x_1 + x_3 - 1 = 0}$$

$$\boxed{a_2(\mathbf{x}) = x_1^2 + x_2^2 - 2x_1 = 0}$$

# 3D Example

$$f(\mathbf{x}) = x_1^2 + x_2^2 + \frac{1}{4}x_3^2$$

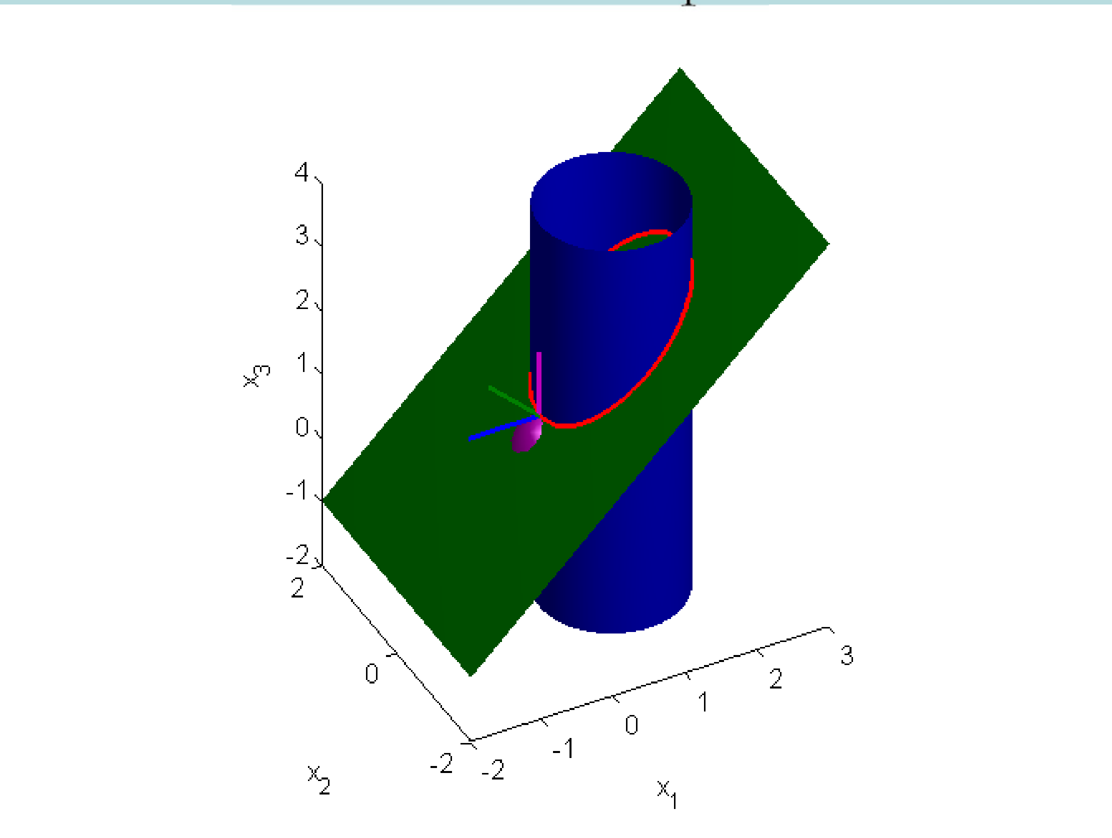



$$f(\mathbf{x}) = 3$$

Gradients of constraints and objective function are linearly independent.

# 3D Example

$$f(\mathbf{x}) = x_1^2 + x_2^2 + \frac{1}{4}x_3^2$$



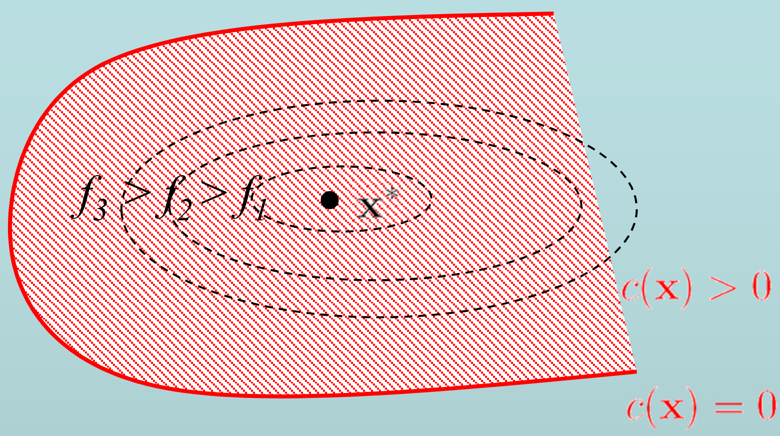

$$f(\mathbf{x}) = 1$$

Gradients of constraints and objective function are linearly dependent.

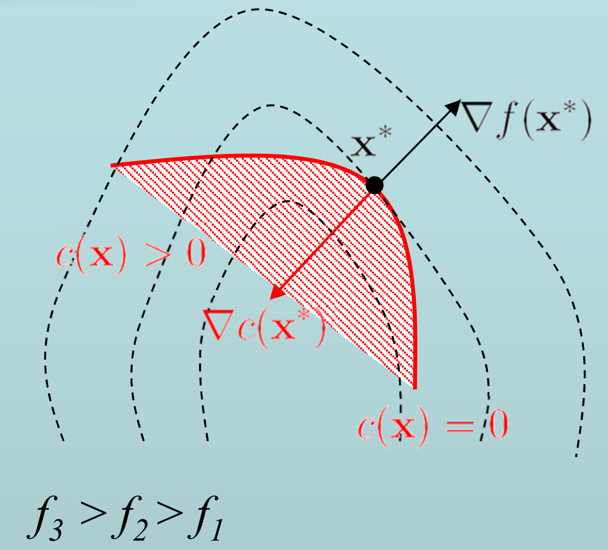
# Inequality constraints

- Minimize $f(\mathbf{x})$ subject to: $c_j(\mathbf{x}) \geq 0$ for $j = 1, 2, \ldots, q$
- The gradient of $f(\mathbf{x})$ at a local minimizer is equal to the linear combination of the gradients of $c_j(\mathbf{x})$, which are active ( $c_j(\mathbf{x}) = 0$ )
- and Lagrange multipliers must be positive, $\mu_j \geq 0,\ j \in A$

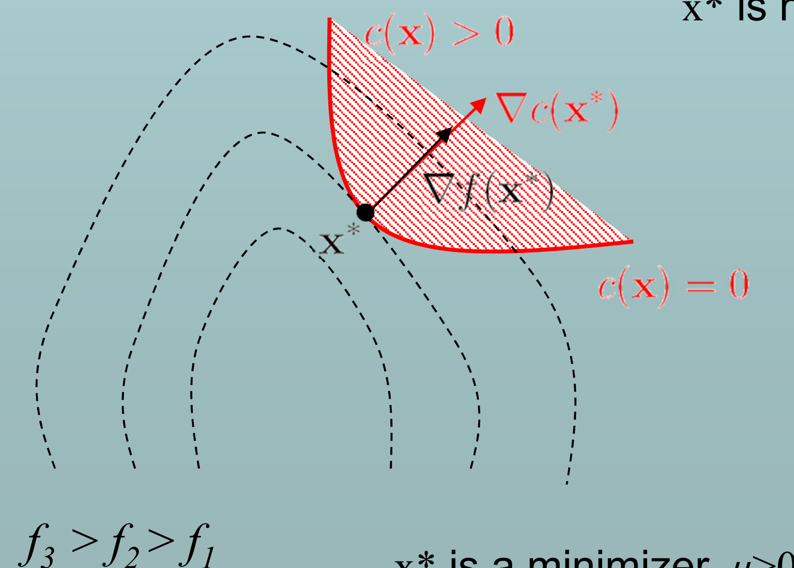
$$\nabla f(\mathbf{x}^*) = \sum_{j \in A} \mu_j^* \nabla c_j(\mathbf{x}^*)$$




No active constraints
at x*, $\nabla f(\mathbf{x}) = \mathbf{0}$




x* is not a minimizer, $\mu<0$




x* is a minimizer, $\mu>0$

# Lagrangien

- We can introduce the function (Lagrangien)

$$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) - \sum_{i=1}^{p} \lambda_i a_i(\mathbf{x}) - \sum_{j=1}^{q} \mu_j c_j(\mathbf{x})$$

- The necessary condition for the local minimizer is

$$\nabla_x L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \mathbf{0}$$

  and it must be a feasible point (i.e. constraints are satisfied).

- These are Karush-Kuhn-Tucker conditions

# Quadratic Programming (QP)

- Like in the unconstrained case, it is important to study quadratic functions. Why?
- Because general nonlinear problems are solved as a sequence of minimizations of their quadratic approximations.
- QP with constraints

Minimize $$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{p}$$

subject to linear constraints.

- $\mathbf{H}$ is symmetric and positive semidefinite.

# QP with Equality Constraints

- Minimize $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{p}$

  Subject to: $\mathbf{A}\mathbf{x} = \mathbf{b}$

- Ass.: $\mathbf{A}$ is $p \times N$ and has full row rank ($p<N$)
- Convert to unconstrained problem by variable elimination:

$$\mathbf{x} = \mathbf{Z}\phi + \mathbf{A}^+\mathbf{b}$$

$\mathbf{Z}$ is the null space of $\mathbf{A}$

$\mathbf{A}^+$ is the pseudo-inverse.

Minimize $\hat{f}(\phi) = \frac{1}{2}\phi^T\hat{\mathbf{H}}\phi + \phi^T\hat{\mathbf{p}}$

$$\hat{\mathbf{H}} = \mathbf{Z}^T\mathbf{H}\mathbf{Z}$$

$$\hat{\mathbf{p}} = \mathbf{Z}^T(\mathbf{H}\mathbf{A}^+\mathbf{b} + \mathbf{p})$$

This quadratic unconstrained problem can be solved, e.g., by Newton method.

# QP with inequality constraints

- Minimize $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{x}^T\mathbf{p}$

  Subject to: $\mathbf{A}\mathbf{x} \geq \mathbf{b}$

- First we check if the unconstrained minimizer $\mathbf{x}^* = -\mathbf{H}^{-1}\mathbf{p}$ is feasible.

  If yes we are done.

  If not we know that the minimizer must be on the boundary and we proceed with an active-set method.

  - $\mathbf{x}_k$ is the current feasible point
  - $\mathcal{A}_k$ is the index set of active constraints at $\mathbf{x}_k$
  - Next iterate is given by $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$

# Active-set method

$$\mathbf{A}^T = [\mathbf{a}_1 \dots \mathbf{a}_p]$$

- $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$ How to find $\mathbf{d}_k$?
  - To remain active $\mathbf{a}_j^T \mathbf{x}_{k+1} - b_j = 0$ thus $\mathbf{a}_j^T \mathbf{d}_k = 0 \quad j \in \mathcal{A}_k$
  - The objective function at $\mathbf{x}_k$+$\mathbf{d}$ becomes

$$f_k(\mathbf{d}) = \frac{1}{2}\mathbf{d}^T \mathbf{H} \mathbf{d} + \mathbf{d}^T \mathbf{g}_k + f(\mathbf{x}_k)$$ where $\mathbf{g}_k = \nabla f(\mathbf{x}_k)$

- The major step is a QP sub-problem

$$\mathbf{d}_k = \arg\min_{\mathbf{d}} \frac{1}{2}\mathbf{d}^T \mathbf{H} \mathbf{d} + \mathbf{d}^T \mathbf{g}_k$$

subject to: $\mathbf{a}_j^T \mathbf{d} = 0 \quad j \in \mathcal{A}_k$

- Two situations may occur: $\mathbf{d}_k = \mathbf{0}$ or $\mathbf{d}_k \neq \mathbf{0}$

# Active-set method

- $\mathbf{d}_k = \mathbf{0}$

  We check if KKT conditions are satisfied

  $$\nabla_x L(\mathbf{x}, \boldsymbol{\mu}) = \mathbf{H}\mathbf{x}_k + \mathbf{p} - \sum_{j \in \mathcal{A}_k} \mu_j \mathbf{a}_j = \mathbf{0} \quad \text{and} \quad \mu_j \geq 0$$

  If YES we are done.

  If NO we remove the constraint from the active set $\mathcal{A}_k$ with the most negative $\mu_j$ and solve the QP sub-problem again but this time with less active constraints.

- $\mathbf{d}_k \neq \mathbf{0}$

  We can move to $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}_k$ but some inactive constraints may be violated on the way.

  In this case, we move by $\alpha_k \mathbf{d}_k$ till the first inactive constraint becomes active, update $\mathcal{A}_k$, and solve the QP sub-problem again but this time with more active constraints.

# General Nonlinear Optimization

- Minimize $f(\mathbf{x})$
  subject to: $a_i(\mathbf{x}) = 0$
  $c_j(\mathbf{x}) \geq 0$
  where the objective function and constraints are nonlinear.

1. For a given $\{\mathbf{x}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k\}$ approximate Lagrangien by Taylor series → QP problem
2. Solve QP → descent direction $\{\boldsymbol{\delta}_x, \boldsymbol{\delta}_\lambda, \boldsymbol{\delta}_\mu\}$
3. Perform line search in the direction $\boldsymbol{\delta}_x$ → $\mathbf{x}_{k+1}$
4. Update Lagrange multipliers → $\{\boldsymbol{\lambda}_{k+1}, \boldsymbol{\mu}_{k+1}\}$
5. Repeat from Step 1.

# General Nonlinear Optimization

Lagrangien $$L(\mathbf{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = f(\mathbf{x}) - \sum_{i=1}^{p} \lambda_i a_i(\mathbf{x}) - \sum_{j=1}^{q} \mu_j c_j(\mathbf{x})$$

At the $k$th iterate: $\{\mathbf{x}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k\}$
and we want to compute a set of increments: $\{\boldsymbol{\delta}_x, \boldsymbol{\delta}_\lambda, \boldsymbol{\delta}_\mu\}$

First order approximation of $\nabla_x L$ and constraints:

- $\nabla_x L(\mathbf{x}_{k+1}, \boldsymbol{\lambda}_{k+1}, \boldsymbol{\mu}_{k+1}) \approx \nabla_x L(\mathbf{x}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k) +$

  $+\nabla_x^2 L(\mathbf{x}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)\boldsymbol{\delta}_x + \nabla_{x\lambda}^2 L(\mathbf{x}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)\boldsymbol{\delta}_\lambda + \nabla_{x\mu}^2 L(\mathbf{x}_k, \boldsymbol{\lambda}_k, \boldsymbol{\mu}_k)\boldsymbol{\delta}_\mu = \mathbf{0}$

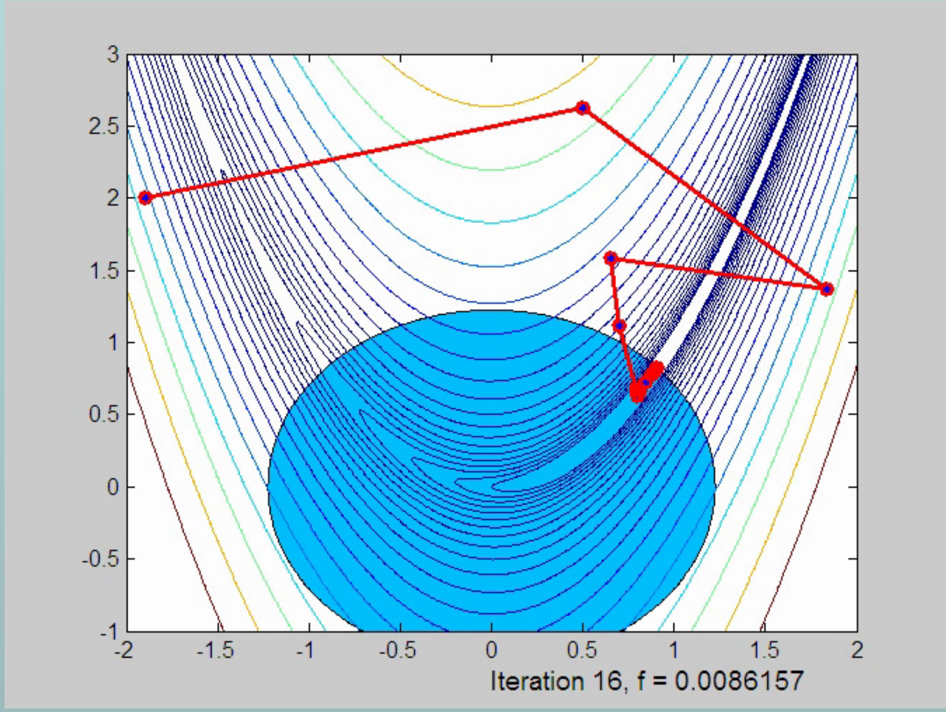
- $c_i(\mathbf{x}_k \boldsymbol{\delta}_x) \approx c_i(\mathbf{x}_k) + \boldsymbol{\delta}_x^T \nabla_x c_i(\mathbf{x}_k) \geq 0$
- $a_i(\mathbf{x}_k \boldsymbol{\delta}_x) \approx a_i(\mathbf{x}_k) + \boldsymbol{\delta}_x^T \nabla_x a_i(\mathbf{x}_k) = 0$

These approximate KKT conditions corresponds to a QP program

# SQP example

Minimize $f(x,y) = 100(y-x^2)^2 + (1-x)^2$

subject to: $1.5 - x_1^2 - x_2^2 \geq 0$



# Linear Programming (LP)

- LP is common in economy and is meaningful only if it is with constraints.
- Two forms:
  1. Minimize $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$

     subject to: $\mathbf{A}\mathbf{x} = \mathbf{b}$, $\mathbf{x} \geq 0$

     $\mathbf{A}$ is $p \times N$ and has full row rank ($p<N$)
  2. Minimize $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$

     subject to: $\mathbf{A}\mathbf{x} \geq \mathbf{b}$

- QP can solve LP.
- If the LP minimizer exists it must be one of the vertices of the feasible region.

  Prove it!
- A fast method that considers vertices is the Simplex method.