3B1B Optimization

4 Lectures
1 Examples Sheet

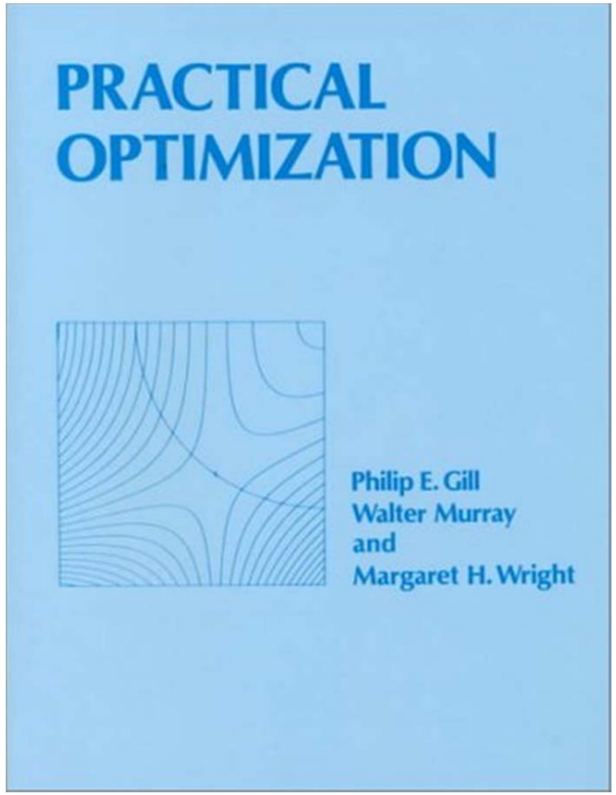
Michaelmas Term 2016
Prof. A. Zisserman

- 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.

Textbooks

- **Practical Optimization**

Philip E. Gill, Walter Murray, and Margaret H. Wright, Academic Press, 1981



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

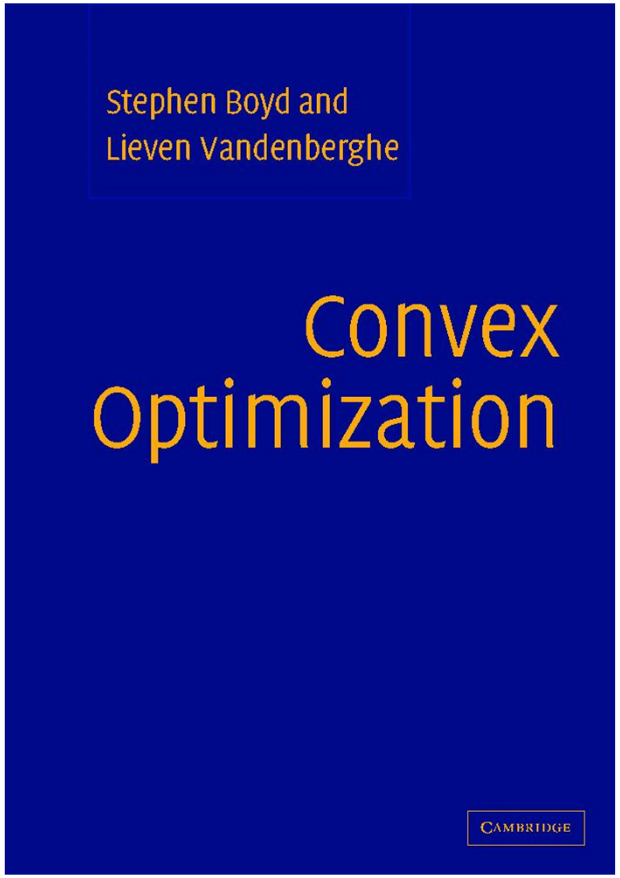
CUP 1992/2002

  - Good chapter on optimization
  - Available on line at

    http://www.nrbook.com/a/bookcpdf.php

Background reading and web resources

- **Convex Optimization**
- **Stephen Boyd and Lieven Vandenberghe**
  CUP 2004
  - Available on line at
    http://www.stanford.edu/~boyd/cvxbook/



- **Further reading, web resources, and the lecture notes are on http://www.robots.ox.ac.uk/~az/lectures/b1**

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.

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 this function

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

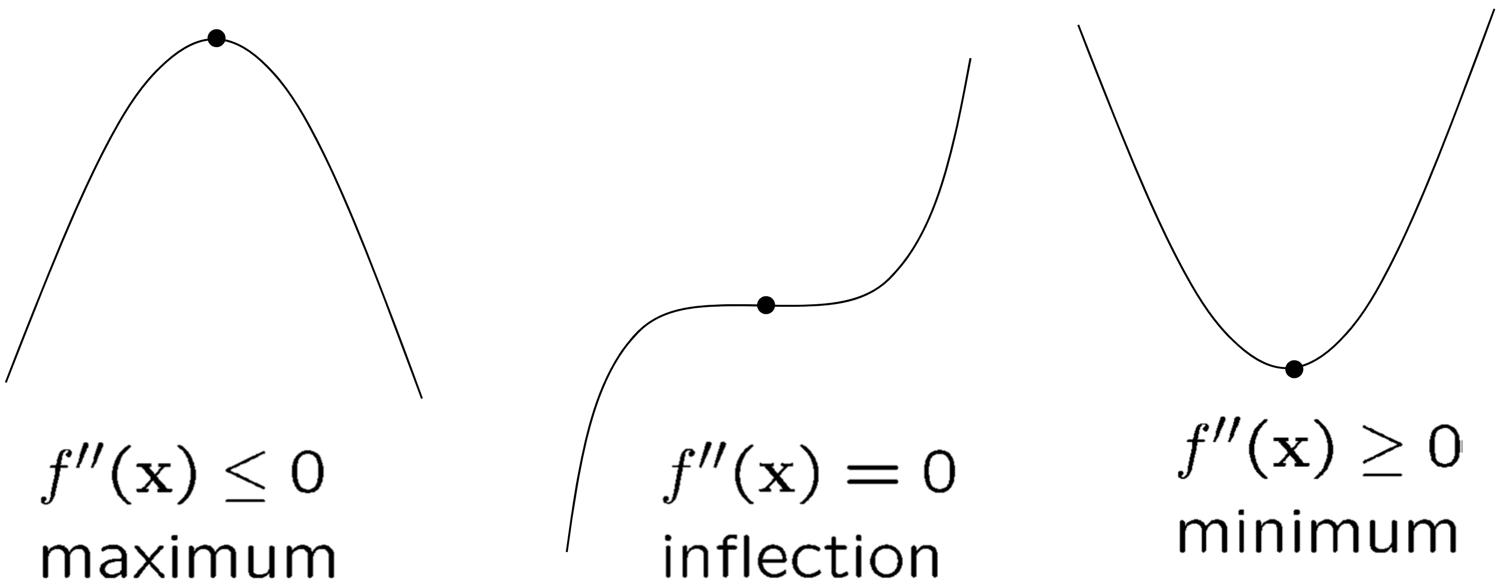
subject to the following constraints:

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

We will start by focussing on unconstrained problems

Recall: One dimensional functions

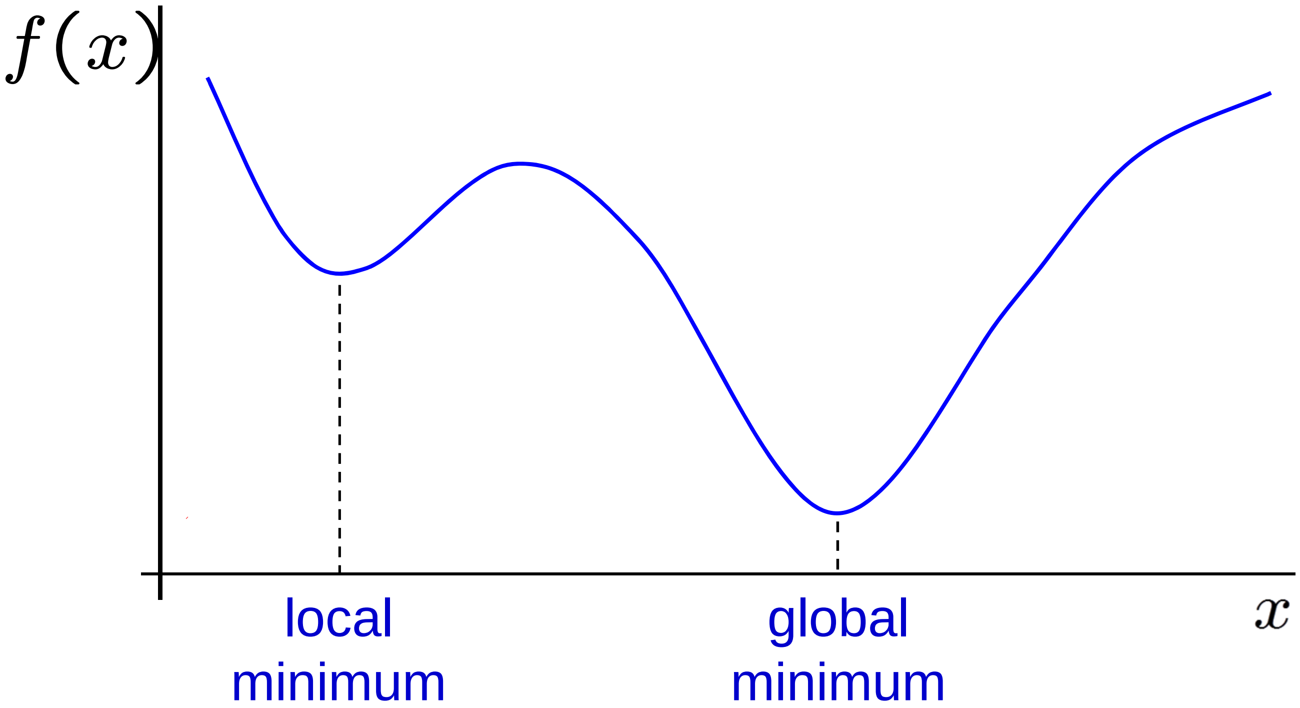
- A differentiable function has a stationary point when the derivative is zero: $df/dx = 0$.
- The second derivative gives the type of stationary point



Unconstrained optimization

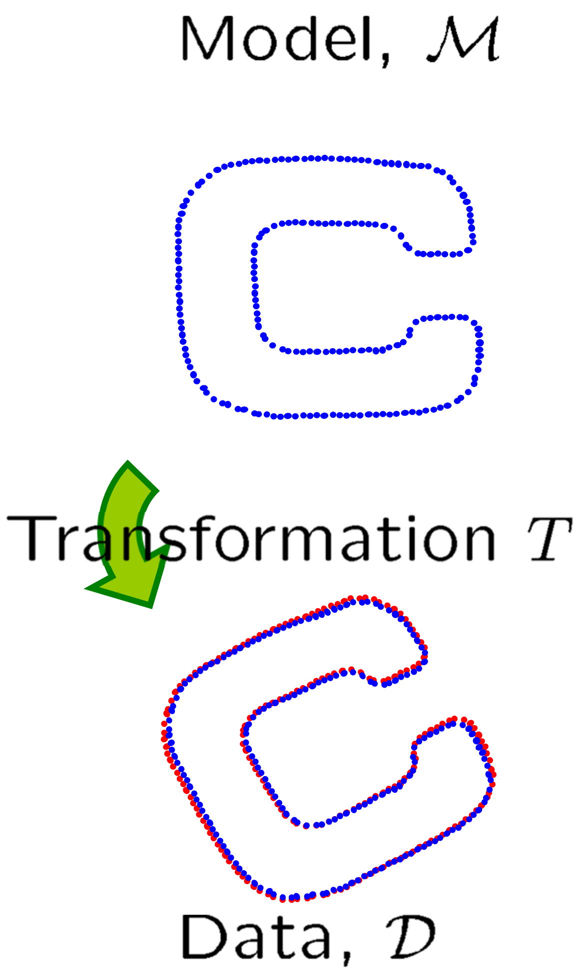
function of one variable

$$\min_x f(x)$$



- 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:

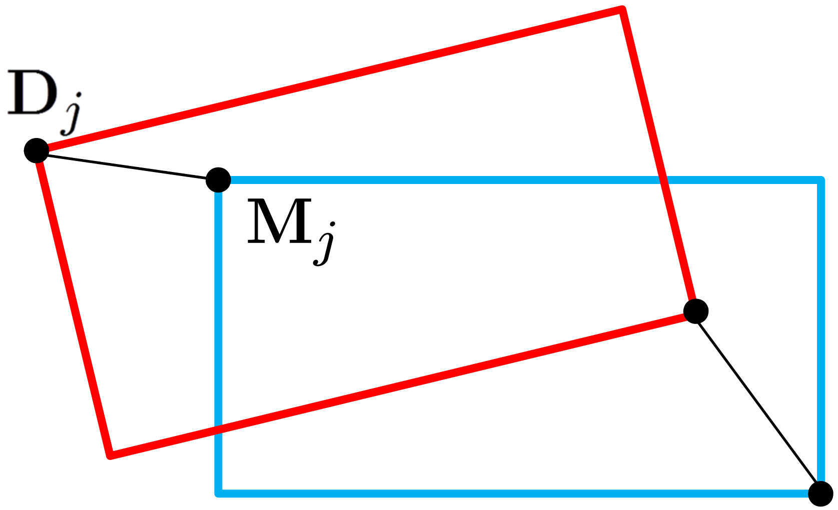
Two point sets
$\mathcal{M} = \{\mathbf{M}_i\}$ and $\mathcal{D} = \{\mathbf{D}_j\}$

Task:

Determine the transformation T that minimizes the error between $\mathcal{D}$ and the transformed $\mathcal{M}$

Cost function

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



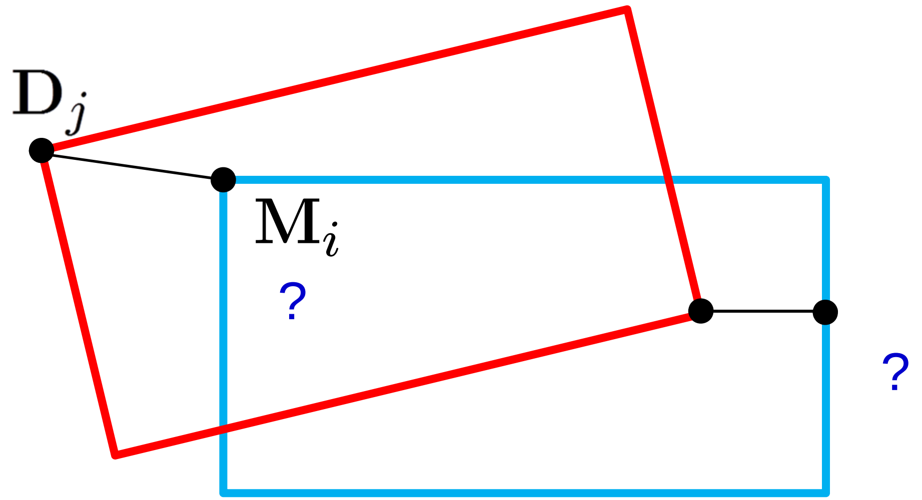
$$f(\theta, t_x, t_y) = \sum_j \|\mathbf{R}(\theta)\mathbf{M}_j + \mathbf{t} - \mathbf{D}_j\|^2$$

Transformation parameters:

- rotation angle θ
- translation $\mathbf{t} = (t_x, t_y)^\top$

Cost function

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



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

Transformation parameters:

- rotation angle θ
- translation $\mathbf{t} = (t_x, t_y)^\top$

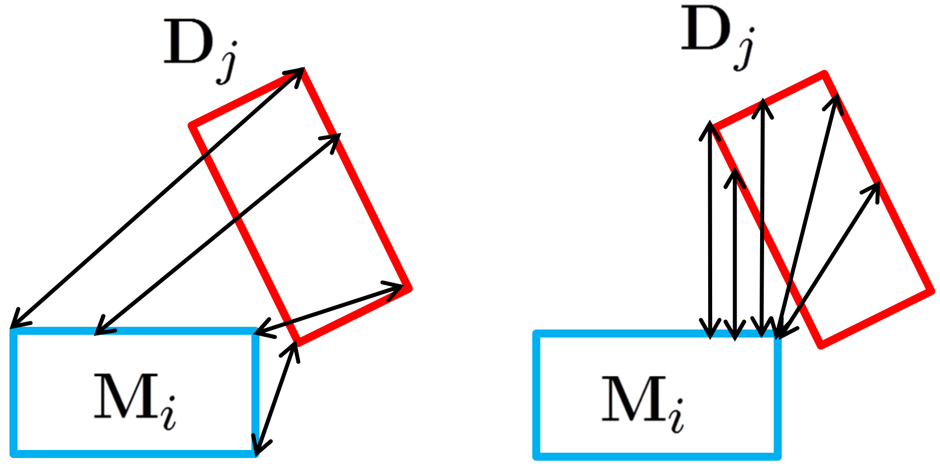
Cost function

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

Model point: $\mathbf{M}_i = (x_i, y_i)^\top$

Transformation parameters:

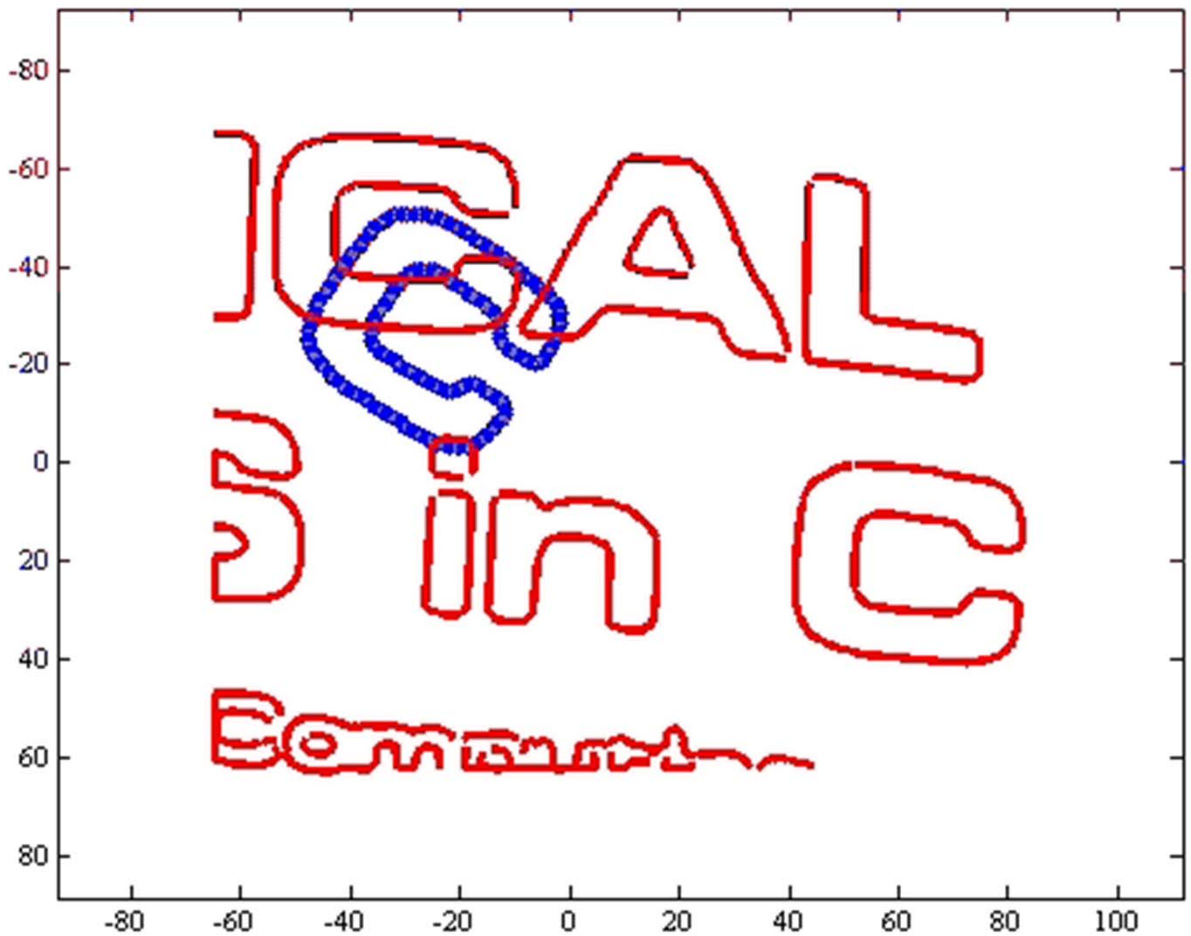
- rotation angle θ
- translation $\mathbf{t} = (t_x, t_y)^\top$



correct correspondences

closest point correspondences

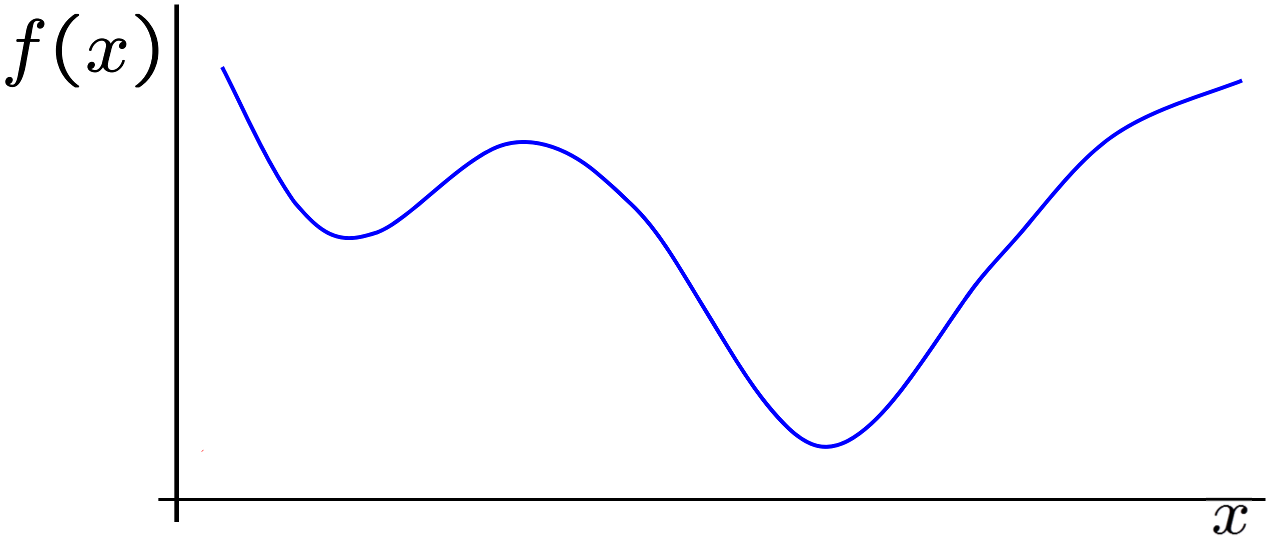
Performance



Unconstrained univariate optimization

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

$$\min_x f(x)$$



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

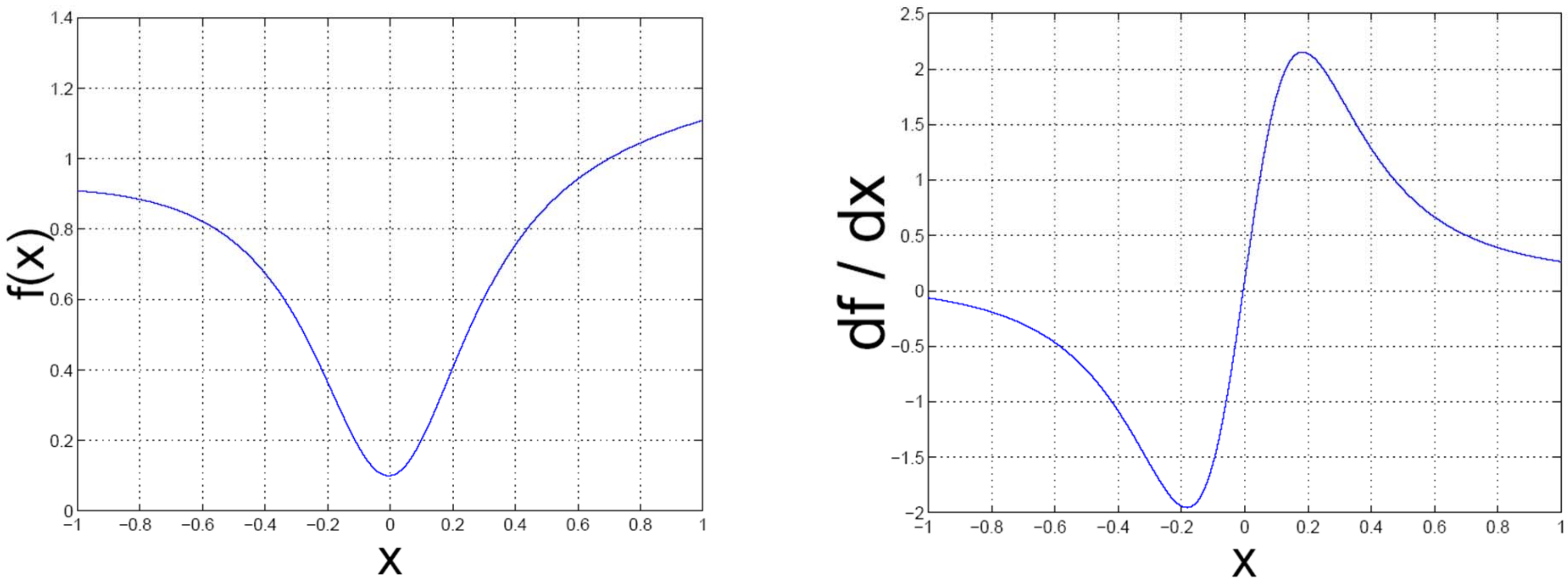
1. gradient descent
2. polynomial interpolation
3. Newton's method

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

A typical 1D function

As an example, consider the function

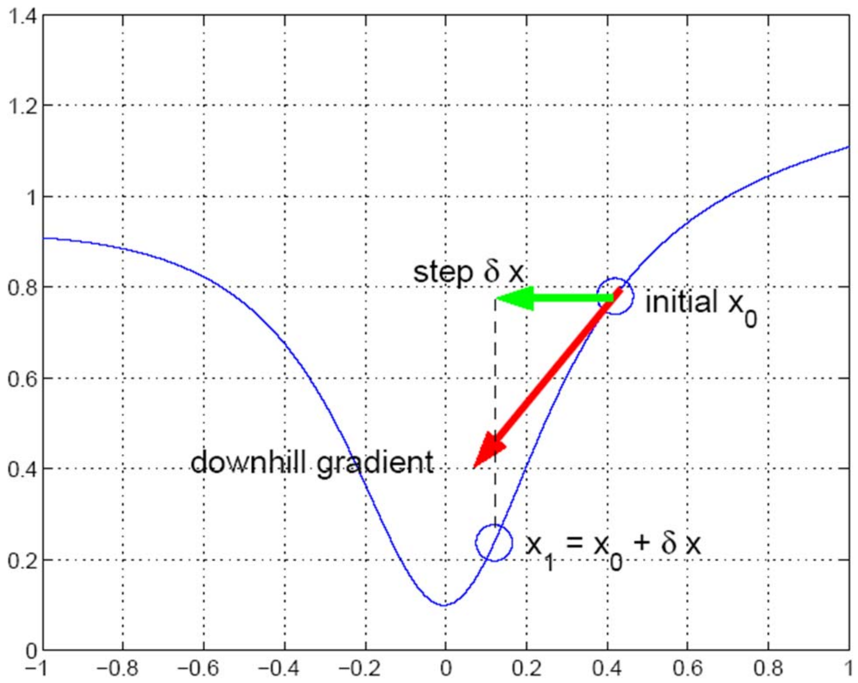
$$f(x) = 0.1 + 0.1x + x^2/(0.1 + x^2)$$



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

1. Gradient descent

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



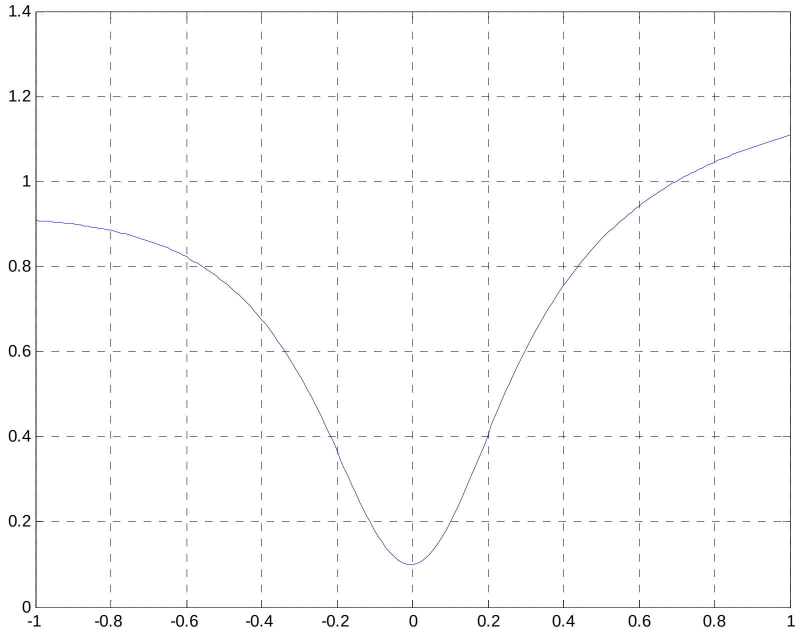
$$\delta x = -\alpha \frac{df}{dx}$$

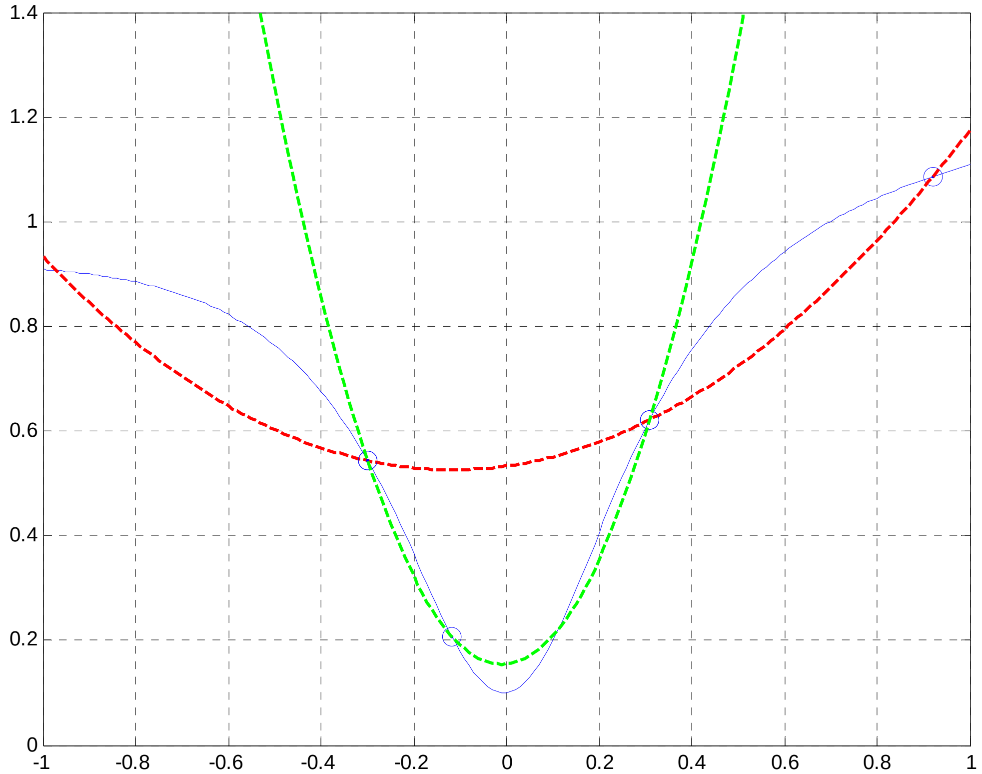
How to determine the step size δx ?

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 of 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 approximation to $f(x)$ using both gradient and curvature information at $\mathbf{x}$.

- 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 which minimizes this local quadratic approximation

$$f'(x+\delta x) = f'(x) + \delta x f''(x) = 0$$

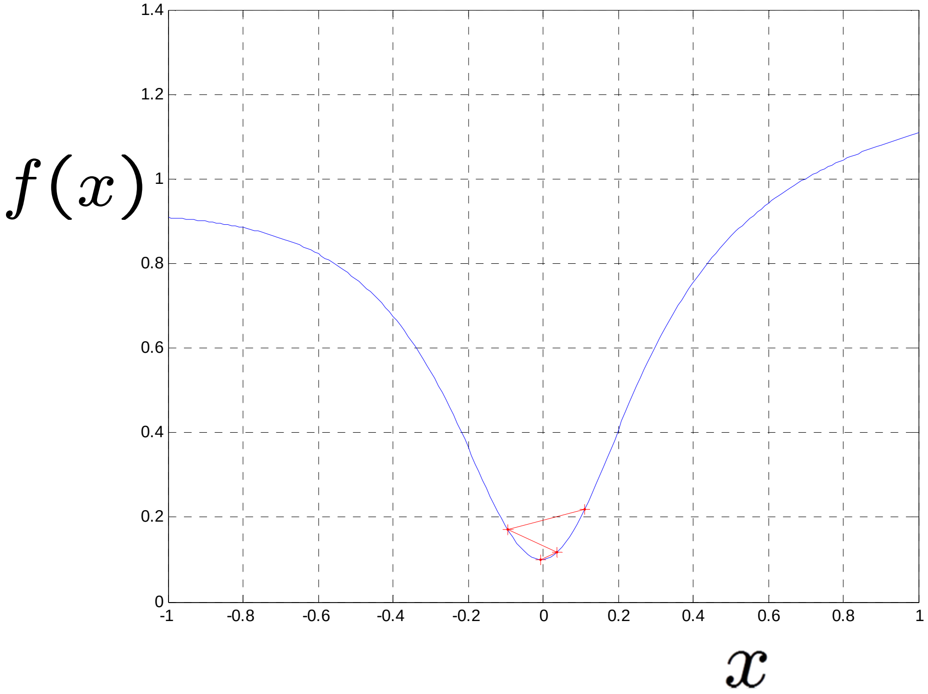
- and rearranging

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

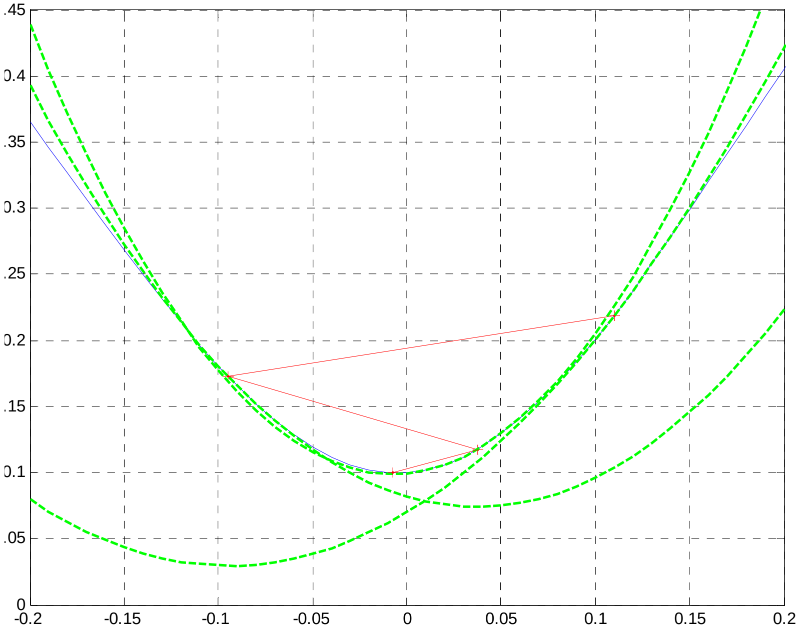
- Update for x

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

Newton iterations

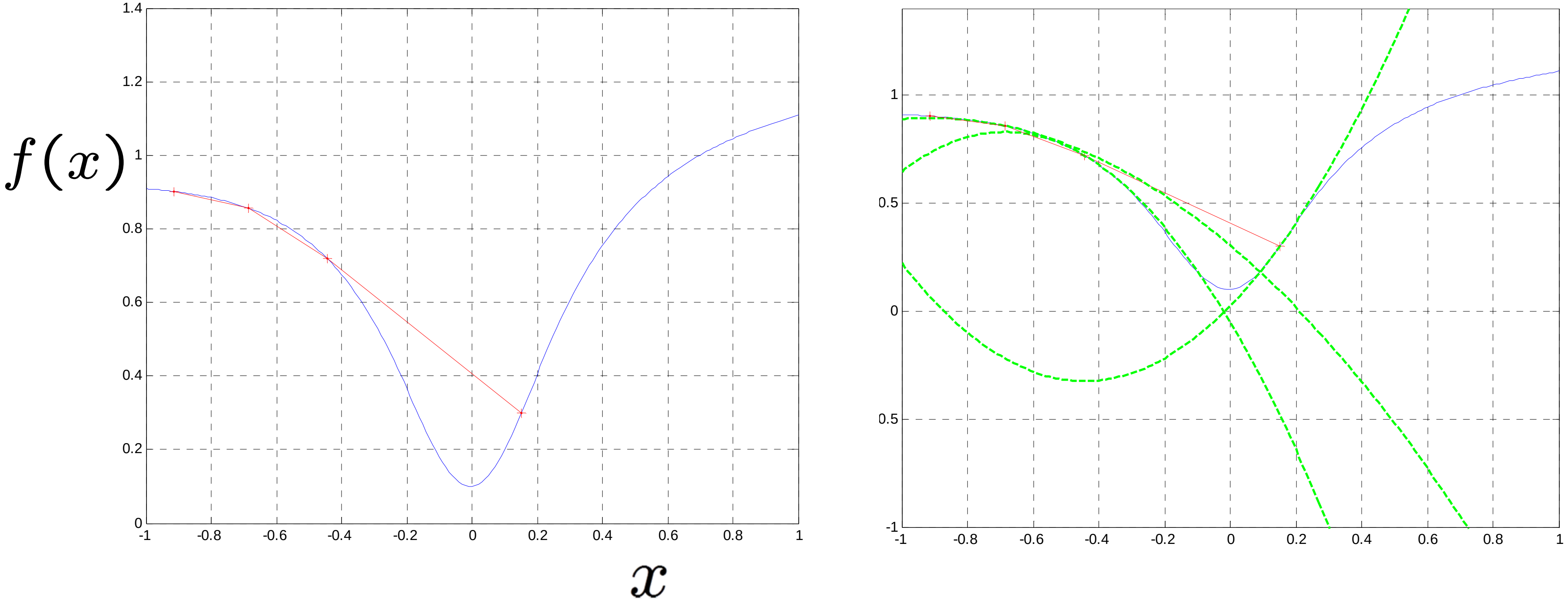


detail with quadratic approximations



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

- global convergence of Newton's method is poor
- often fails if the starting point is too far from the minimum



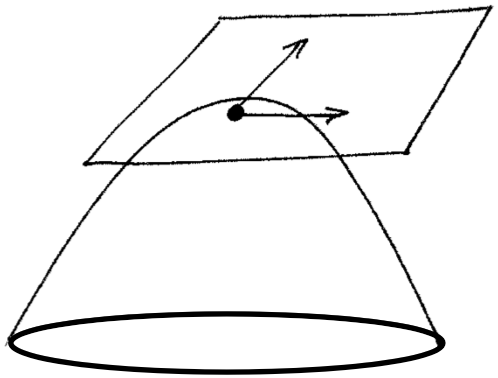
- in practice, must be used with a globalization strategy which reduces the step length until function decrease is assured

Stationary Points for Multidimensional functions

A function $f : \mathbb{R}^n \to \mathbb{R}$,

has a *stationary point* when the gradient

$$\nabla f = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)^\top = \mathbf{0}$$



$$\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)^\top = \mathbf{0}$$

Extension to N dimensions

- How big can N be?
  - problem sizes can vary from a handful of parameters to many thousands

- 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 may be approximated locally by its Taylor series expansion about a point $\mathbf{x}_0$

$$f(\mathbf{x}_0 + \mathbf{x}) \approx f(\mathbf{x}_0) + \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) \begin{pmatrix} x \\ y \end{pmatrix} + \frac{1}{2}(x, y) \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial x \partial y} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \text{h.o.t}$$

This is a generalization of the 1D Taylor series

$$f(x_0 + x) = f(x_0) + x f'(x_0) + \frac{x^2}{2} f''(x_0) + \text{h.o.t}$$

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

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

Taylor expansion in ND

A function may be approximated locally by its Taylor series expansion about a point $\mathbf{x}_0$

$$f(\mathbf{x}_0 + \mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \mathtt{H}\ \mathbf{x} + \mathrm{h.o.t}$$

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

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

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

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

The expansion to second order is a quadratic function

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

Properties of Quadratic functions

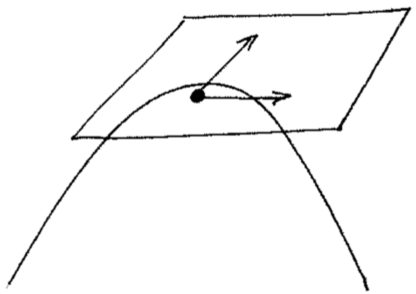
Taylor expansion

$$f(x_0 + \mathbf{x}) \;=\; f(\mathbf{x}_0) + \mathbf{g}^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \mathtt{H}\ \mathbf{x}$$

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

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

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



At a stationary point the behaviour is determined by H

$\mathtt{H}$ is a symmetrix matrix, and so has orthogonal eigenvectors

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

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

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.

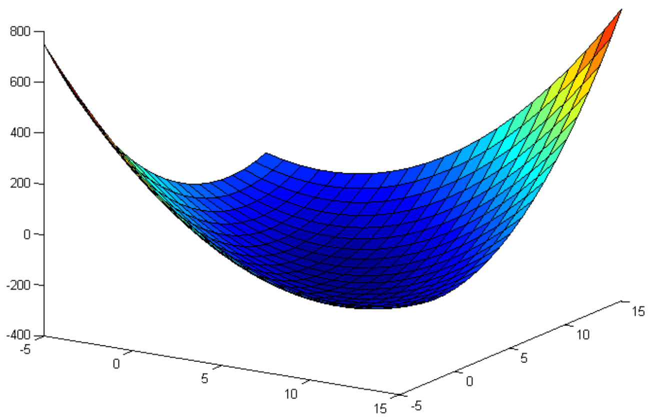
Examples of Quadratic functions

Case 1: both eigenvalues positive

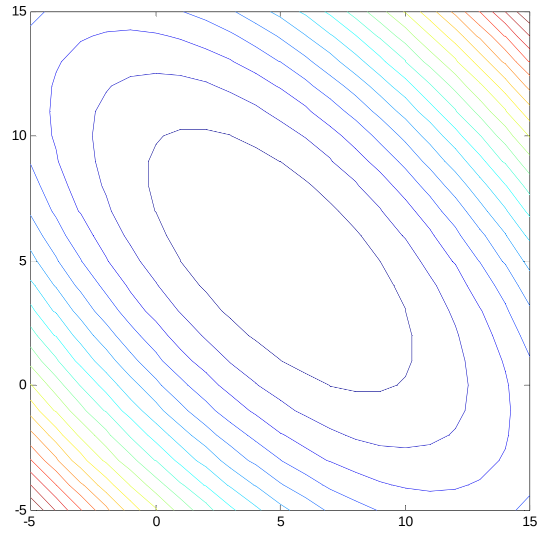
$$f(\mathbf{x}) = a + \mathbf{g}^\top \mathbf{x} + \frac{1}{2}\mathbf{x}^\top \mathtt{H}\ \mathbf{x}$$

with

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



minimum

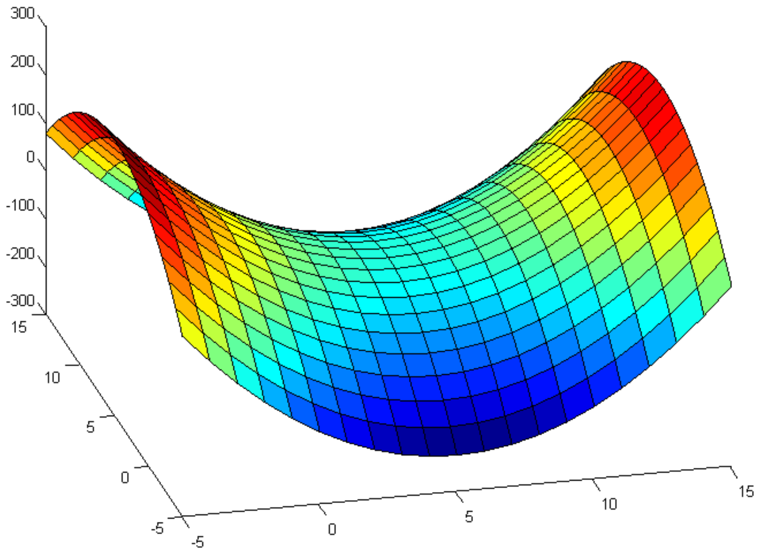


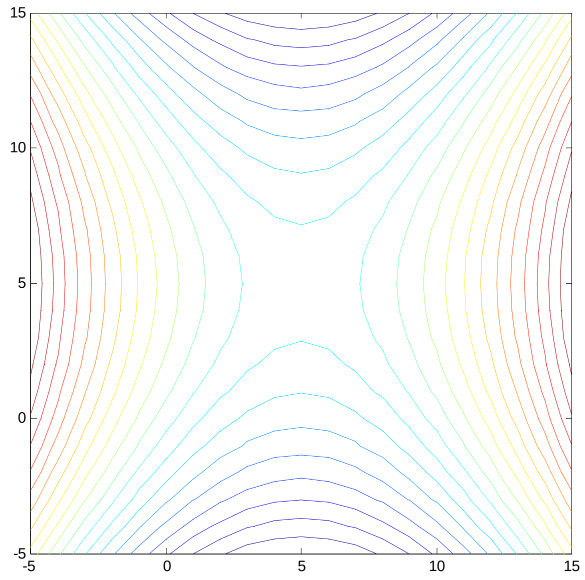
Case 2: eigenvalues have different signs

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

with

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





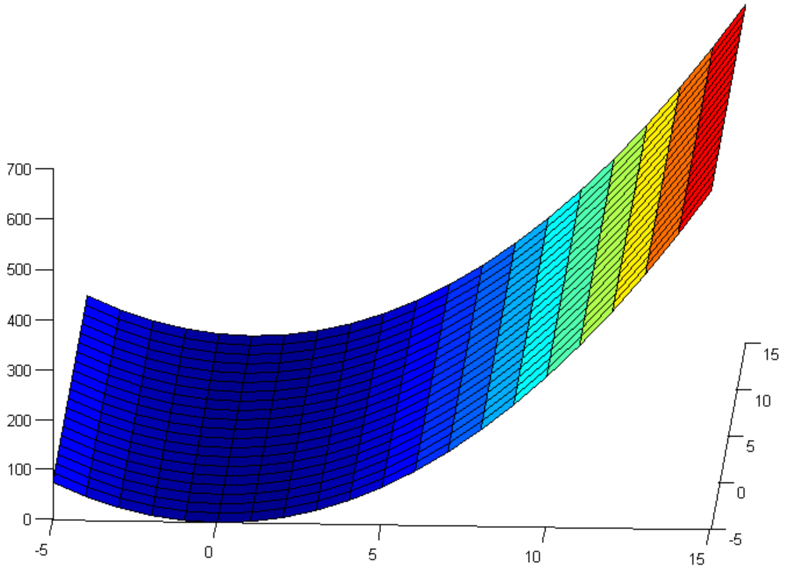
saddle surface: extremum but not a minimum

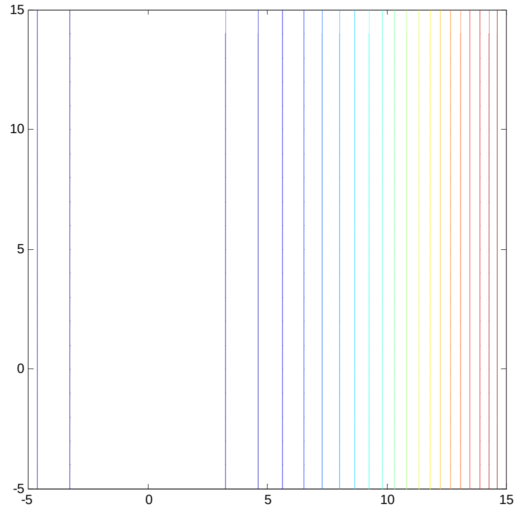
Case 3: one eigenvalue zero

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

with

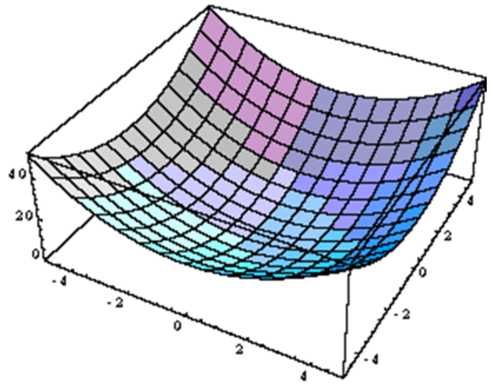
$$a = 0 \qquad \mathbf{g} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \mathtt{H} = \begin{bmatrix} 6 & 0 \\ 0 & 0 \end{bmatrix}$$



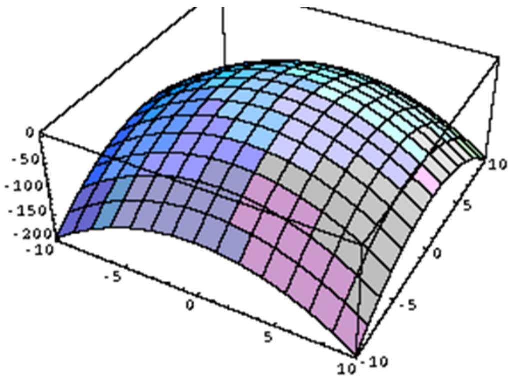


parabolic cylinder

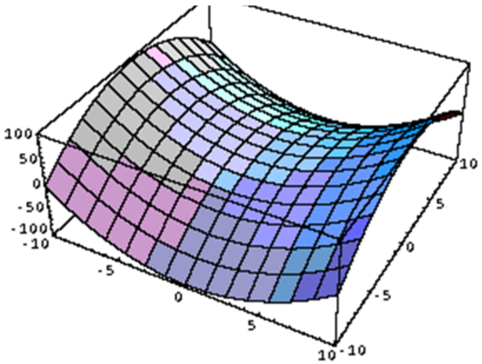
Types of Stationary Point



Hessian positive definite
Convex function.
Minimum point.



Hessian negative definite
Concave function
Maximum point.

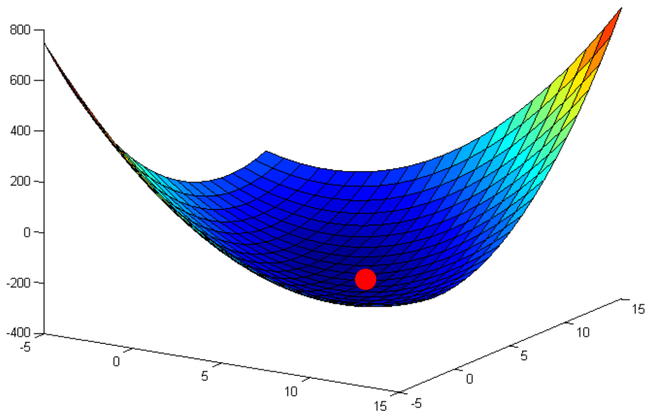


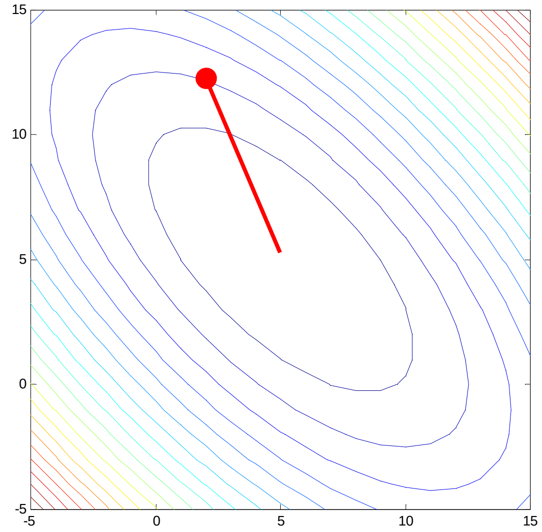
Hessian mixed.
Surface has negative curvature.
Saddle point.

Optimization in N dimensions – line search

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

• Use methods developed in 1D (e.g. polynomial interpolation)





An Optimization Algorithm

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

4. check for convergence (termination criteria) e.g. $\nabla f = \mathbf{0}$

Reduces 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 descent method chooses $\mathbf{p}_n$ to be parallel to the negative gradient

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

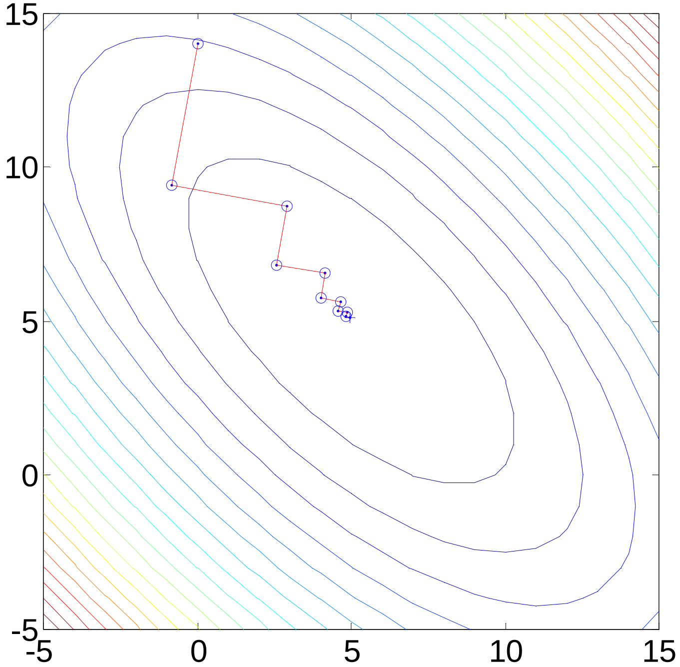
Step-size α_n is chosen to minimize $f(\mathbf{x}_n + \alpha_n \mathbf{p}_n)$. For quadratic forms there is a closed form solution :

$$\alpha_n = -\frac{\mathbf{p}_n^\top \mathbf{p}_n}{\mathbf{p}_n^\top \mathtt{H}\ \mathbf{p}_n}$$

[exercise]

Example

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



Steepest descent ($x_0 = [0, 14]$)

- 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

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 2nd derivatives) and Newton-like methods for general functions