

Optimization Methods

Optimization models

- Single x Multiobjective models
- Static x Dynamic models
- Deterministic x Stochastic models

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

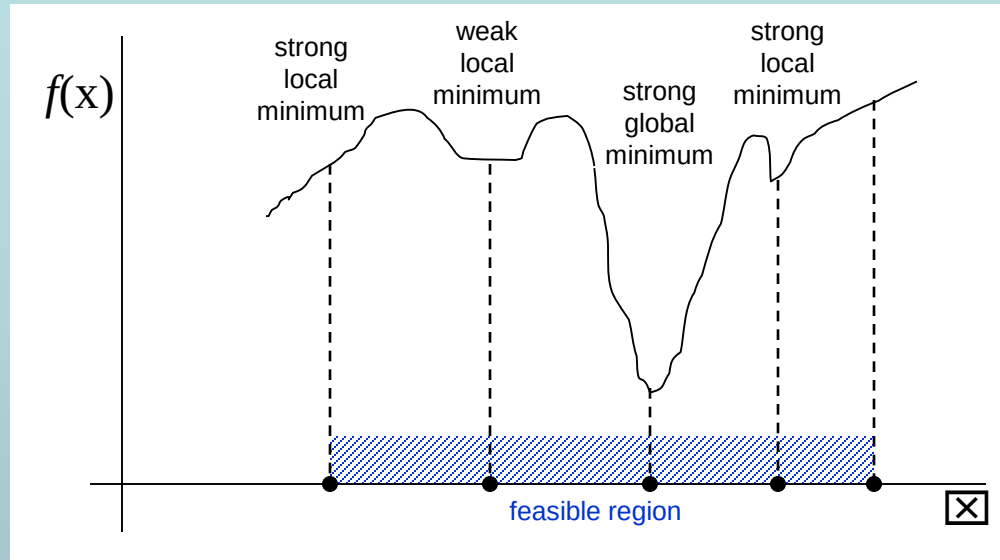
$$\mathbf{x}^* = \arg \min_{\mathbf{x}} f(\mathbf{x})$$

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 $-f(\mathbf{x})$

Types of minima

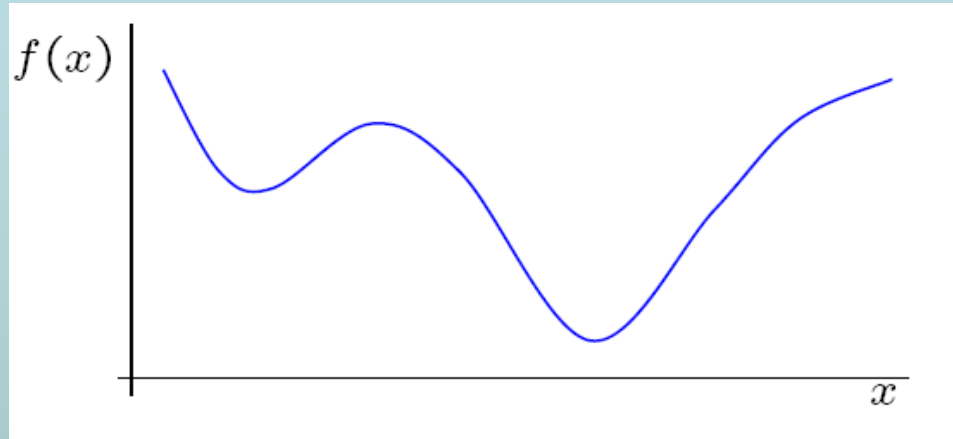


- which of the minima is found depends on the starting point
- such minima often occur in real applications

Unconstrained **univariate** optimization

Assume we can start close to the global minimum

$$\min_x f(x)$$



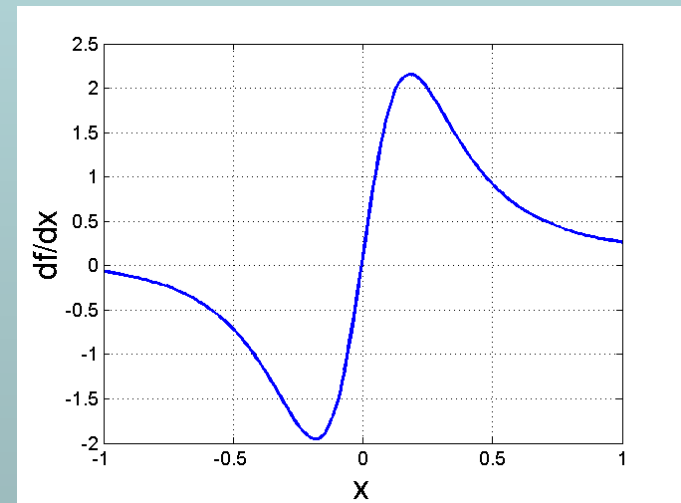
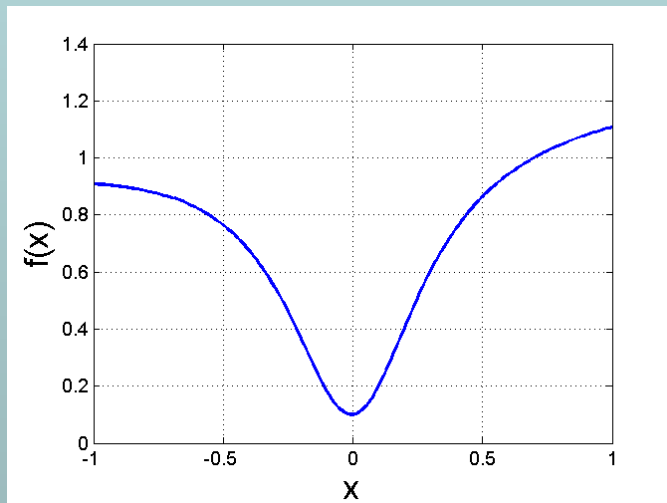
How to determine the minimum?

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

1D function

As an example consider the function

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

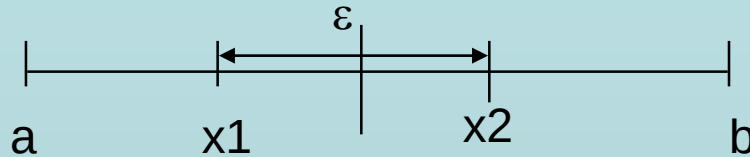


(assume we do not know the actual function expression from now on)

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.

Basic bracketing algorithm



Two point search (dichotomous search) for finding the solution to minimizing $f(x)$:

0) assume an interval $[a,b]$

1) Find $x_1 = a + (b-a)/2 - \varepsilon/2$ and $x_2 = a + (b-a)/2 + \varepsilon/2$ where ε is the resolution.

2) Compare $f(x_1)$ and $f(x_2)$

3) If $f(x_1) < f(x_2)$ then eliminate $x > x_2$ and set $b = x_2$

If $f(x_1) > f(x_2)$ then eliminate $x < x_1$ and set $a = x_1$

If $f(x_1) = f(x_2)$ then pick another pair of points

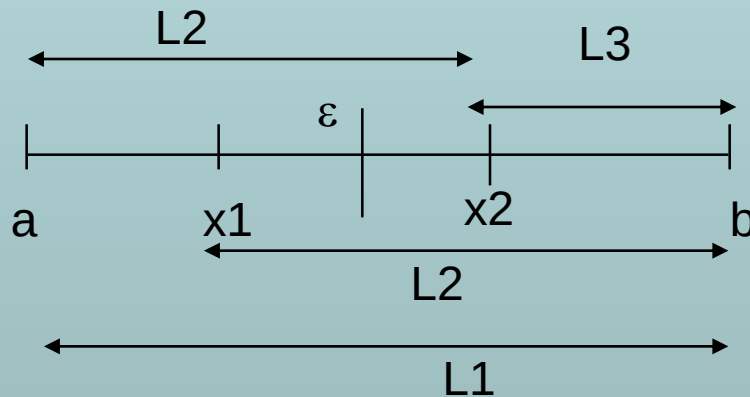
4) Continue placing point pairs until interval $< 2 \varepsilon$

Fibonacci Search

Fibonacci numbers are:

1,1,2,3,5,8,13,21,34,.. that is , the sum of the last 2 numbers

$$F_n = F_{n-1} + F_{n-2}$$

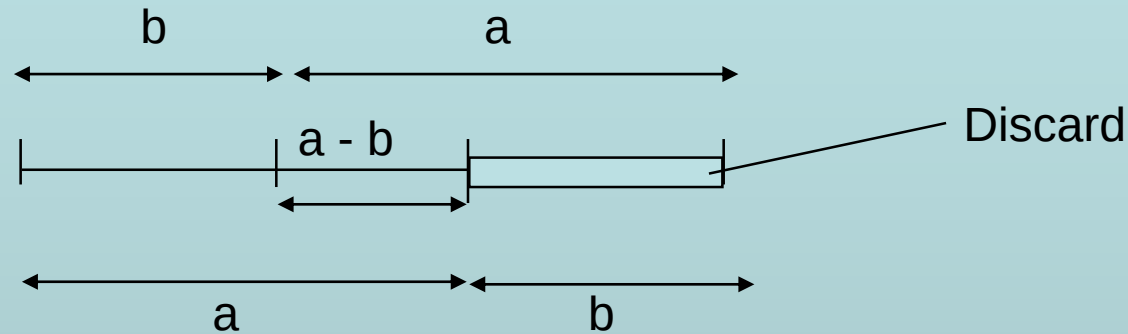


$$L1 = L2 + L3$$

It can be derived that

$$L_n = (L1 + F_{n-2} \epsilon) / F_n$$

Golden Section



In Golden Section, you try to have $b/(a-b) = a/b$

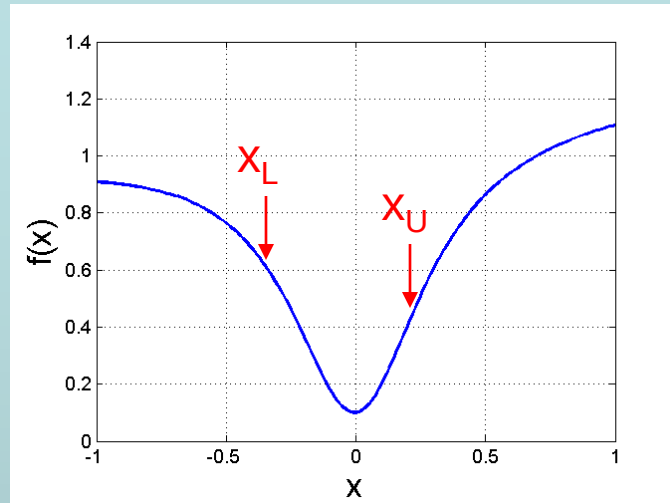
which implies $b^2 = a^2 - ab$

Solving this gives $a = (b \pm b \cdot \sqrt{5}) / 2$

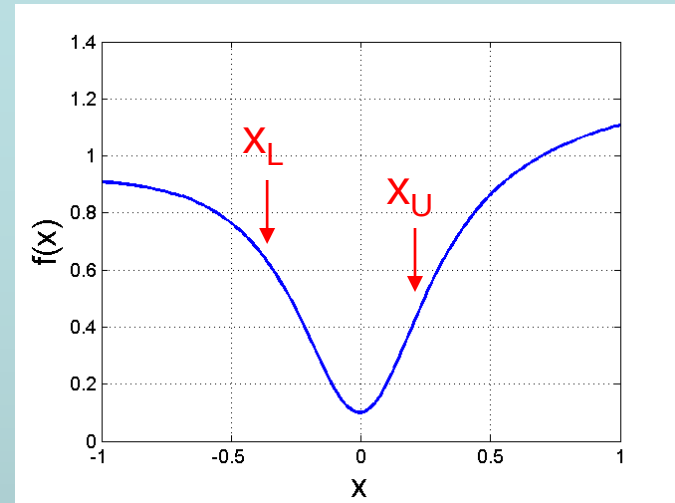
$a/b = -0.618$ or 1.618 (Golden Section ratio)

Note that $1/1.618 = 0.618$

Search methods



Dichotomous

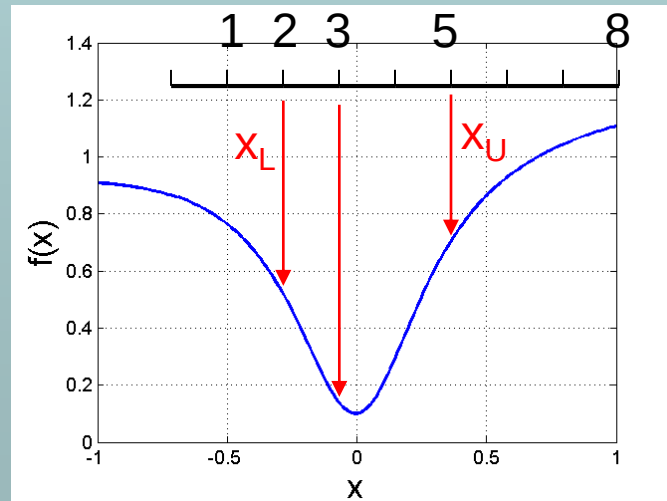


Fibonacci:

1 1 2 3 5 8 ...

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

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



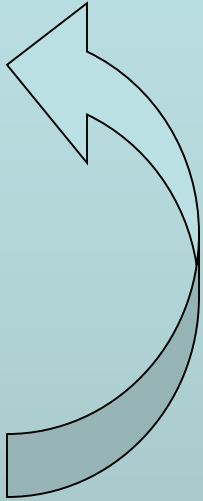
Golden-Section Search
divides intervals by

$$K = 1.6180$$

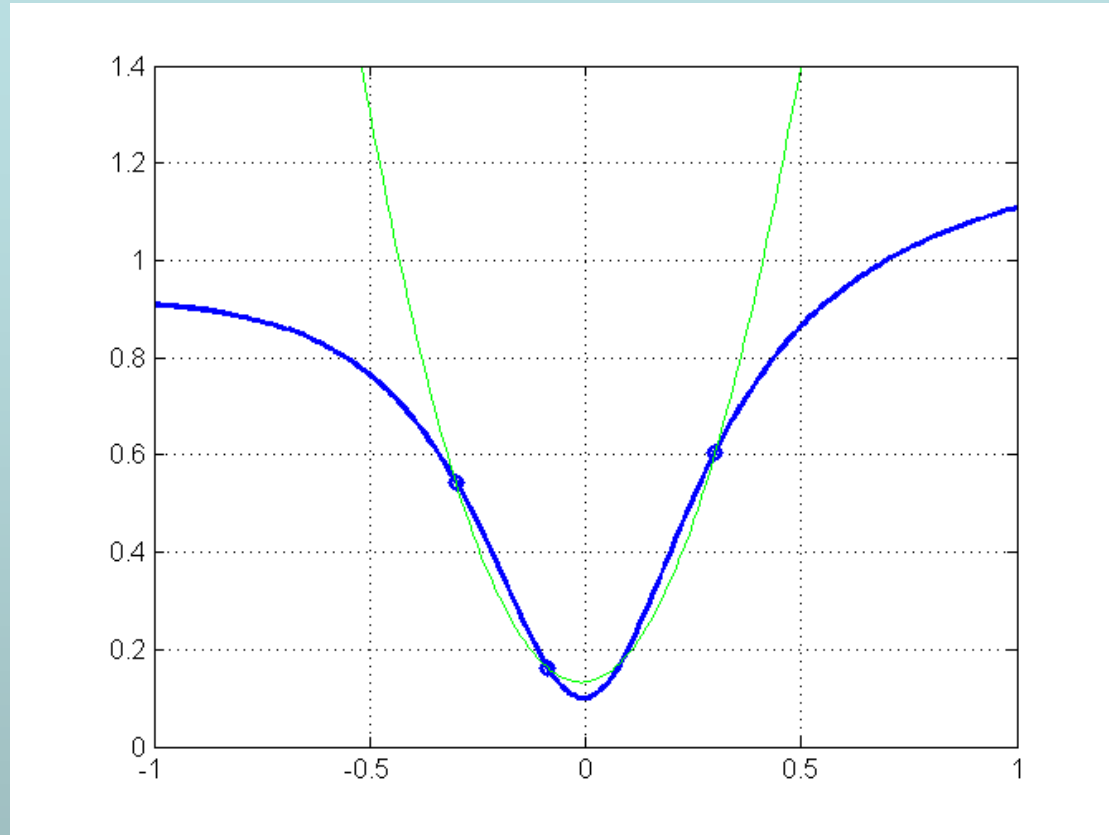
$$\frac{I_k}{I_{k+1}} = K$$

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.

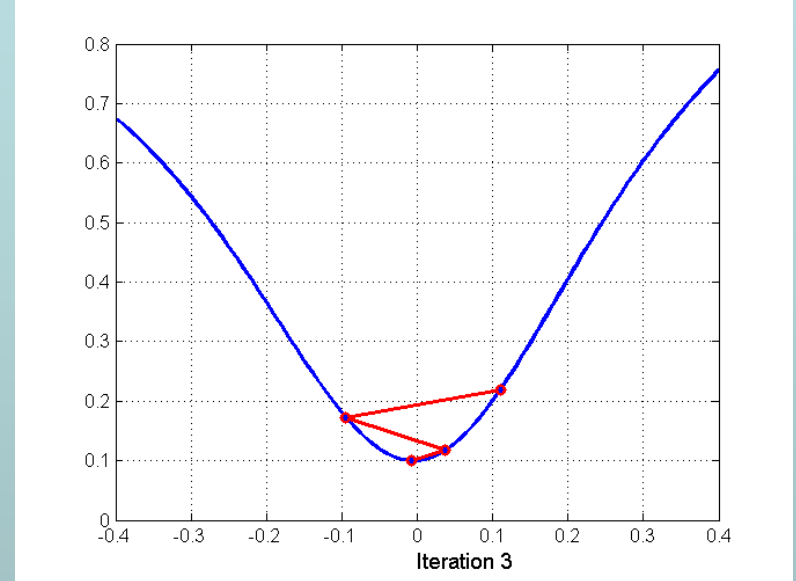
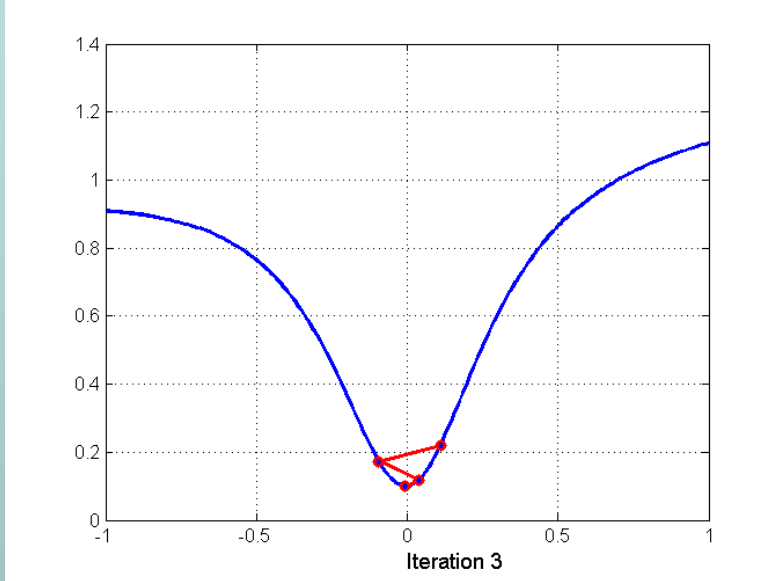
$$f(x + \delta x) = f(x) + f'(x)\delta x + \frac{1}{2}f''(x)\delta x^2 + o(\delta x^2)$$

- Find the δx which minimizes this local quadratic approximation.

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

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

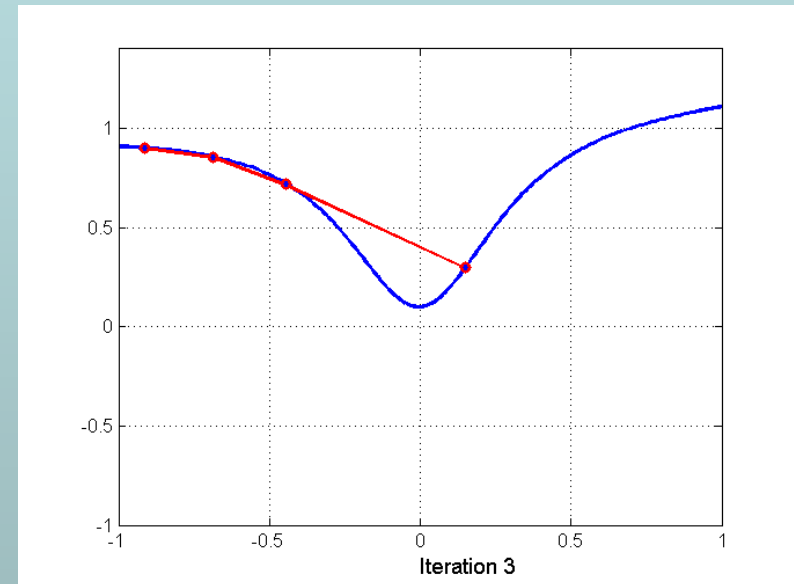
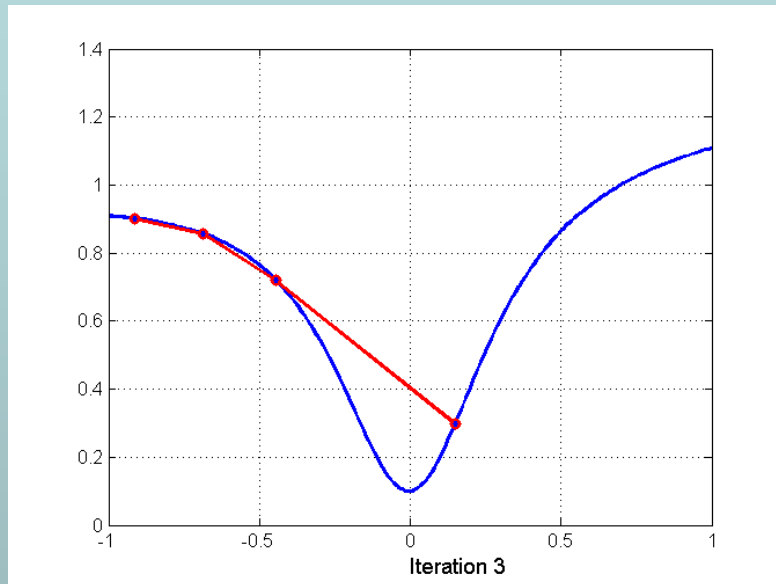
Newton method



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

Newton method

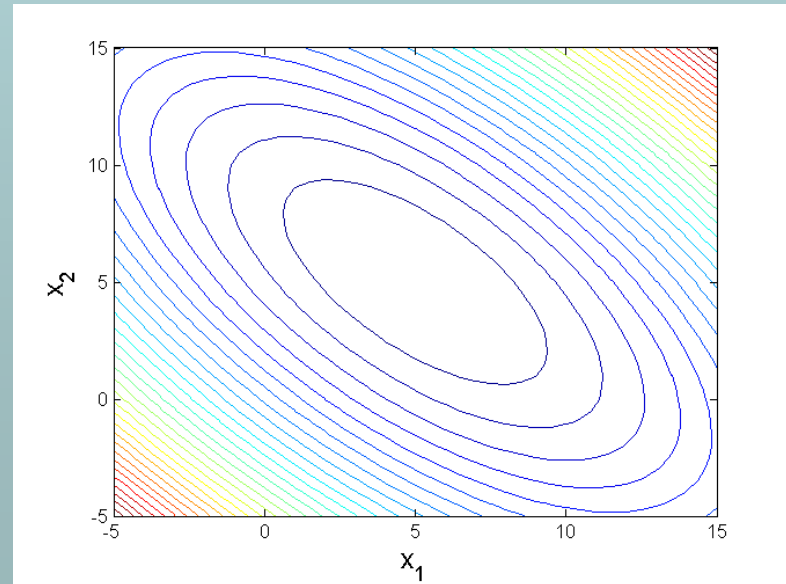
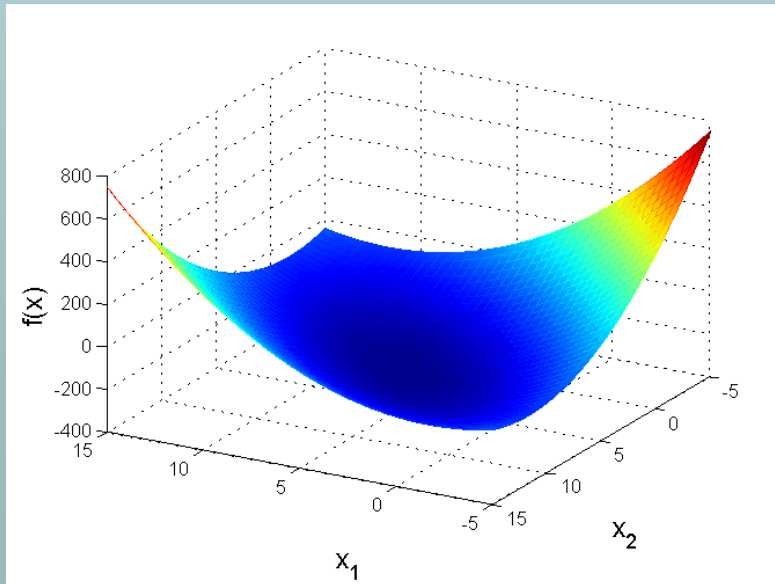
- Global convergence of Newton's method is poor.
- Often fails if the starting point is too far from the minimum.



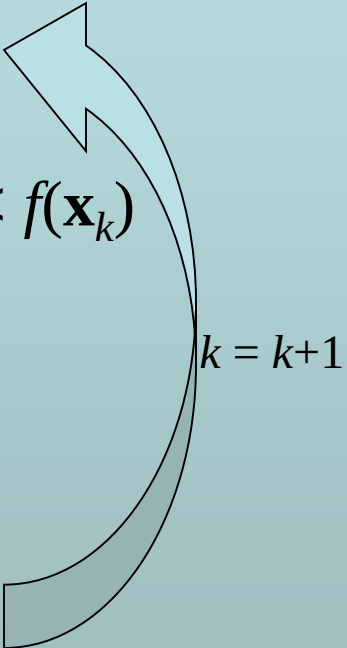
- in practice, must be used with a globalization strategy which reduces the step length until function decrease is assured

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 α_k , such that $f(\mathbf{x}_k + \alpha_k \mathbf{p}_k) < f(\mathbf{x}_k)$
 3. Update $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
 4. Check for convergence (stopping criteria)
e.g. $df/d\mathbf{x} = \mathbf{0}$ or $\frac{\|x_{k+1} - x_k\|}{\|x_k\|} < \epsilon$
- 

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

Rates of Convergence

$$\beta = \lim_{k \rightarrow \infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^p}$$

x^* ... minimum

p ... order of convergence

β ... convergence ratio

Linear conv.: $p=1, \beta < 1$

Superlinear conv.: $p=1, \beta=0$ or $p \geq 2$

Quadratic conv.: $p=2$

Taylor expansion

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

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

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

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

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

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

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

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

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

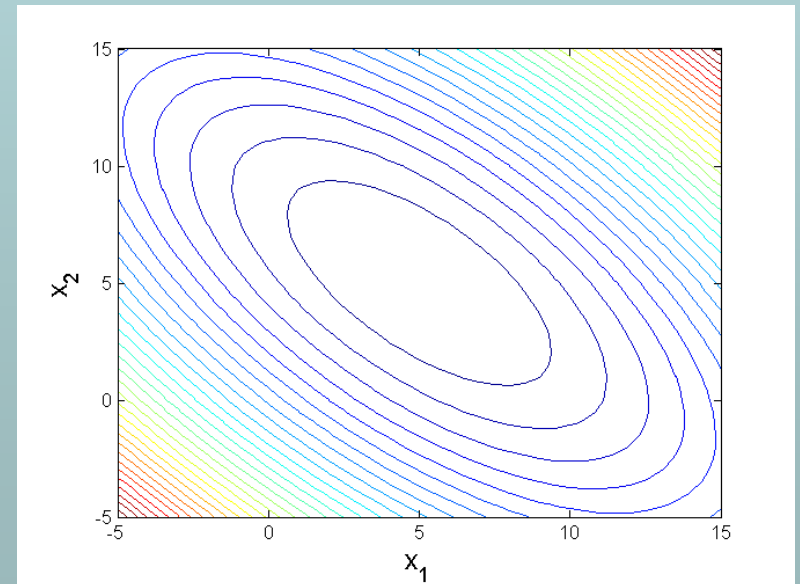
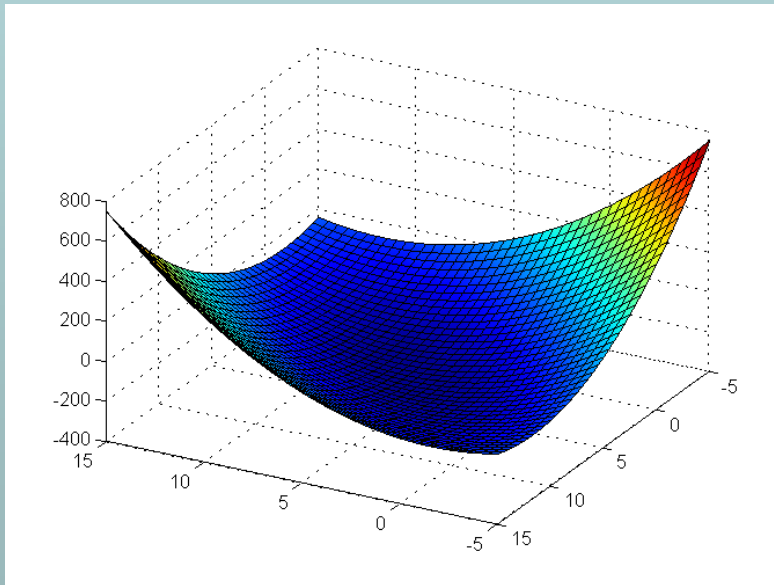
- As $|\alpha|$ increases, $f(\mathbf{x}^* + \alpha \mathbf{u}_i)$ increases, decreases or is unchanging according to whether λ_i is positive, negative or zero

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



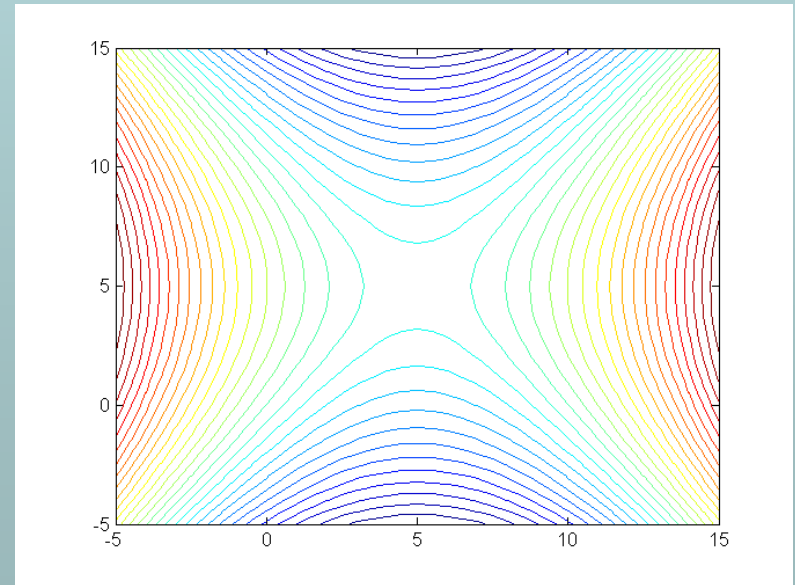
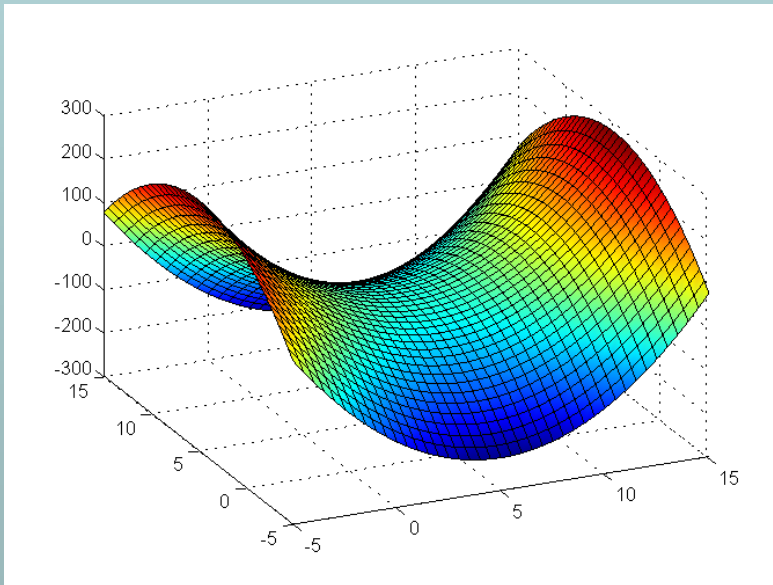
minimum

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



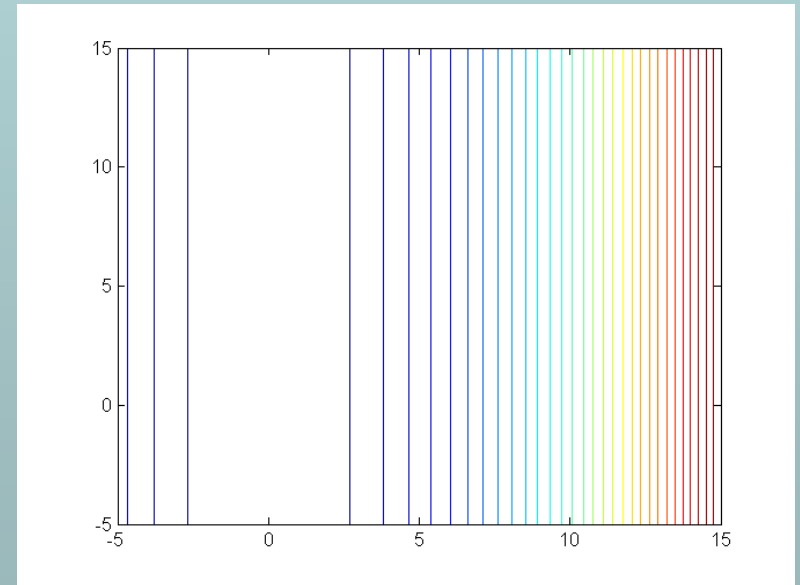
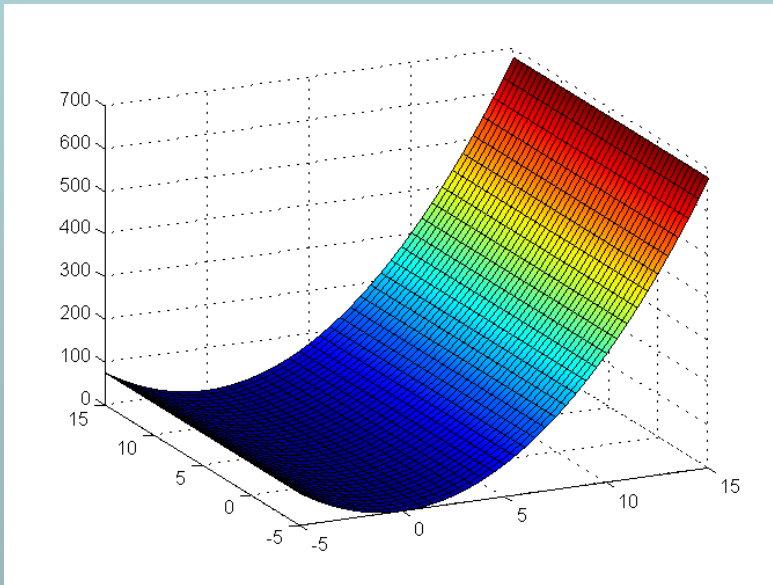
saddle point

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

- 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

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

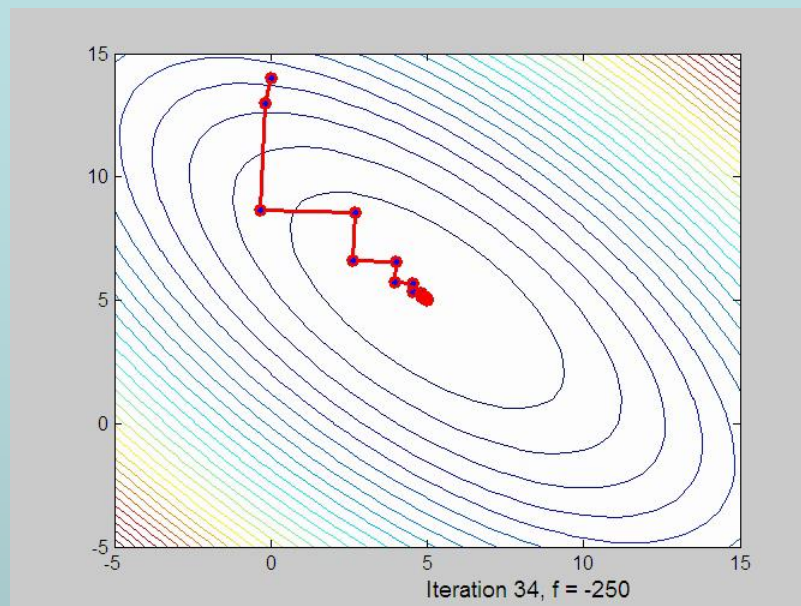
- Step-size α_k is chosen to minimize $f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)$.

For quadratic forms there is a closed form solution:

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

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

Conjugate gradient

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

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

- The resulting search directions are mutually linearly independent.

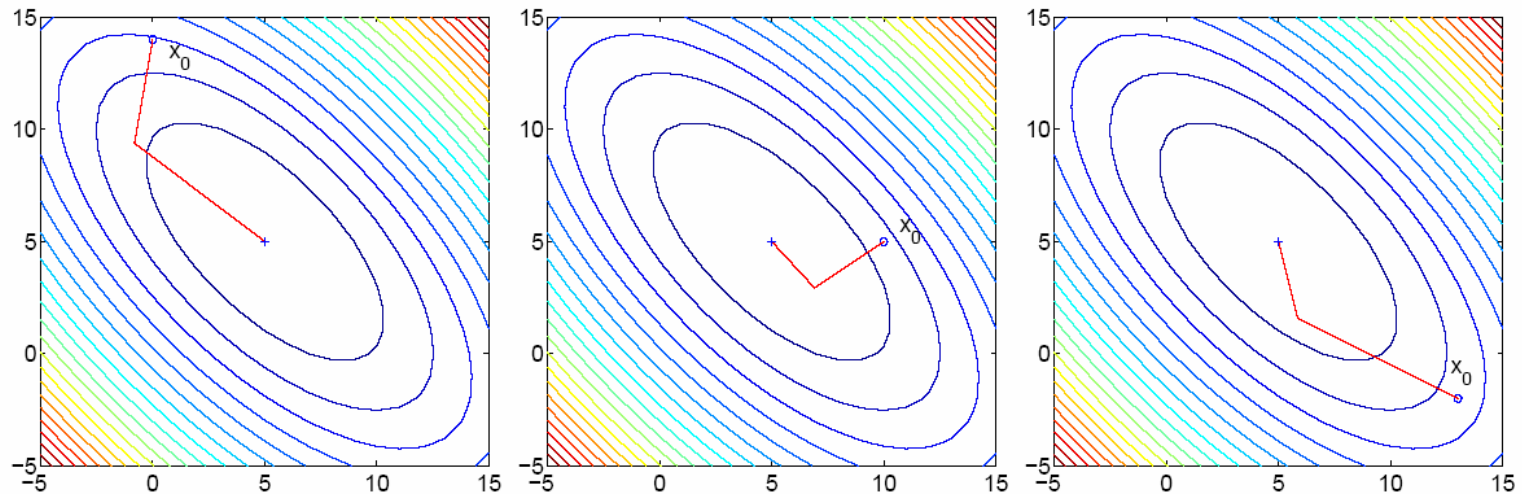
Prove it!

- 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

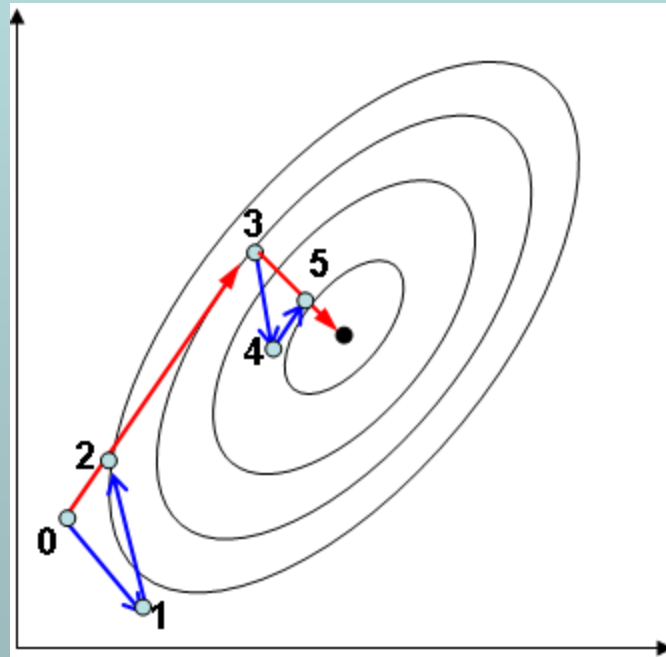
- An N-dimensional quadratic form can be minimized in at most N conjugate descent steps.



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

Powell's Algorithm

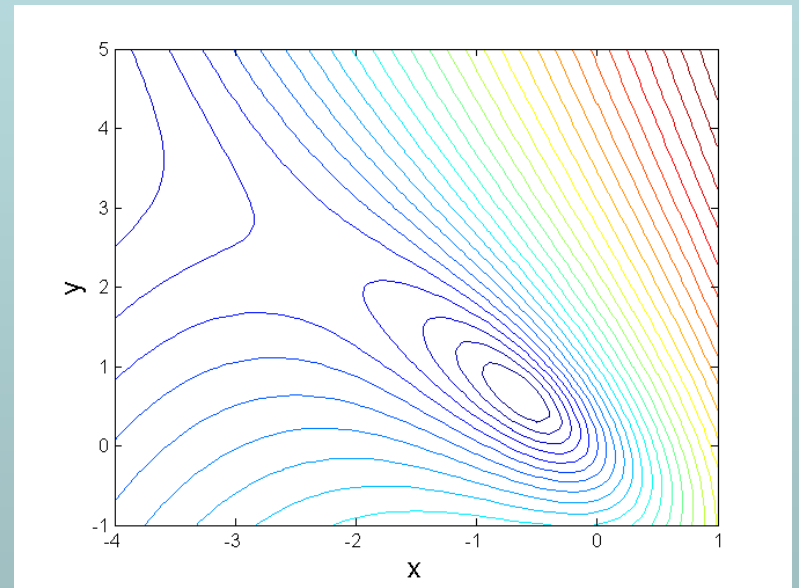
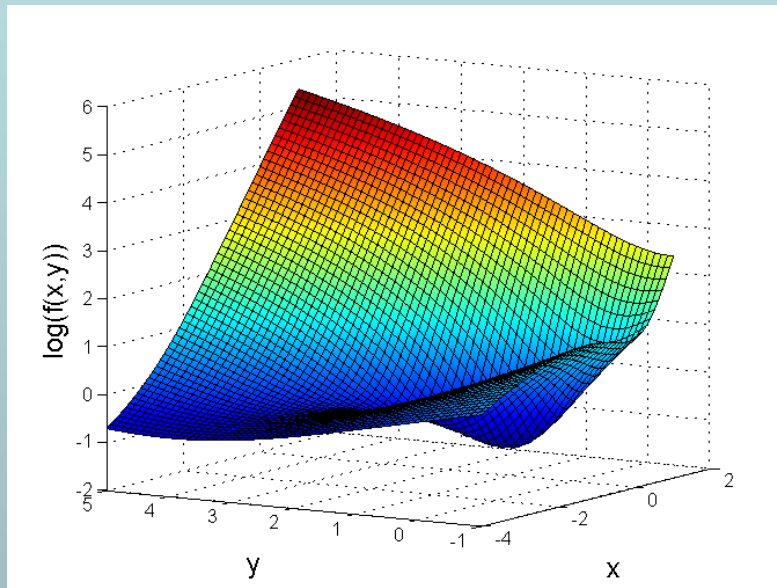
- Conjugate-gradient method that does not require derivatives
- Conjugate directions are generated through a series of line searches



- N-dim quadratic function is minimized with $N(N+1)$ line searches

Optimization for General functions

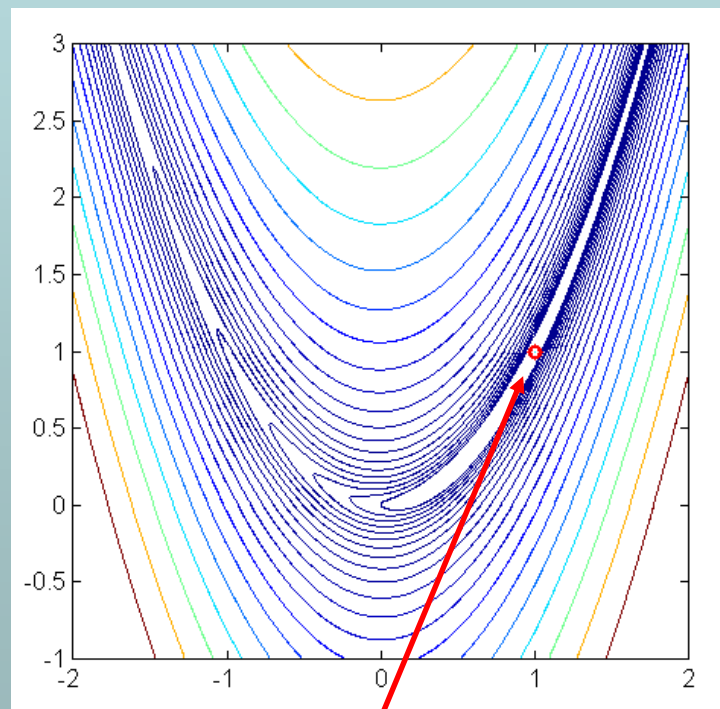
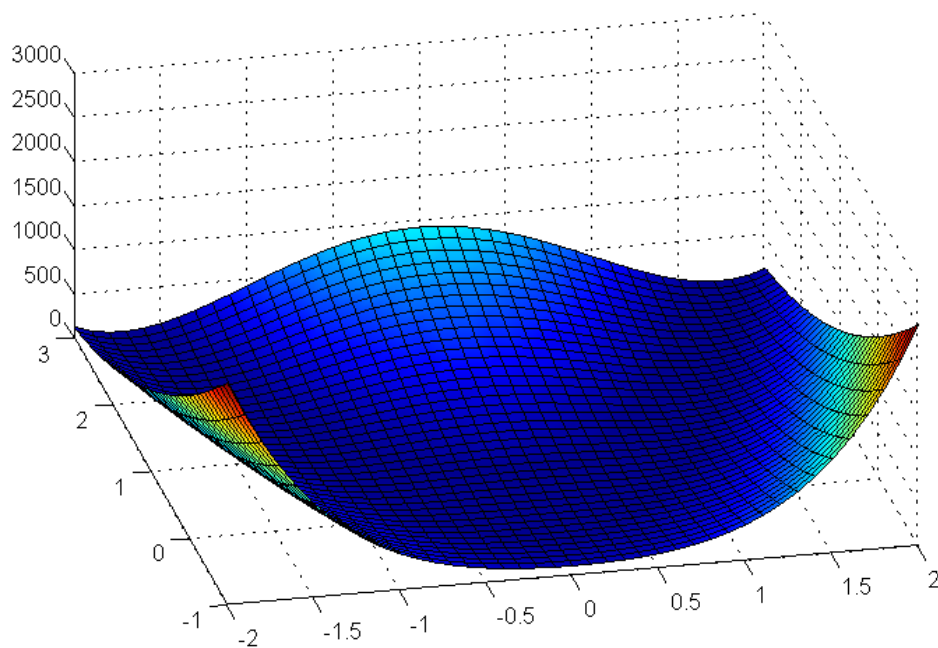
$$f(x, y) = \exp(x)(4x^2 + 2y^2 + 4xy + 2x + 1)$$



Apply methods developed using quadratic Taylor series expansion

Rosenbrock's function

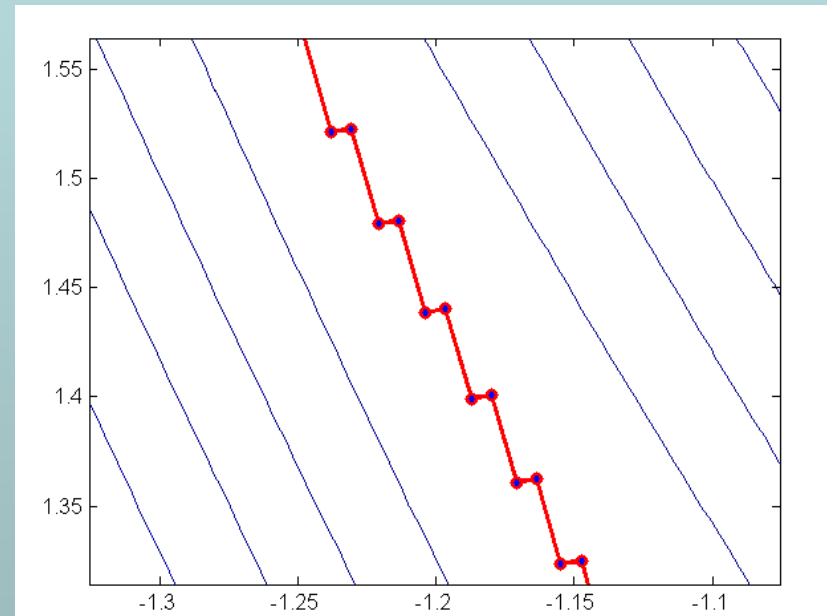
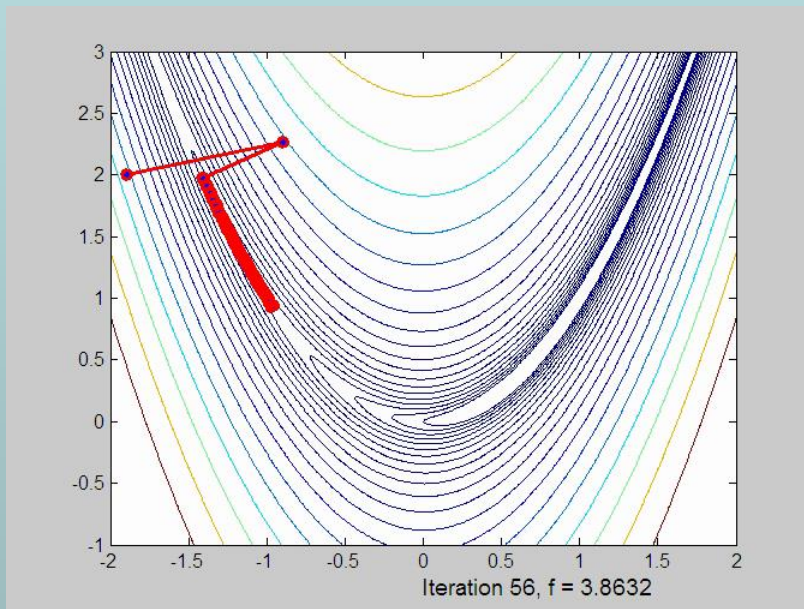
$$f(x, y) = 100(y - x^2)^2 + (1 - x)^2$$



Minimum at [1, 1]

Steepest descent

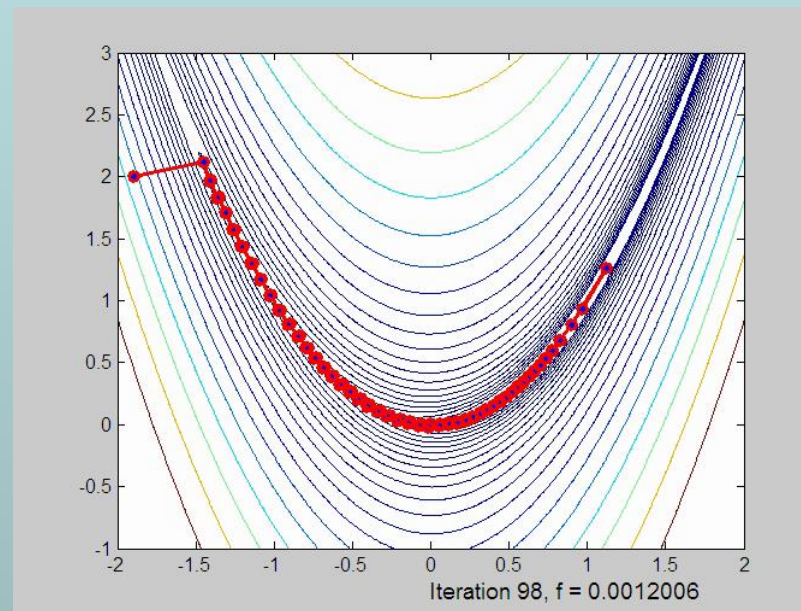
- The 1D line minimization must be performed using one of the earlier methods (usually cubic polynomial interpolation)



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

Conjugate gradient

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



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

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

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

and the Hessian is the symmetric matrix

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

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

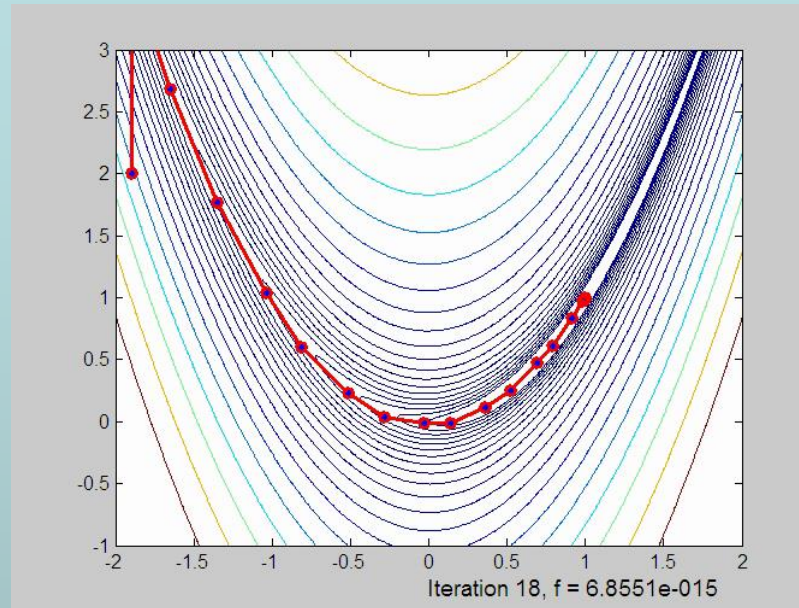
$$\mathbf{x}_{k+1} = \mathbf{x}_k - \mathbf{H}_k^{-1} \mathbf{g}_k$$

- 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



- 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

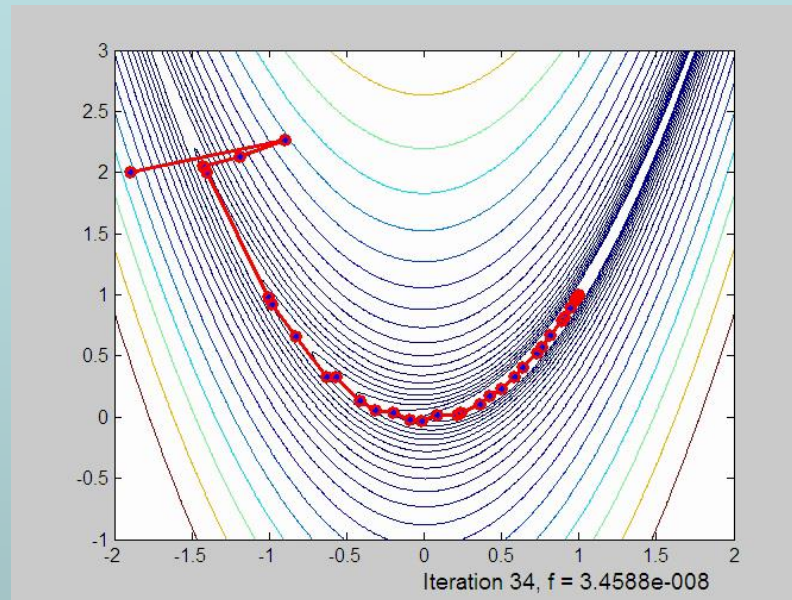
Quasi-Newton methods

- 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 $\mathbf{H}$.

BFGS example



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

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

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

- 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 $M \times N$ 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

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

- Hence

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

Non-linear least squares

- For the Hessian holds

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

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

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

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

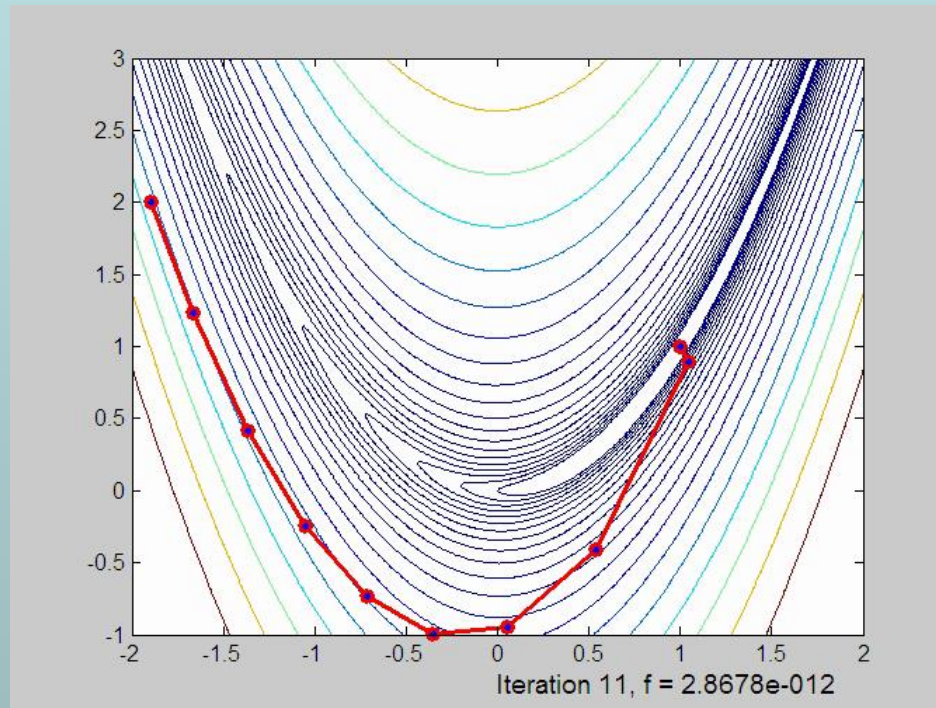
- can be written as a least-squares problem with residual vector

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

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

Gauss-Newton example

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



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

Levenberg-Marquardt Algorithm

- For non-linear least square problems
- Combines Gauss-Newton with Steepest Descent
- Fast convergence even for very “flat” functions
- Descend direction $\delta \mathbf{x}$:

– Newton

$$\mathbf{H}\delta \mathbf{x} = -\mathbf{g}$$

$$\mathbf{J}^T \mathbf{J} \delta \mathbf{x} = -\mathbf{J}^T \mathbf{r}$$

- Steepest Descent

$$\delta \mathbf{x} = -\mathbf{g}$$

$$(\mathbf{J}^T \mathbf{J} + \lambda \mathbf{I}) \delta \mathbf{x} = -\mathbf{J}^T \mathbf{r}$$

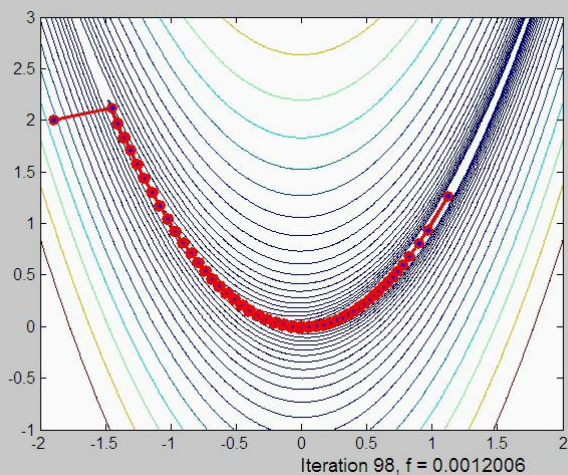
$$(\mathbf{J}^T \mathbf{J} + \lambda \text{diag}(\mathbf{J}^T \mathbf{J})) \delta \mathbf{x} = -\mathbf{J}^T \mathbf{r}$$

Gauss-Newton:

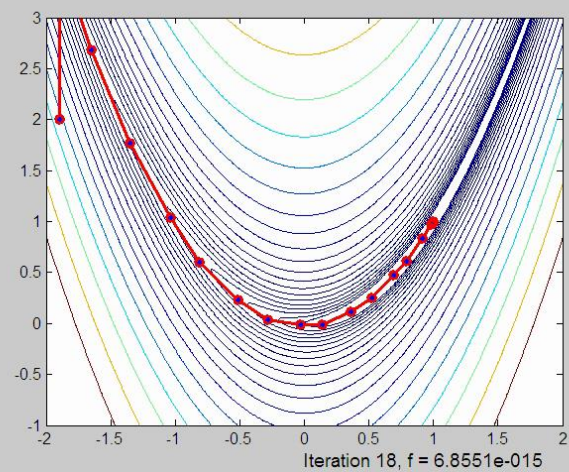
$$\mathbf{g} = 2\mathbf{J}^T \mathbf{r}$$

$$\mathbf{H} = 2\mathbf{J}^T \mathbf{J}$$

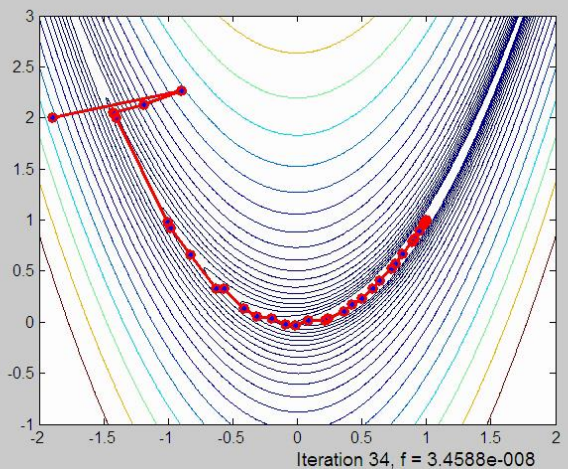
Comparison



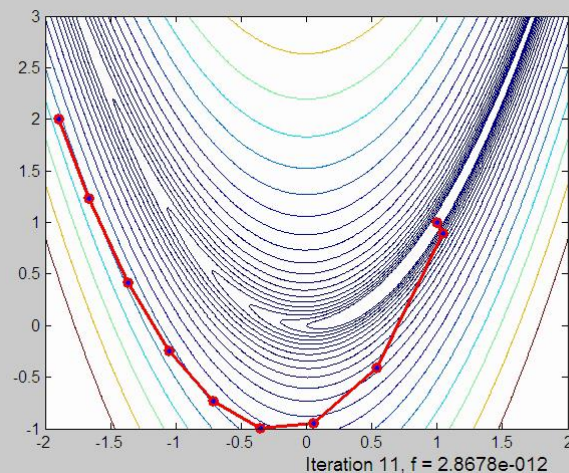
CG



Newton



Quasi-Newton



Gauss-Newton