B1 Optimization

4 Lectures
1 Examples Sheet

Michaelmas 2023
Prof V A Prisacariu

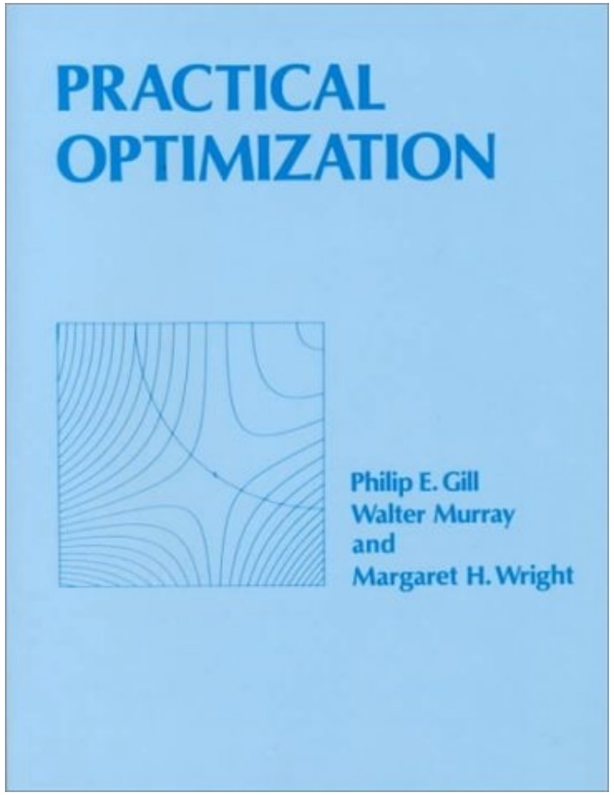
- Lecture 1: Local and global optima, unconstrained univariate and multivariate optimization, stationary points, steepest descent.
- Lecture 2: Newton and Newton like methods – Quasi-Newton, Gauss-Newton; the Nelder-Mead (amoeba) simplex algorithm.
- Lecture 3: Linear programming constrained optimization; the simplex algorithm, interior point methods; integer programming.
- Lecture 4: Convexity, robust cost functions, methods for non-convex functions – grid search, multiple coverings, branch and bound, simulated annealing, evolutionary optimization.
- Course based on previous B1 Optimization, by Prof A Zisserman.

Textbooks

Practical Optimization

Philip E. Gill, Walter Murray, and Margaret H. Wright

Covers unconstrained and constrained optimization. Very clear and comprehensive.



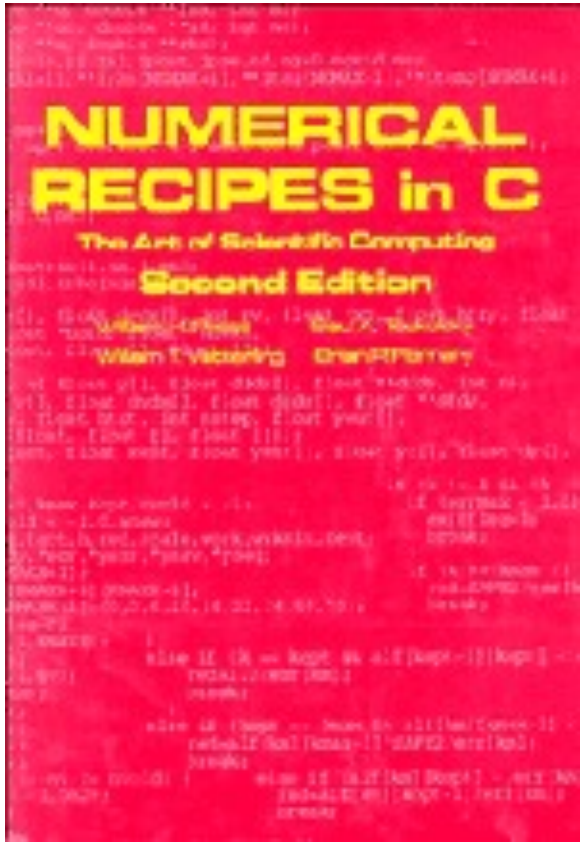
Background reading and web resources

Numerical Recipes in C (or C++) : The Art of Scientific Computing

William H. Press, Brian P. Flannery, Saul A. Teukolsky, William T. Vetterling

CUP 1992/2002

- Good chapter on optimization
- Available on line as pdf



Lecture 1

Topics covered in this lecture:

- Problem formulation;
- Local and global optima;
- Unconstrained univariate optimization;
- Unconstrained multivariate optimization for quadratic functions:
  - Stationary points;
  - Steepest descent.

Introduction

Optimization is used to find the best or optimal solution to a problem.

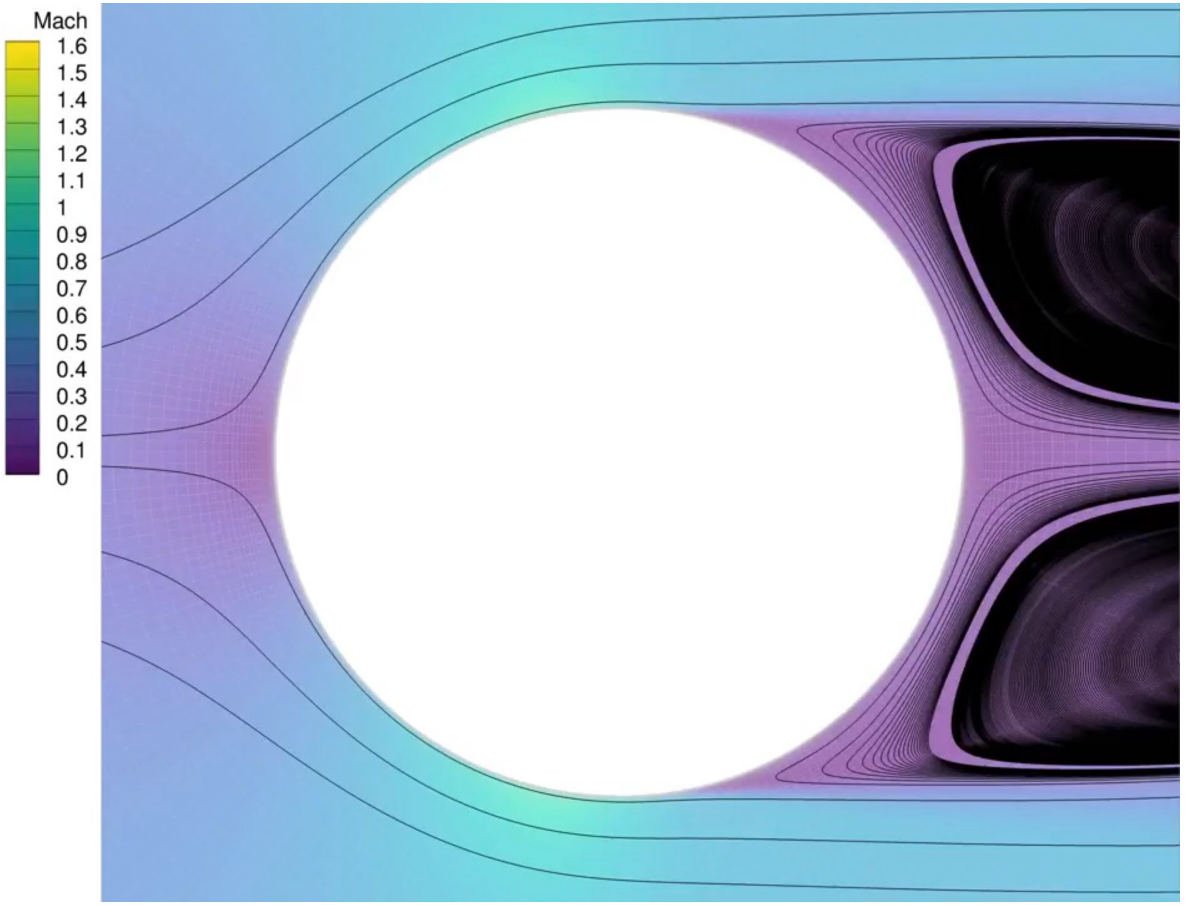
Steps involved in formulating an optimization problem:

- Conversion of the problem into a mathematical model that abstracts all the essential elements;
- Choosing a suitable optimization method for the problem;
- Obtaining the optimum solution.

Example: airfoil/wing aerodynamic shape optimization

Optimization problem:

- constraints: lift, area, chord;
- minimize drag;
- vary shape.



Aircraft Design via Numerical Optimization *Are We There Yet?* Martins *et al,* 2017

Introduction: Problem specification

Suppose we have a cost function (or objective function):

$$f(\mathbf{x}): \mathbb{R}^n \to \mathbb{R}$$

Our aim is find the value of the parameters $\mathbf{x}$ that minimize the function:

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

subject to the following constraints:

- equality: $c_i(\mathbf{x}) = 0,\ i = 1, \dots, m_e$.
- inequality: $c_i(\mathbf{x}) \geq 0,\ i = m_e + 1, \dots, m$.

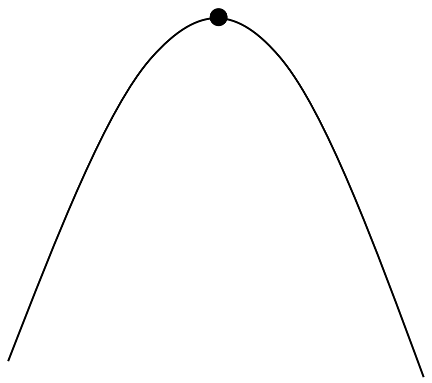
We will start by focussing on unconstrained problems.

Recall: One dimensional functions

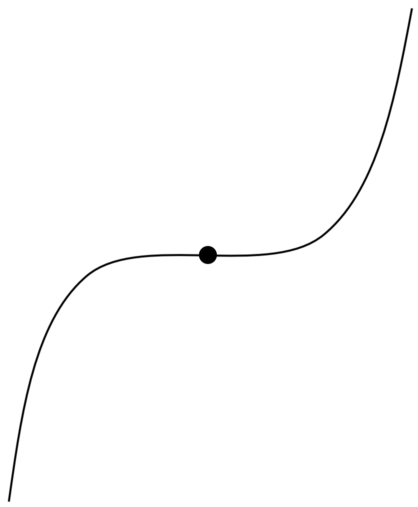
A differentiable function has a stationary point when the derivative is zero:

$$\frac{df}{dx} = 0$$

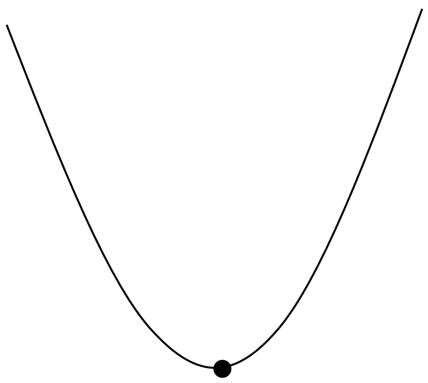
The second derivative gives the type of stationary point.



$f''(x) < 0$
maximum



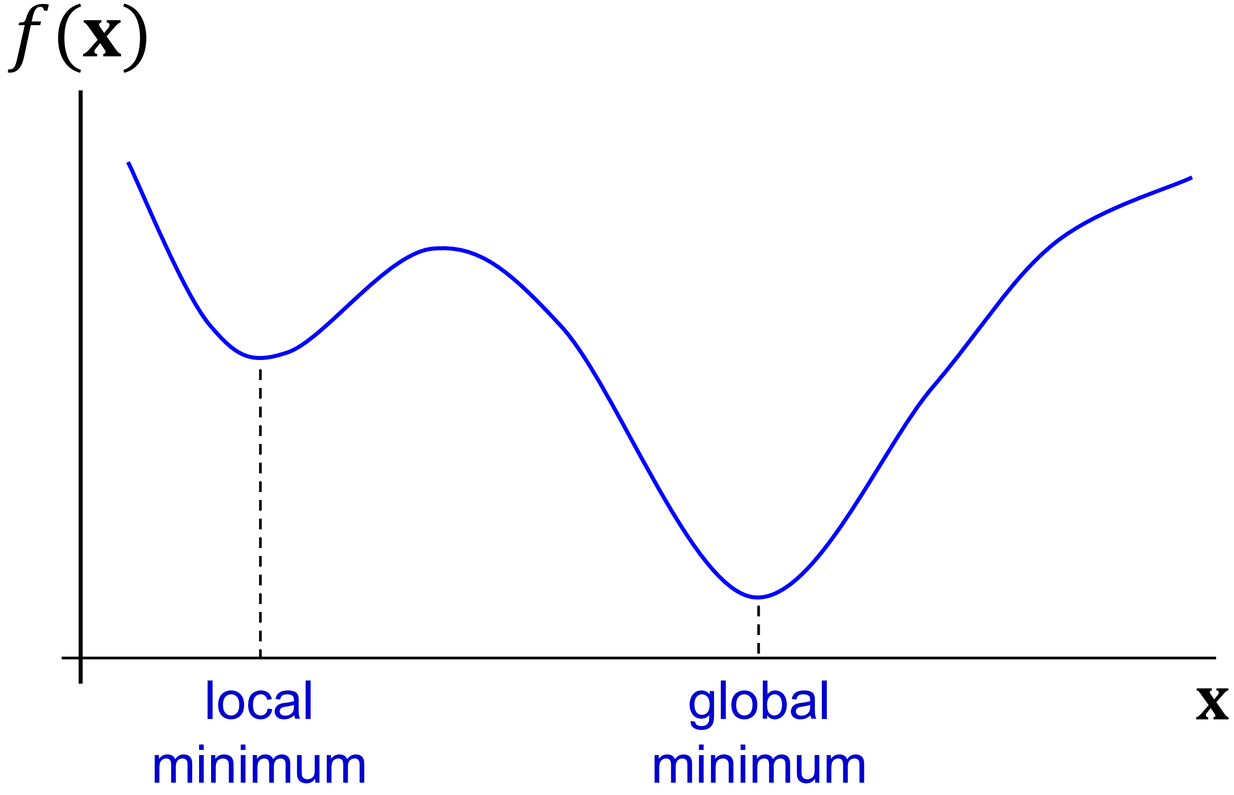
$f''(x) = 0$
inflection



$f''(x) > 0$
minimum

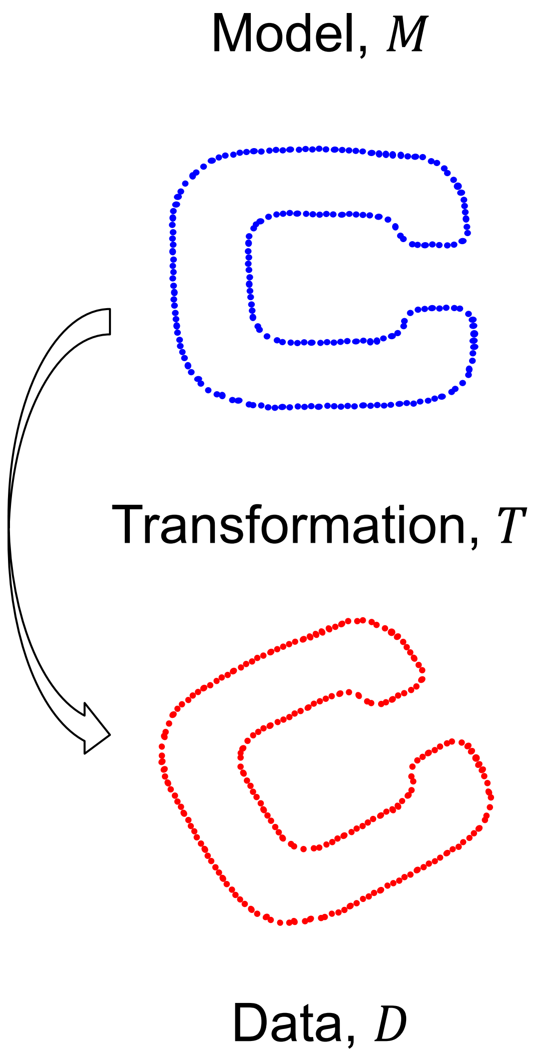
Unconstrained optimization

$$\min_{\mathbf{x}} f(\mathbf{x})$$



- down-hill search (gradient descent) algorithms can find local minima;
- which of the minima is found depends on the starting point;
- such minima often occur in real applications.

Example: template matching in 2D images



Input:

Two point sets $M = \{\mathbf{M}_i\}$ and $D = \{\mathbf{D}_j\}$.

Task:

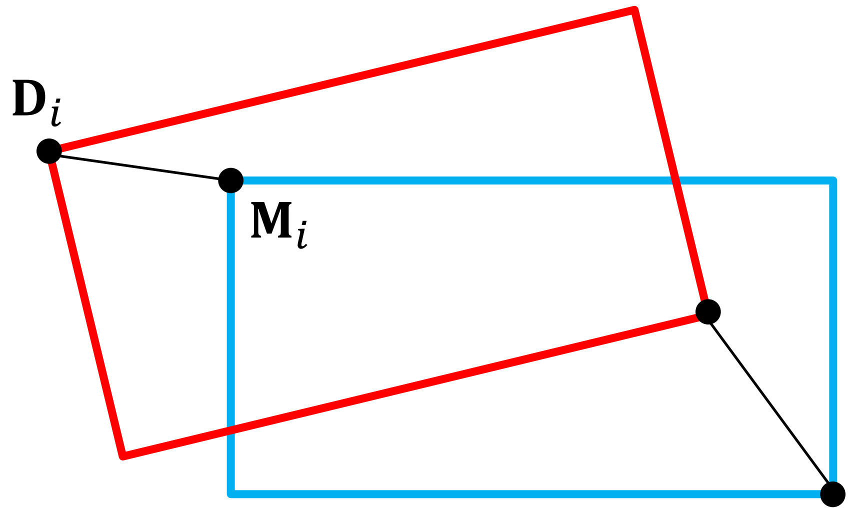
Determine the transformation T that minimizes the error between D and the transformed M.

Motivation:

Robot picking up “C”

Cost function (correspondences known)

2D points $(x, y)^T$, Model $\mathbf{M}_i$, Data $\mathbf{D}_i$



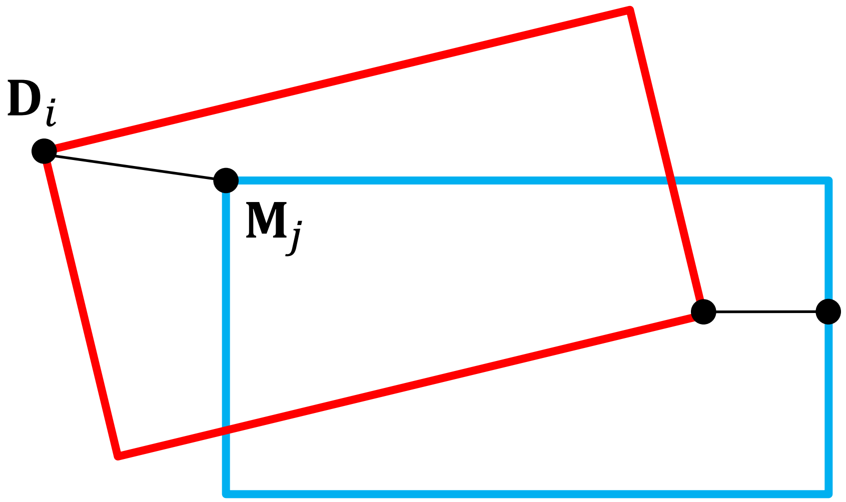
$$f(\theta, t_x, t_y) = \sum_i |\mathbf{R}(\theta)\mathbf{M}_i + t - \mathbf{D}_i|^2$$

Transformation parameters:

- Rotation angle θ
- Translation $\mathbf{t} = (t_x, t_y,)^T$

Cost function (correspondences unknown)

2D points $(x, y)^T$, Model $\mathbf{M}_j$, Data $\mathbf{D}_i$



$$f(\theta, t_x, t_y) = \underbrace{\sum_i}_{\text{for each data point}} \underbrace{\min_j \left|\mathbf{R}(\theta)\mathbf{M}_j + t - \mathbf{D}_i\right|^2}_{\text{find closest model point}}$$

Transformation parameters:

- Rotation angle θ
- Translation $\mathbf{t} = (t_x, t_y,)^T$

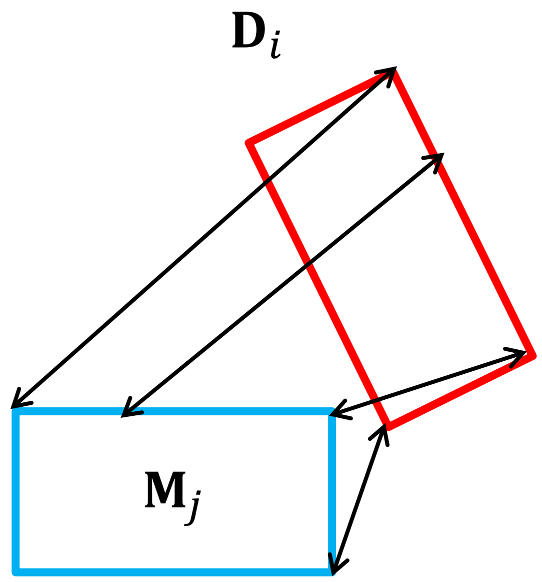
Cost function (matches unknown)

$$f(\theta, t_x, t_y) = \underbrace{\sum_i}_{\text{for each data point}} \underbrace{\min_j \left|\mathbf{R}(\theta)\mathbf{M}_j + t - \mathbf{D}_i\right|^2}_{\text{find closest model point}}$$

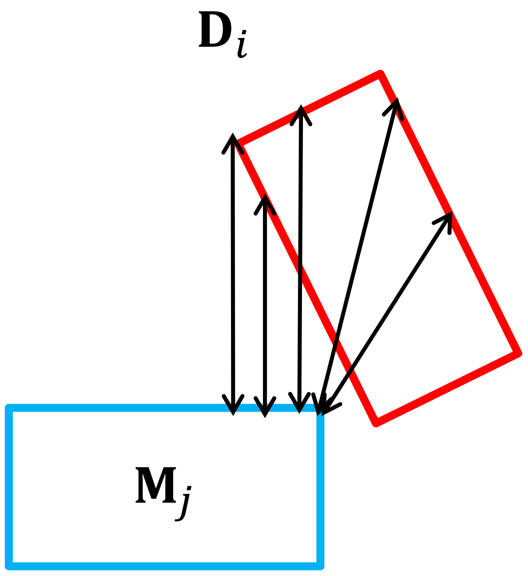
As distances become smaller we get the correct correspondences and model pulled onto the data

Model point: $\mathbf{M}_j = (x_j, y_j)^T$

Transformation parameters:
- Rotation angle θ
- Translation $\mathbf{t} = (t_x, t_y,)^T$

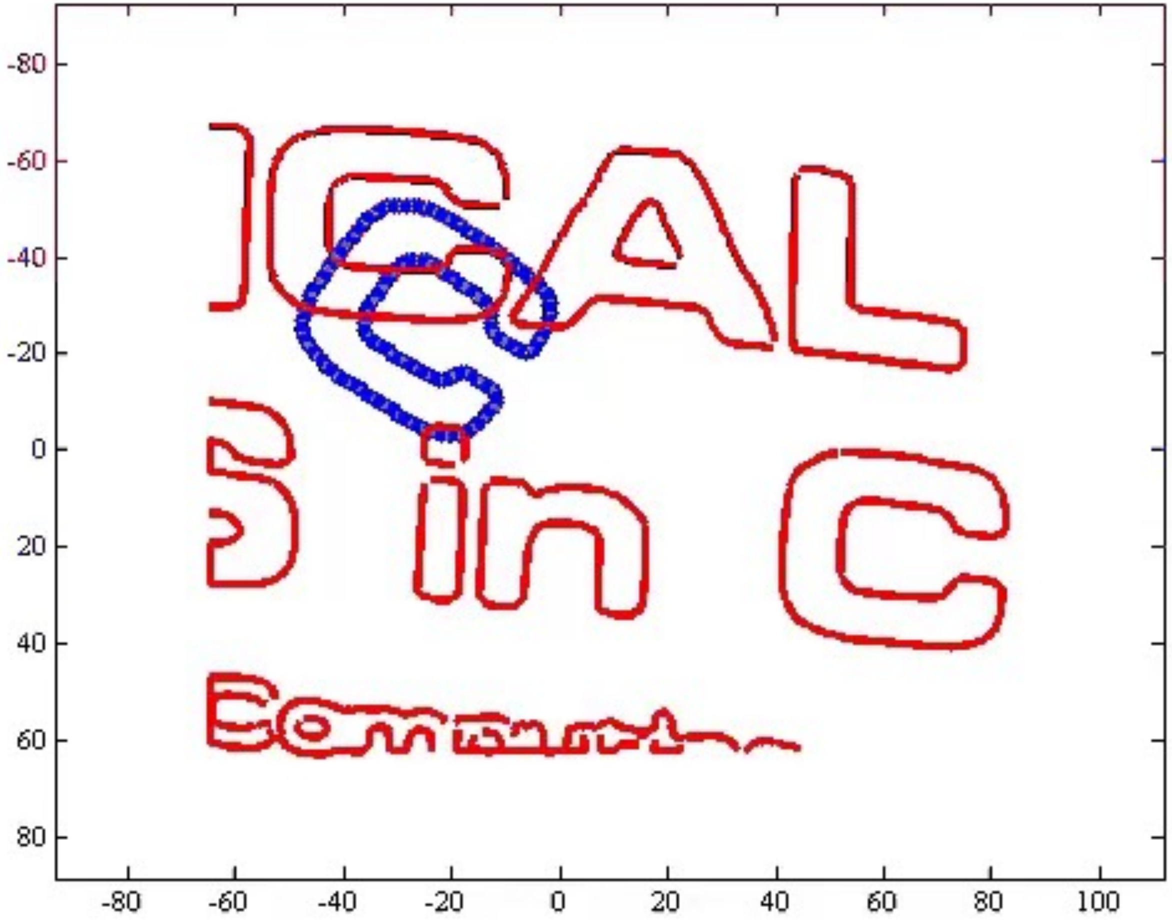


correct correspondences



closest point correspondences

Performance

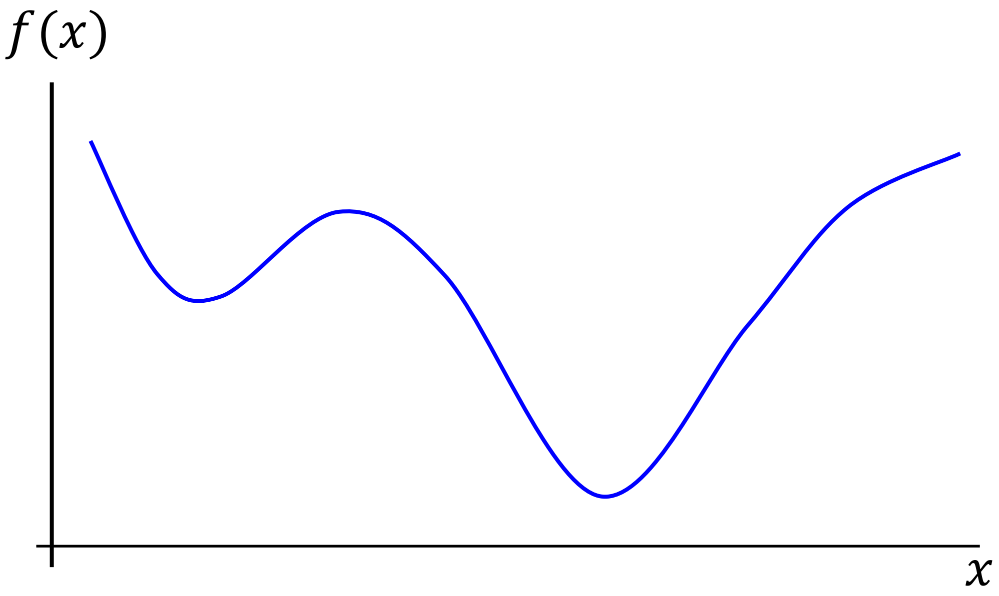


Unconstrained univariate optimization

For the moment, assume we can start close to the global minimum

$$\min_{x} f(x)$$

function of one variable



We will look at three basic methods to determine the minimum:

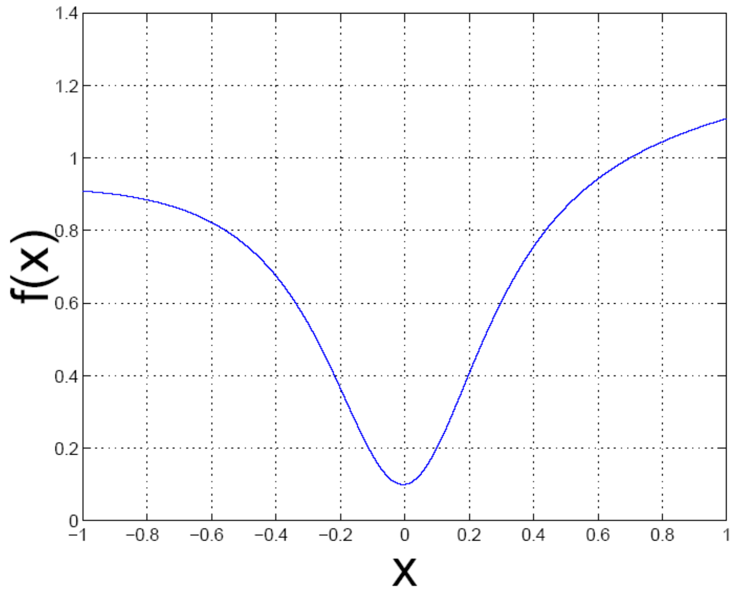
1. Gradient descent;
2. Polynomial interpolation;
3. Newton's method.

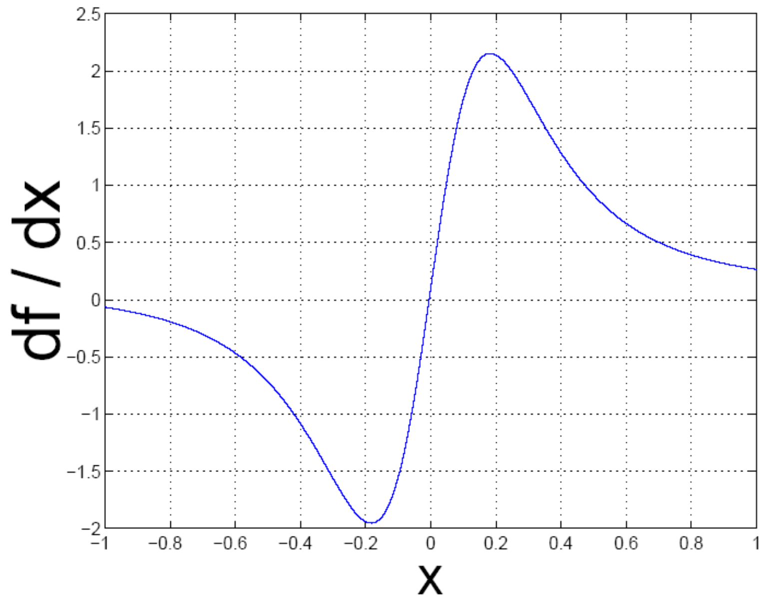
These introduce the ideas that will be applied in the multivariate case.

A typical 1D function

As an example, consider the function:

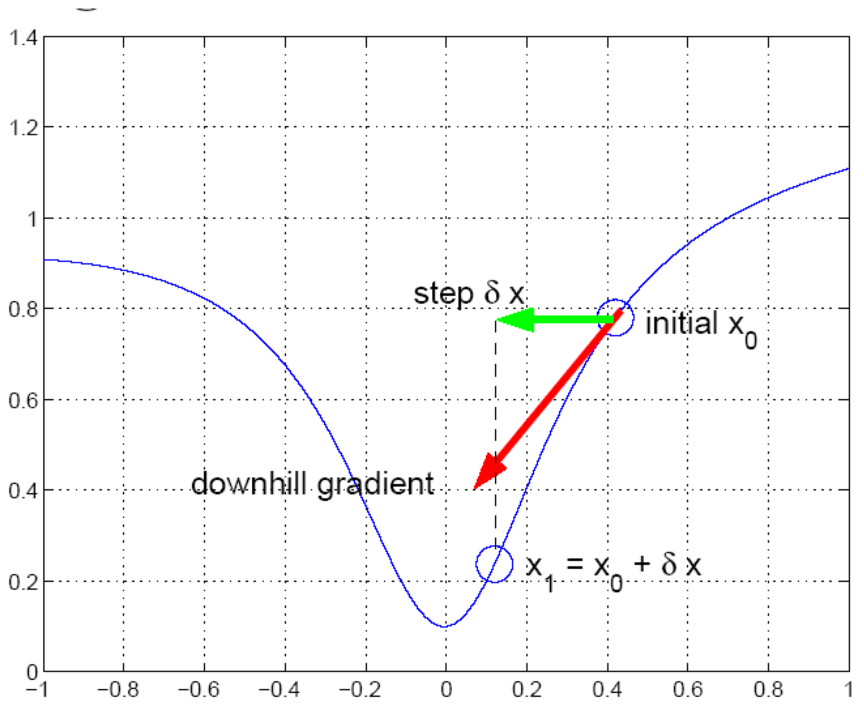
$$f(x) = 0.1 + 0.1x + x^2/(0.1 + x^2)$$





1. Gradient descent

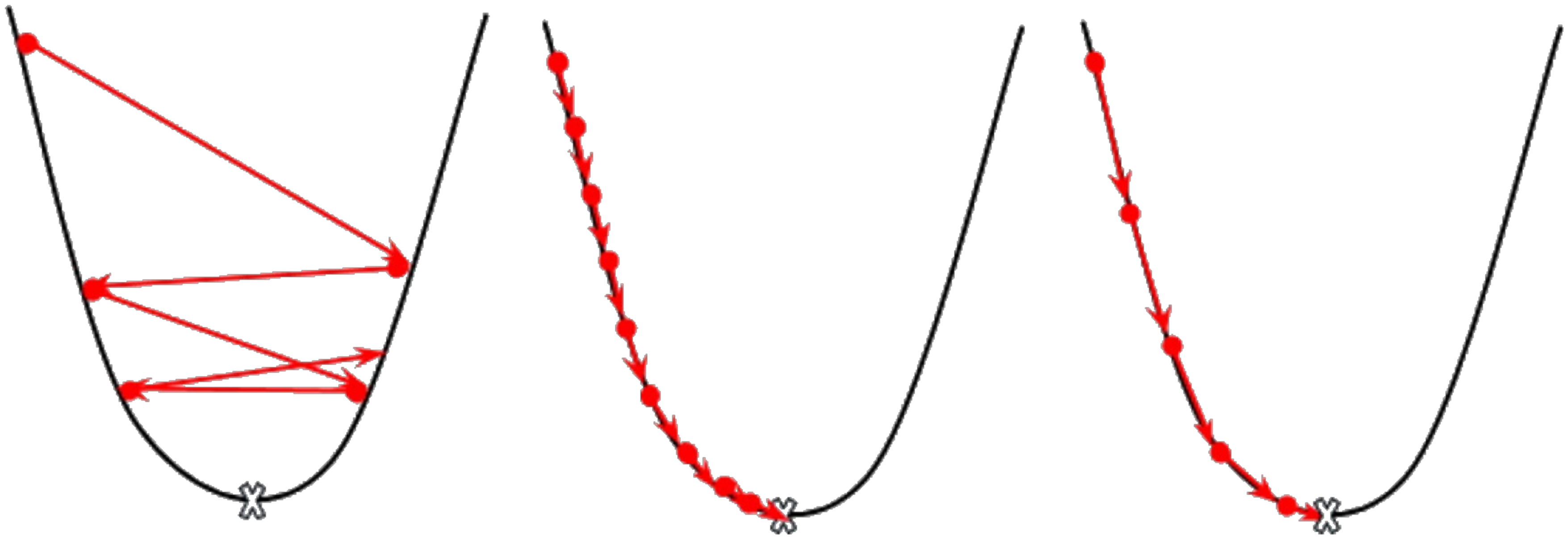
Given a starting location, x_0, examine $\frac{df}{dx}$ and move in the *downhill* direction to generate a new estimate $x_1 = x_0 + \delta x$.



$$\delta x = -\alpha \frac{df}{dx}$$

How to determine the step size δx ?

Setting alpha ..



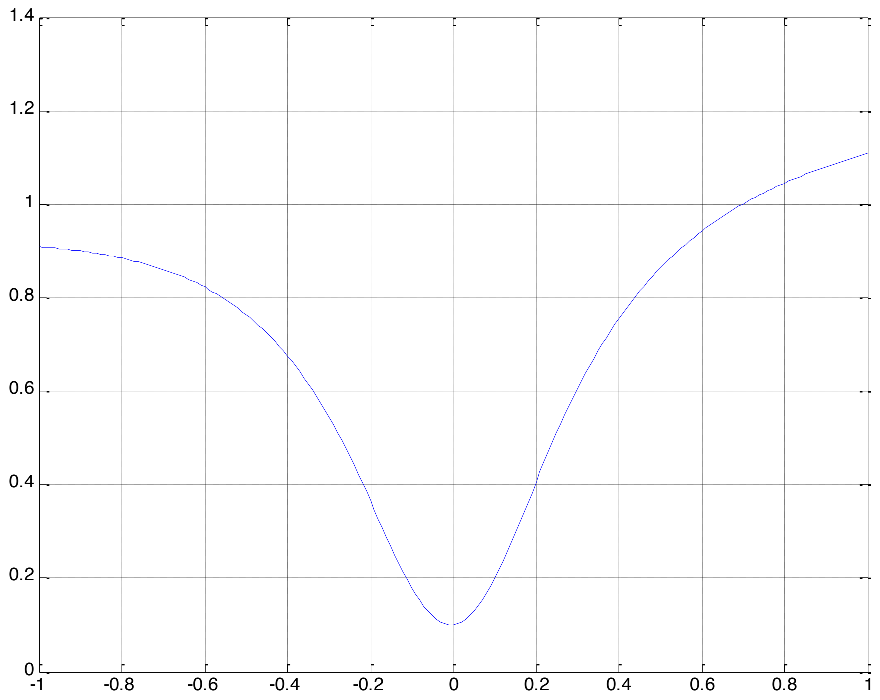
If the step size is too large, gradient descent may not converge (left)
If it is too small, convergence will be slow (middle).
A good step size leads to fast convergence (right)

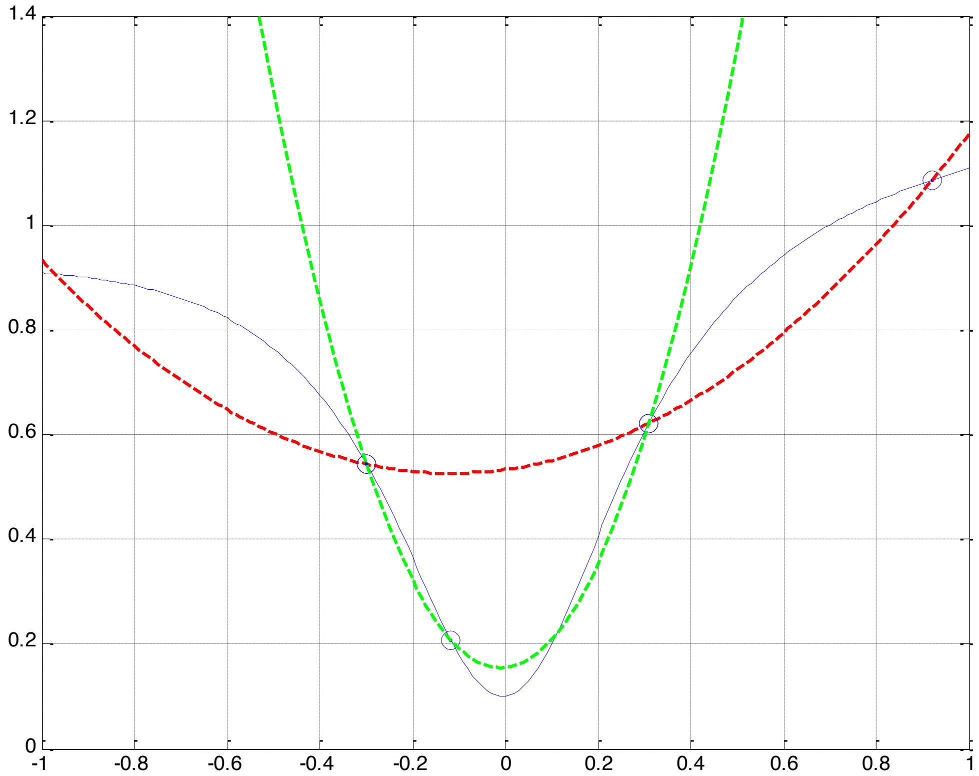
Slide credit: Bert De Brabandere

2. Polynomial interpolation (trust region method)

Approximate $f(x)$ with a simpler function which reasonably approximates the function in a neighbourhood around the current estimate x. This neighbourhood is the trust region.

- 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 for the polynomial.
- Throw away the worst point and repeat the process.





Quadratic interpolation using 3 points, 2 iterations

Other methods to interpolate a quadratic ?

- e.g. 2 points and one gradient

3. Newton's method

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

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

$$f(x+\delta x) = f(x) + \delta x f'(x) + \frac{\delta x^2}{2} f''(x) + \text{ h.o.t.}$$

- Find the δx (the variable here) such that $x + \delta x$ is a stationary point of f:

$$\frac{d}{d\delta x}\left(f(x) + \delta x f'(x) + \frac{\delta x^2}{2} f''(x)\right) = f'(x) + \delta x f''(x) = 0$$

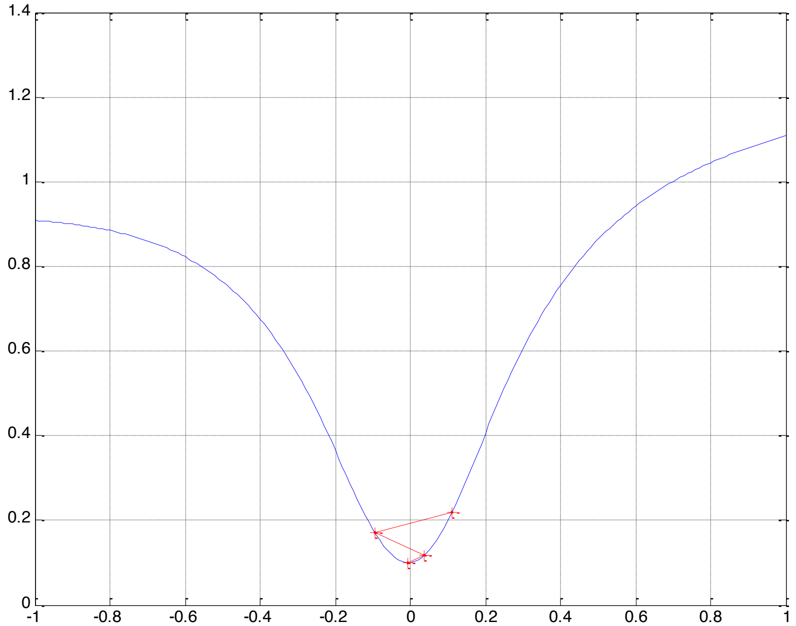
- and rearranging:

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

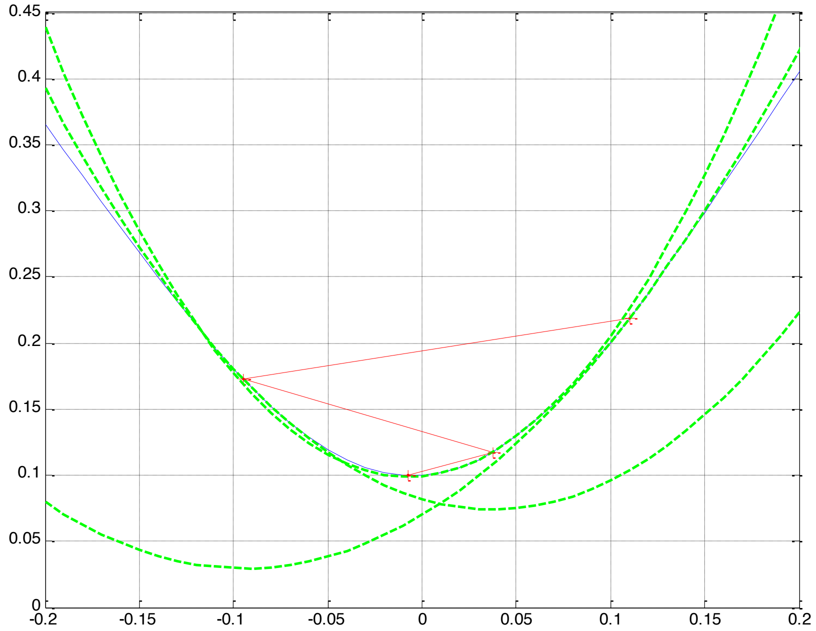
- Update for x:

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

Newton iterations

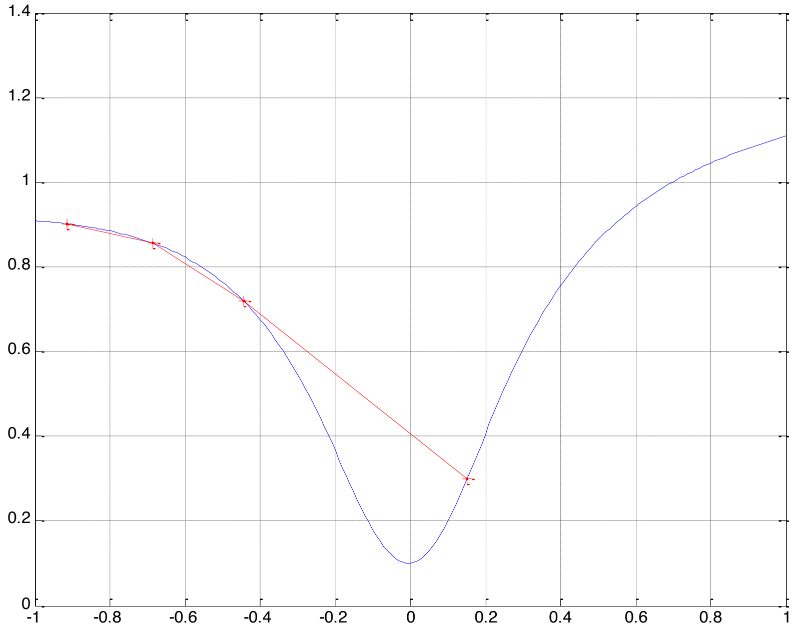


Quadratic approximations

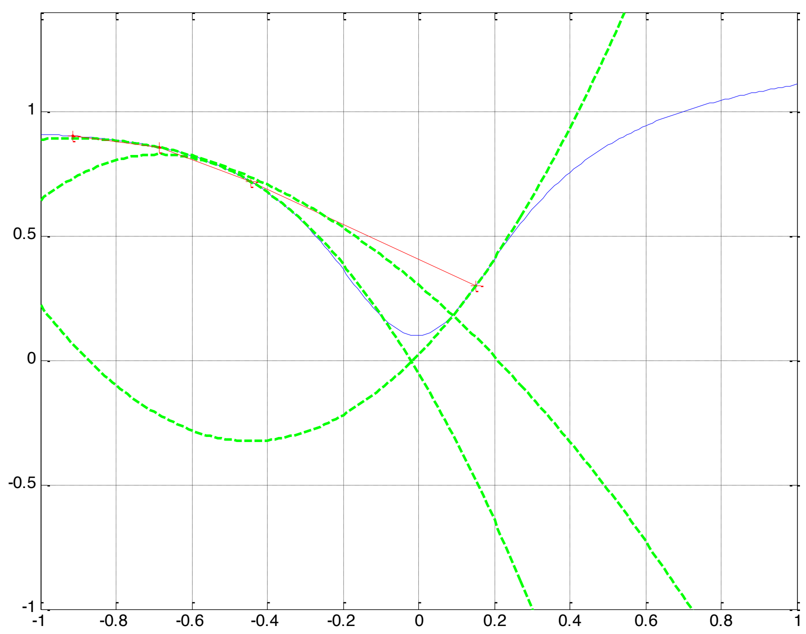


- avoids the need to bracket the root;
- quadratic convergence (decimal accuracy doubles at every iteration).

Newton iterations



Quadratic approximations

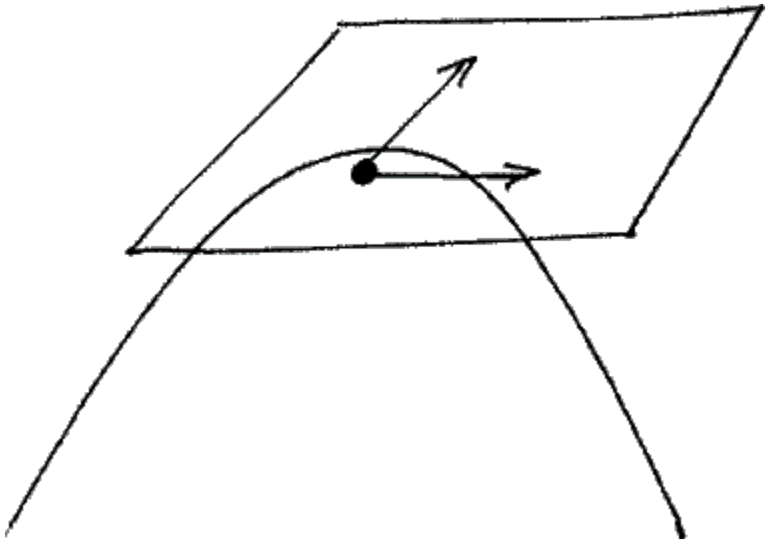


- global convergence of Newton's method is poor;
- often fails if the starting point is too far from the minimum;
- in practice, must be used with a globalization strategy which reduces the step length until function decrease is assured.

Stationary Points for Multidimensional functions

$f(x): \mathbb{R}^n \to \mathbb{R}$ has a stationary point with the gradient

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots, \frac{\partial f}{\partial x_n}\right)^T = 0$$



$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right)^T = 0$$

Extension to N dimensions

- How big can N be?: problem sizes can vary from a handful of parameters to millions.

- In the following we will first examine the properties of stationary points in N dimensions and then move onto optimization algorithms to find the stationary point (minimum).

- We will consider examples for $N = 2$, so that cost function surfaces can be visualized.

Taylor expansion in 2D

A function can be approximated locally by its Taylor series expansion about a point x_0.

$$f(x_0 + x) \approx f(x_0) + \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right)\begin{pmatrix} x \\ y \end{pmatrix} + \frac{1}{2}(x, y)\begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}\begin{pmatrix} x \\ y \end{pmatrix} + \text{h.o.t.}$$

- This is a generalization of the 1D Taylor series:

$$f(x + \delta x) = f(x) + \delta x f'(x) + \frac{\delta x^2}{2} f''(x) + \text{ h.o.t.}$$

- The expansion to second order is a quadratic function in $\mathbf{x}$.

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

Taylor expansion in ND

A function may be approximated locally by its Taylor expansion about a point x_0

$$f(x_0 + x) \approx f(x_0) + \nabla f^T \mathbf{x} + \frac{1}{2}\mathbf{x}^T \mathrm{H}\mathbf{x} + \text{h.o.t.}$$

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

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

and the Hessian $\mathrm{H}(\mathbf{x})$ of $f(\mathbf{x})$ is the symmetric matrix

$$\begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_N} & \cdots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix}$$

Properties of Quadratic functions

Taylor expansion:

$$f(\mathbf{x}_0 + \mathbf{x}) = f(\mathbf{x}_0) + \mathbf{g}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathrm{H}\mathbf{x}$$

Expand about a stationary point $\mathbf{x}_0 = \mathbf{x}^*$ in direction $\mathbf{p}$:

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

since the stationary point $\mathbf{g} = \nabla f|_{\mathbf{x}^*} = 0$.

At a stationary point the behavior is determined by H.

Properties of Quadratic functions

H is a symmetric matrix, so it has orthogonal eigenvectors

$$\mathrm{H}\mathbf{u}_\mathrm{i} = \lambda_\mathrm{i}\mathbf{u}_\mathrm{i} \text{ choose } |\mathbf{u}_i| = 1$$

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

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.

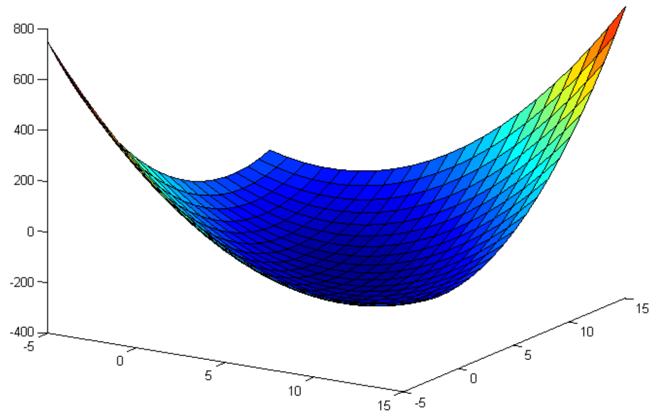
Examples of Quadratic functions

Case 1: both eigenvalues positive

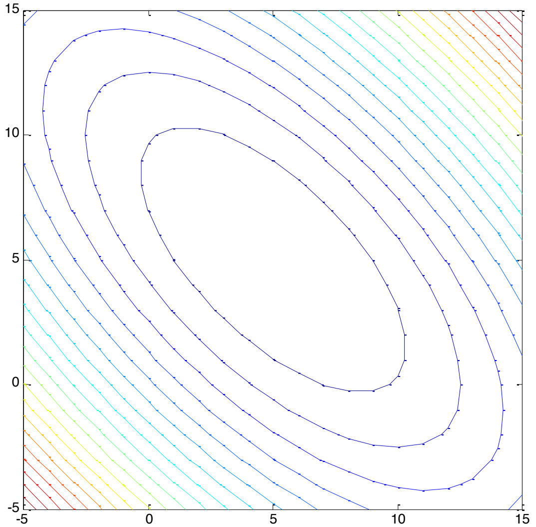
$$f(\mathbf{x}) = a + \mathbf{g}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathrm{H}\mathbf{x}$$

with

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



minimum



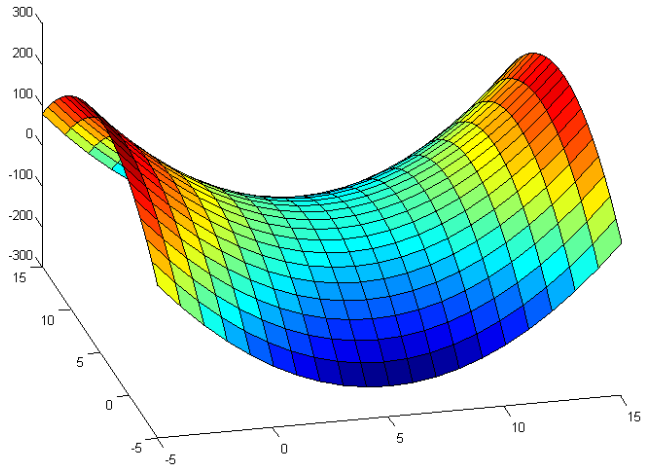
Examples of Quadratic functions

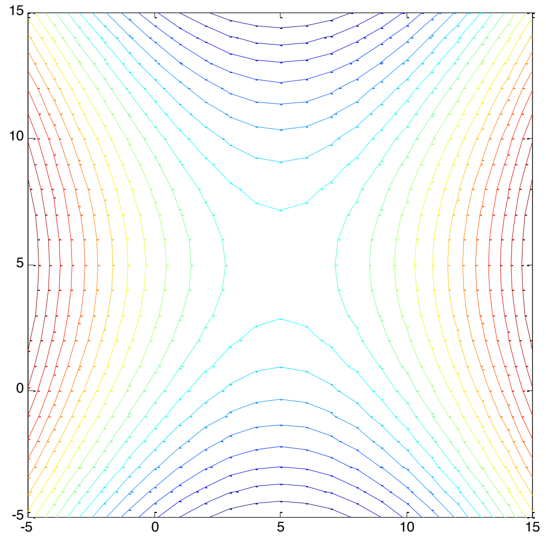
Case 2: eigenvalues have different signs

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

with

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





saddle surface: extremum but not a minimum

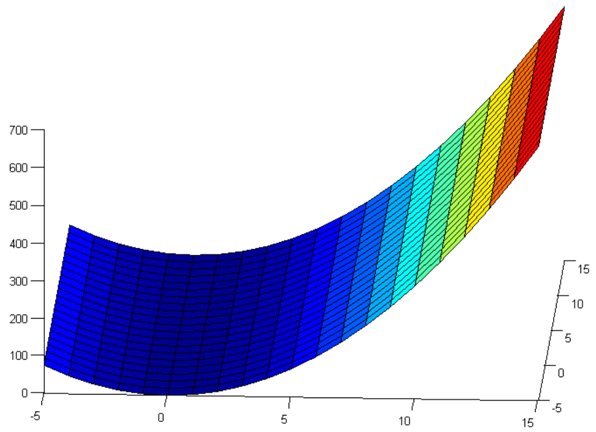
Examples of Quadratic functions

Case 3: one eigenvalue zero.

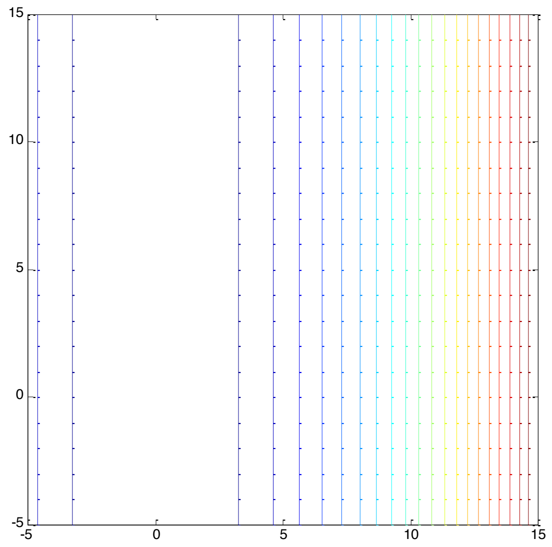
$$f(\mathbf{x}) = a + \mathbf{g}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T \mathrm{H}\mathbf{x}$$

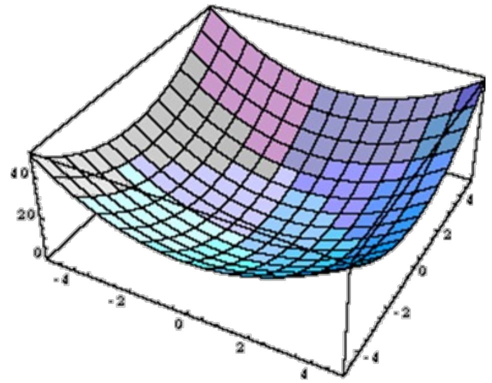
with

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

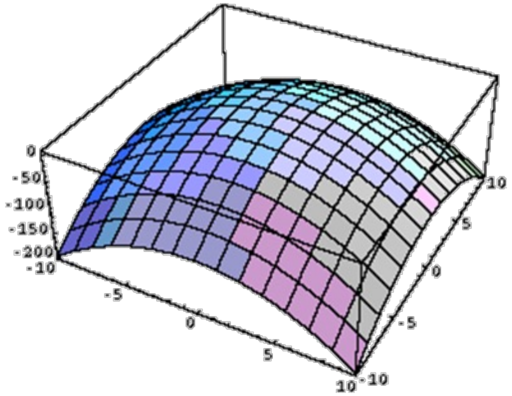


parabolic cylinder

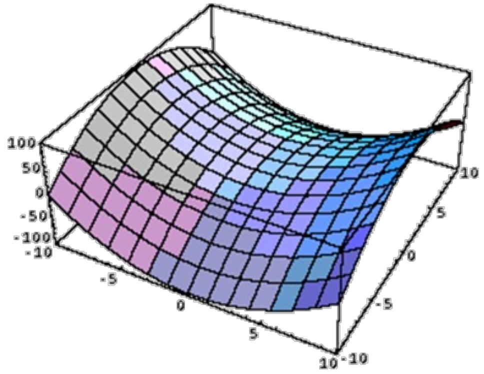




Hessian positive definite.
Convex function.
Minimum point.



Hessian negative definite.
Concave function.
Maximum point.

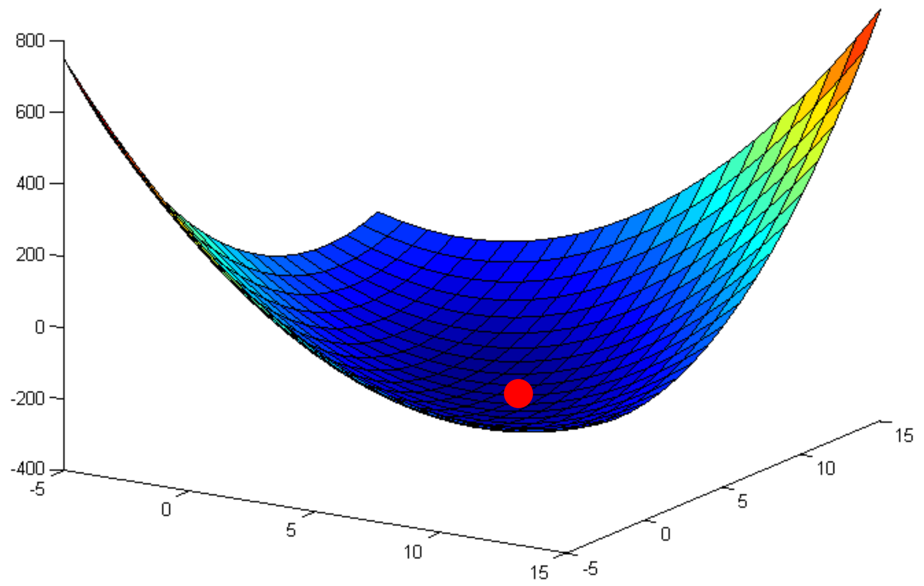


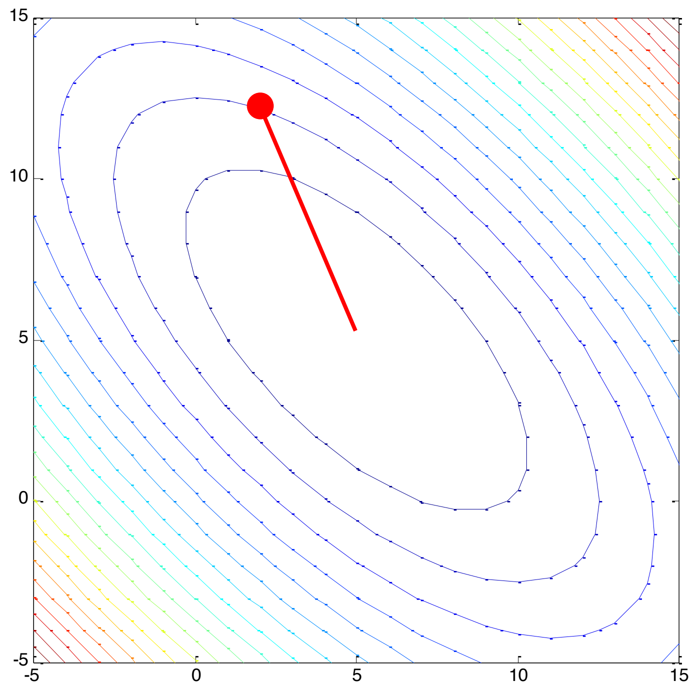
Hessian mixed.
Surface has negative point.
Saddle point.

More in Lecture 4

Optimization in N dimensions

- Reduce optimization in N dimensions to a series of (1D) line minimizations.
- Use methods developed in 1D (e.g. polynomial interpolation).





Optimization in N dimensions

Start at $\mathbf{x}_0$ then repeat:

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 (termination criteria), e.g. $\nabla f \approx 0$.

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

Steepest descent

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

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n \mathbf{p}_n$$

The steepest decent method choses $\mathbf{p}_n$ to be parallel to the negative gradient:

$$\mathbf{p}_n = -\nabla f(\boldsymbol{x}_\mathrm{n})$$

The step-size α_n is chosen to minimize $f(\mathbf{x}_n + \alpha_n \mathbf{p}_n)$.

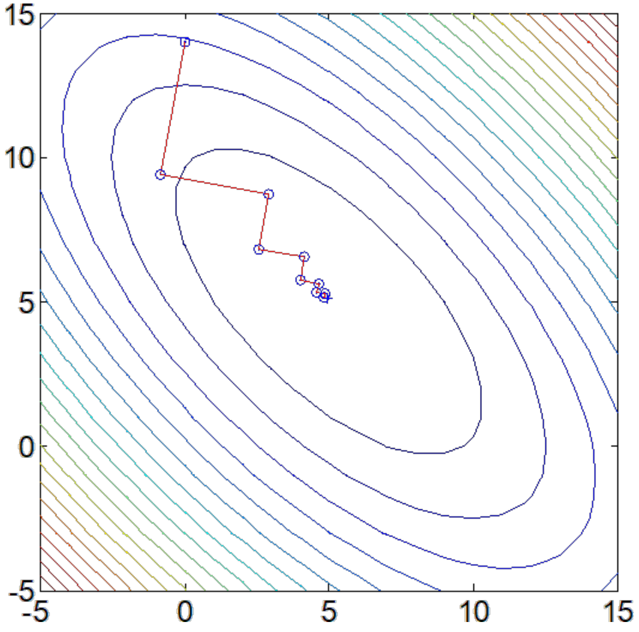
For quadratic forms there is a closed form solution:

$$\alpha_n = -\frac{\mathbf{p}_n^T \mathbf{p}_n}{\mathbf{p}_n^T \mathrm{H} \mathbf{p}_n}$$

[exercise – minimize $f(\mathbf{x}_n + \alpha_n \mathbf{p}_n)$ wrt α]

Steepest descent example

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



Steepest descent at $\mathbf{x}_0 = [0, 14]$

After each line minimization, the new gradient is always orthogonal to the previous direction step

- True for any line minimization.
- Can be proven by examining the derivation for α_n.

Consequently, the iterations may zig-zag down the valley in a very inefficient matter.

Conjugate gradients

The method of conjugate gradients chooses successive descent directions $\mathbf{p}_n$ such that it is guaranteed to reach the minimum in a finite number of steps.

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

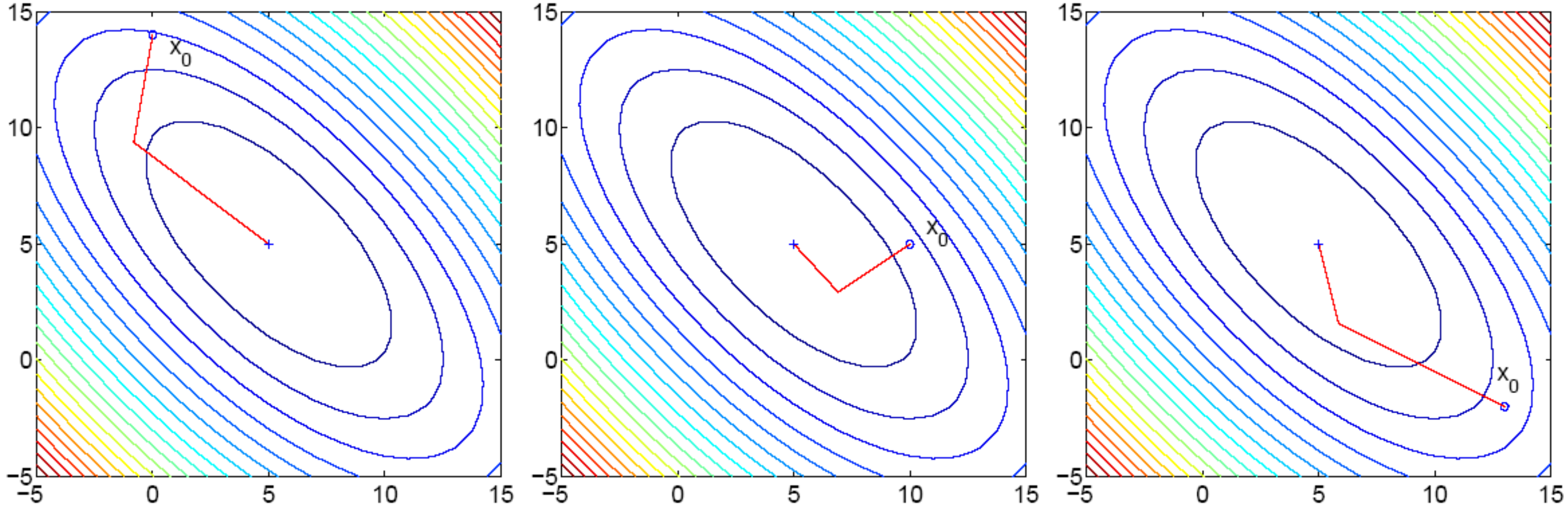
$$\mathbf{p}_n^T H \mathbf{p}_n = 0$$

- The resulting search directions are mutually linearly independent.

- Remarkably, $\mathbf{p}_n$ can be chose using only knowledge of $\mathbf{p}_{n-1}, \nabla f(\mathbf{x}_{n-1})$ and $\nabla f(\mathbf{x}_n)$ (see Numerical Recipes), e.g.

$$\mathbf{p}_n = \nabla \mathrm{f_n} + \left(\frac{\nabla f_n^T \nabla f_n}{\nabla f_{n-1}^T \nabla f_{n-1}} \right) \mathbf{p}_{n-1}$$

Conjugate gradients



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

- In figure: 3 different starting steps, minimum is reached in exactly 2 steps.

What is next?

- Move from functions that are exactly quadratic to general functions that are represented locally by a quadratic
- Newton's method (that uses 2^{nd} derivatives) and Newton-like methods for general functions