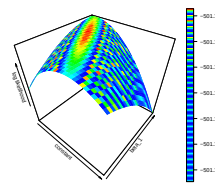
Numerical Optimiziation

Michael Lebacher
Johannes Stoiber
Ken Schröder

Numerical Introductory Course
Seminar
Ladislaus von Bortkiewicz Chair of Statistics
Humboldt–Universität zu Berlin

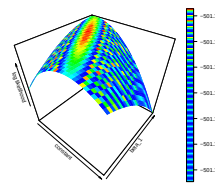
Outline

1. Motivation
2. Methods of optimization
3. Practical examples
4. Algorithms
   - Newton-Raphson
   - Gradient descent
   - Nelder-Mead
5. Apply algorithm to solve a maximum likelihood problem for real data
6. Compare results
7. Conclusion



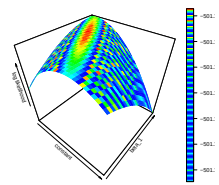
Motivation

- Optimize some objective function
  - Cost function of a business
- Why numerical?
  - Analytical solution might not exist
  - Too complex to deal with manually
  - Computer cannot provide closed form solutions, in general



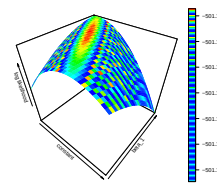
Optimization problems

- Discrete vs. continuous
  - Integer programming
- Constrained vs. non constrained
  - Linear optimization
- Local vs. global minimum
  - Convex programming
- Gradient vs. non gradient
  - Newton methods
  - Nelder-Mead
  - BFGS



Gradient and non gradient methods

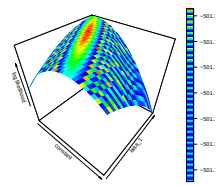
- Gradient based
  - Direction
  - Stepsize
- Non gradient based
  - Works with non-differentiable functions
  - Less knowledge required
  - Computationally intensive



Structure of algorithms

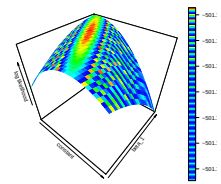
- Objective function $f(\mathbf{x})$
- Starting value for $\mathbf{x}$
- Define updater function
- Define termination criterion

1. $\mathbf{x}_{new} = \mathbf{x}_{old} + updater$
2. Convergence criterion satisfied?
   - $\Rightarrow$ No: start over at step 1.
   - $\Rightarrow$ Yes: terminate procedure.



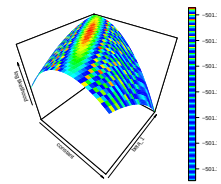
Our algorithms

- Normalized for stability
- $\mathbf{x}_{start}$ is randomly generated
- Termination criterion
  - $||\mathbf{x}_{new} - \mathbf{x}_{old}|| < t^*$
  - $||f(\mathbf{x}_{new}) - f(\mathbf{x}_{old})|| < t^*$
- Compared performance to `optim()`-package in R.
  - Number of iterations required
  - Accuracy of approximation



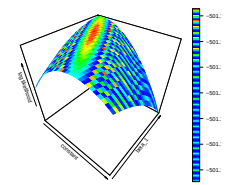
Practical examples

- Operations research
- Finance
- Statistics
  - Principal components
  - Least squares
  - Maximum likelihood etc.



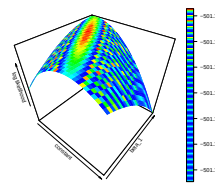
Algorithms

- Gradient descent
- Newton-Raphson
- Nelder-Mead



Gradient descent

- Also known as *steepest descent* or *the method of the steepest descent*
- Standard procedure for the numerical solution of nonlinear equations
- Method for unconstrained optimization problems in n-dimensional spaces
- Requires the first derivative of the objective function (gradient)



Gradient descent

Problem

Minimize a function $f(x)$, $x \in \mathbb{R}^n$
$f \in C^1(\mathbb{R}^n, \mathbb{R})$, $n \geq 1$

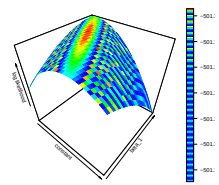
Idea

Gradient descent is based on the idea that a differentiable function decreases fastest if one goes in the direction of the negative gradient. Hence the iteration at the $(n+1)$th step works as following:

$$x^{n+1} = x^n - \alpha^n \nabla f(x^n)$$

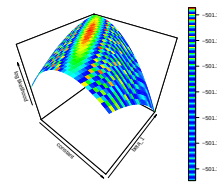
Notation

- Objective function: $f(x)$
- Gradient: $\nabla f(x)$
- The learning parameter α^n determines the step size



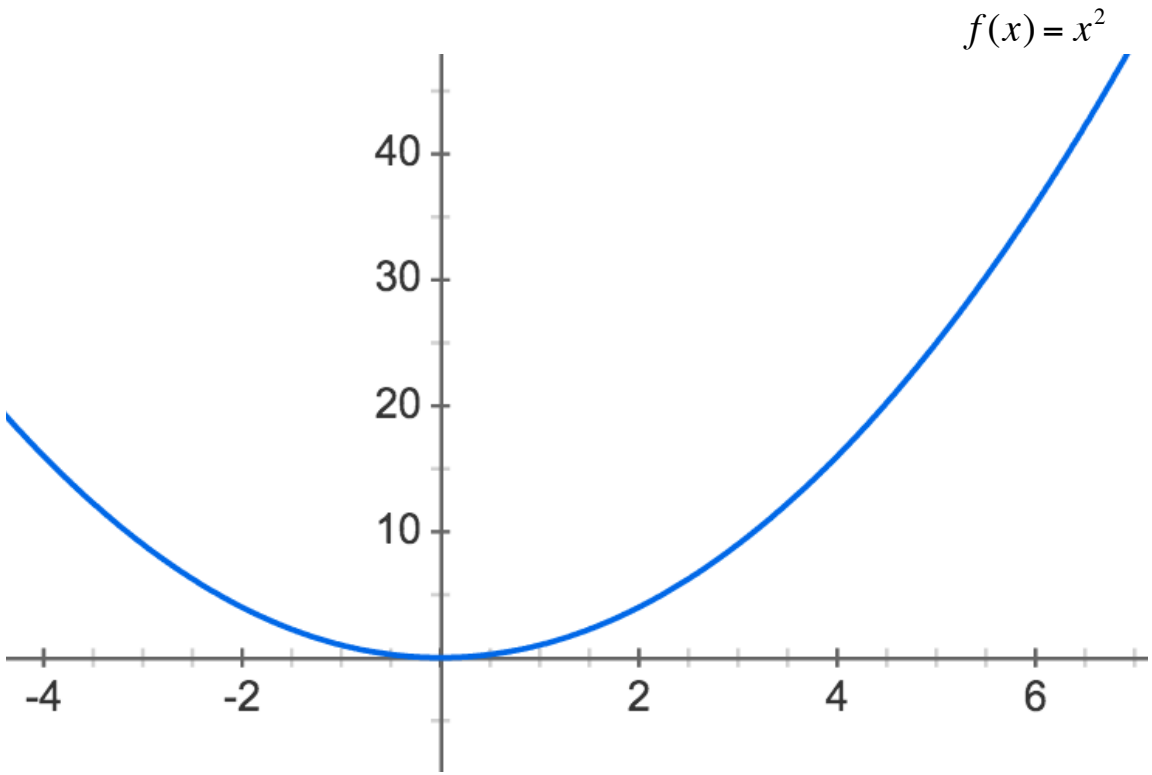
Speed of convergence depends crucially on the learning parameter

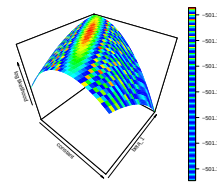
- $\alpha^n = \alpha \ \forall \ n$ is often used
- $\alpha^n = h(n)$ where $h(n)$ gradually decreases with each iteration.
- $\alpha^n = \begin{cases} \beta\alpha^{n-1} & L(x^n) < 0 \\ \alpha^{n-1} & L(x^n) \geq 0 \end{cases}$ where $\beta \in (0,1)$ and $L(x^n)$ specifies the change in a predefined Loss-Function (for example: $L(x^n) = f(x^{n+1}) - f(x^n)$)
- ....



Gradient descent - Example

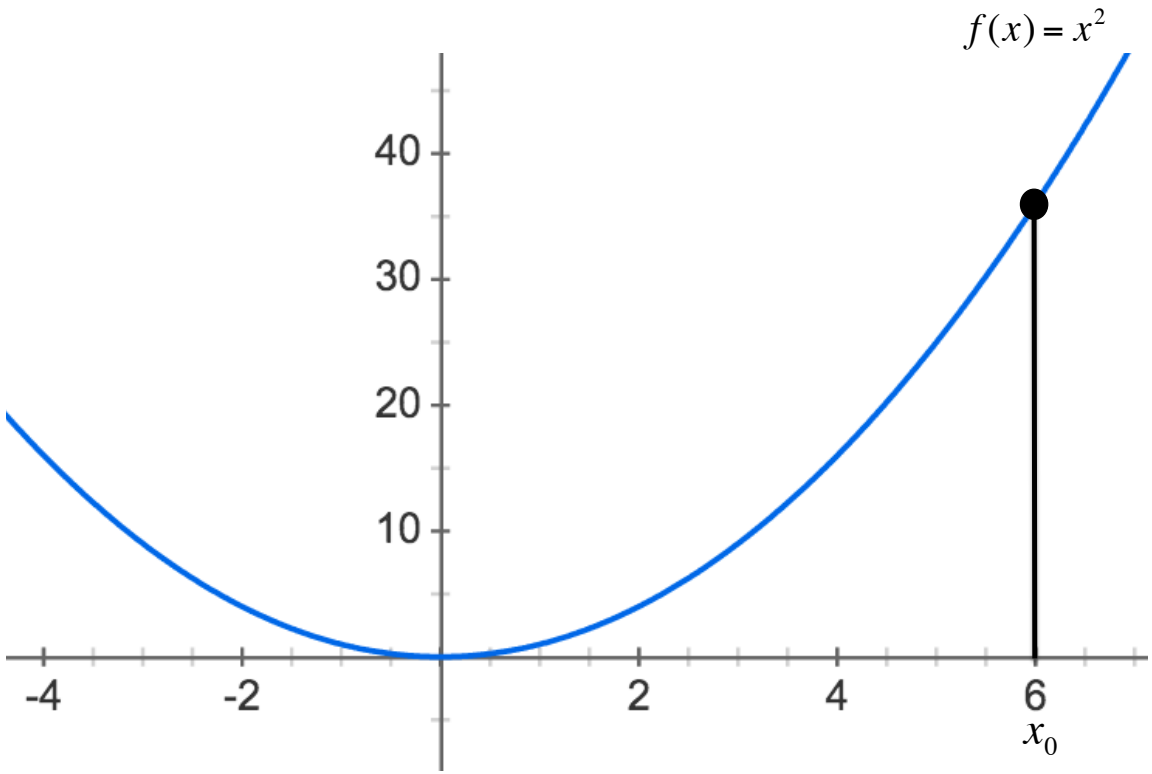
$f(x) = x^2$, $\nabla f(x) = 2x$, $\alpha = 0.25$
$x^{n+1} = x^n - 0.25\nabla f(x^n)$

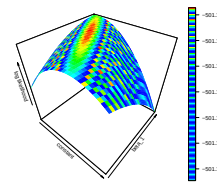




Gradient descent - Example

$f(x) = x^2$, $\nabla f(x) = 2x$, $\alpha = 0.25$
$x_0 = 6$

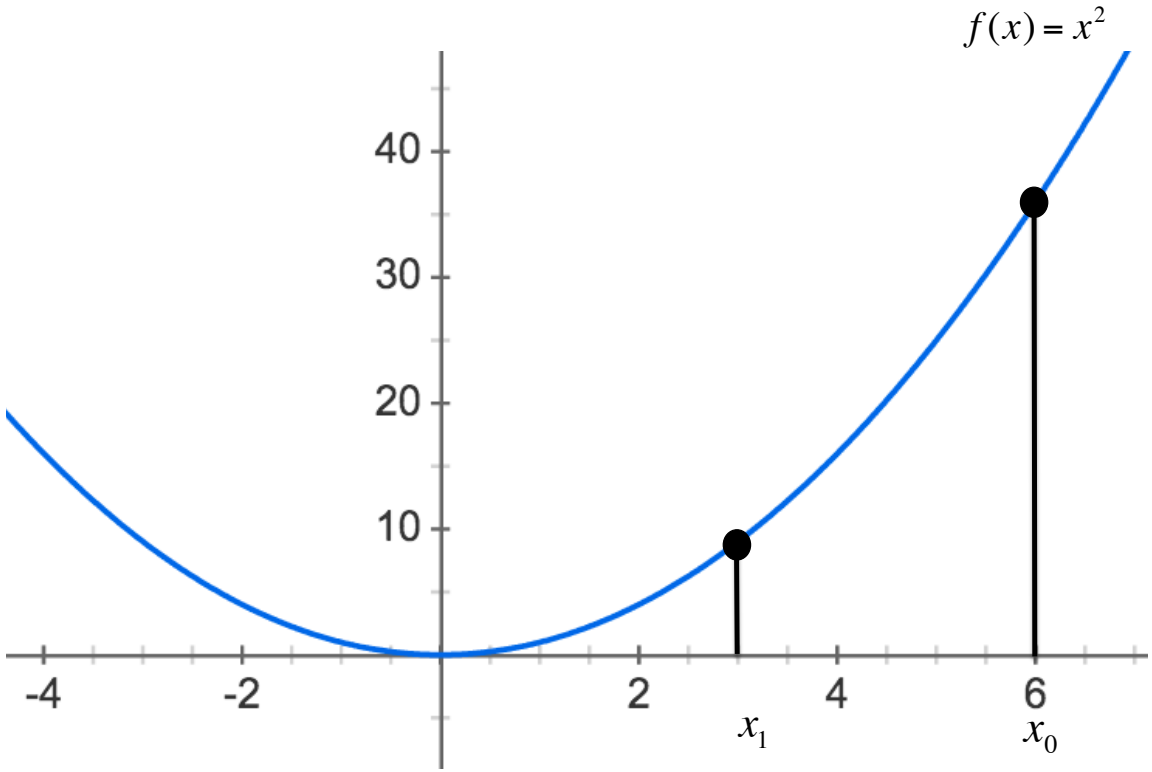


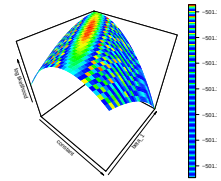


Gradient descent - Example

$f(x) = x^2$, $\nabla f(x) = 2x$, $\alpha = 0.25$, $x_0 = 6$

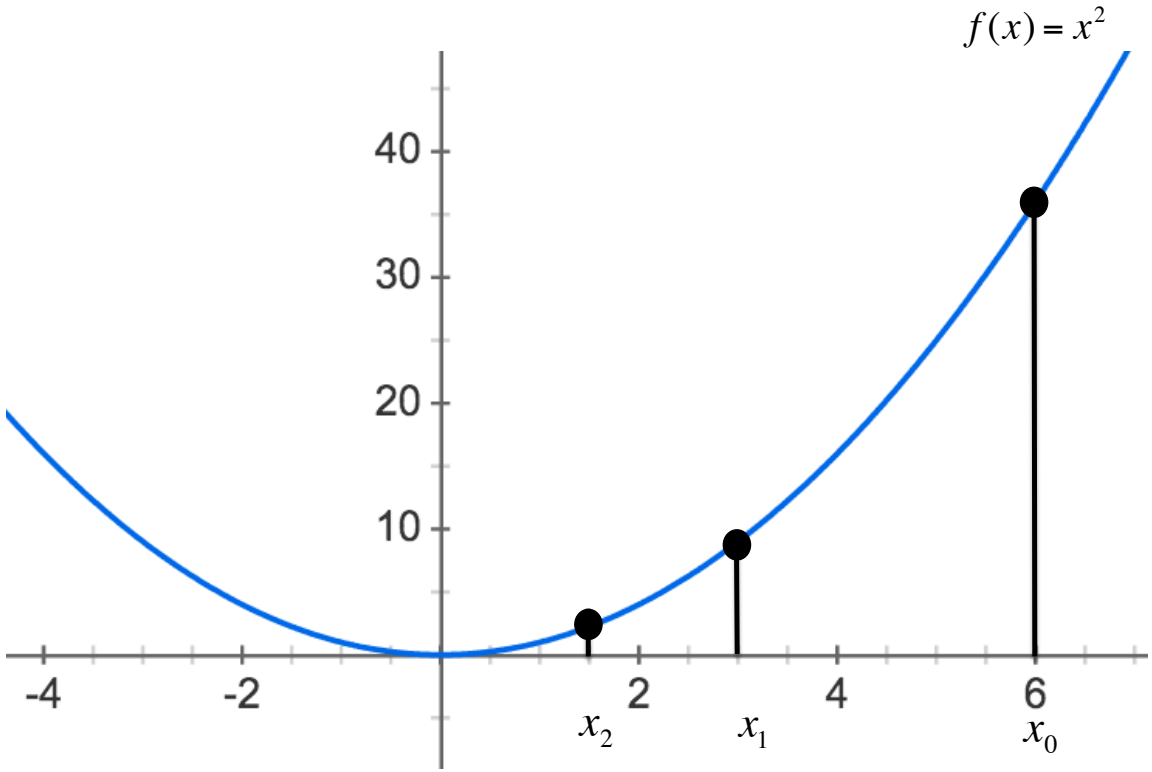
$x_1 = 6 - 0.5 * 6 = 3$

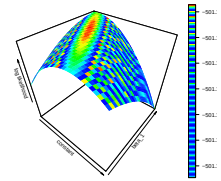




Gradient descent - Example

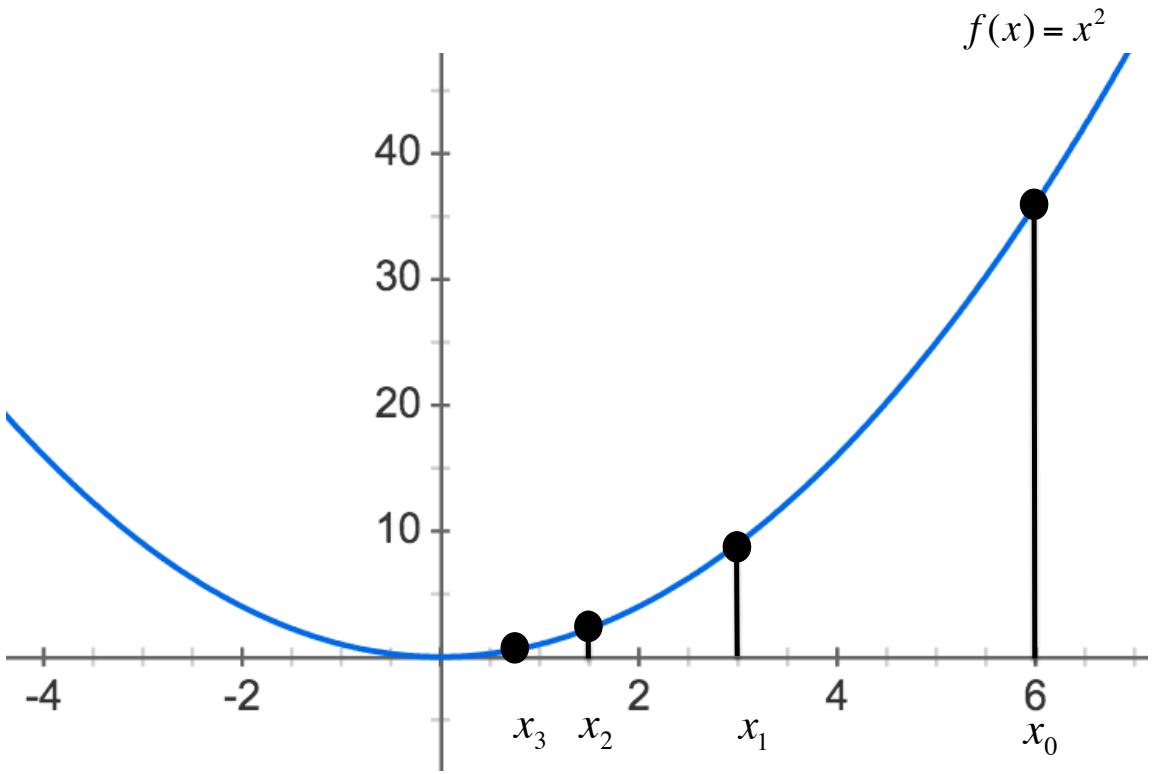
$f(x) = x^2$, $\nabla f(x) = 2x$, $\alpha = 0.25$, $x_1 = 3$
$x_2 = 3 - 0.5 * 3 = 1.5$

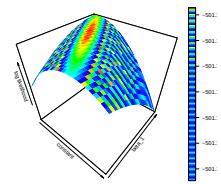




Gradient descent - Example

$f(x) = x^2$, $\nabla f(x) = 2x$, $\alpha = 0.25$, $x_2 = 1.5$
$x_3 = 1.5 - 0.5 * 1.5 = 0.75$

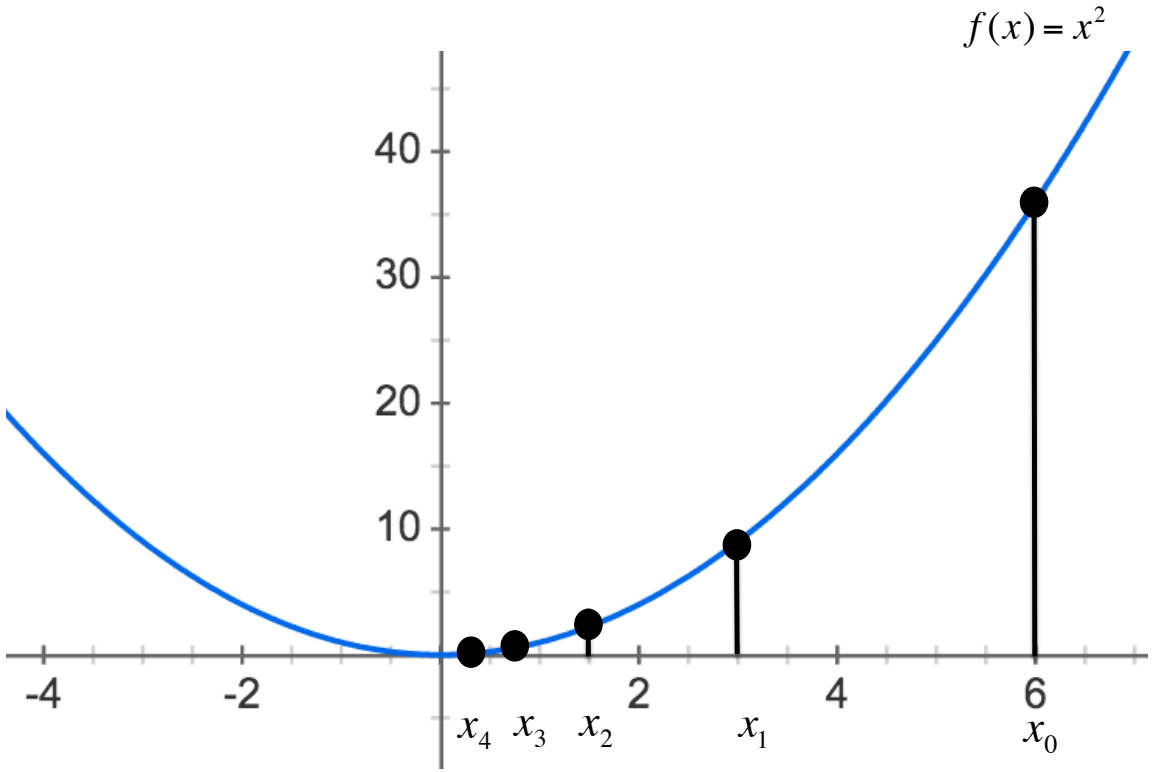


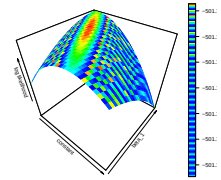


Gradient descent - Example

$f(x) = x^2$, $\nabla f(x) = 2x$, $\alpha = 0.25$, $x_3 = 0.75$

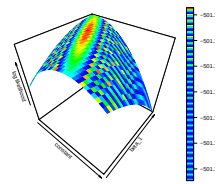
$x_4 = 0.75 - 0.5 * 0.75 = 0.375$





Newton-Raphson

- Named after Sir Isaac Newton and Joseph Raphson
- Very popular procedure for the numerical solution of nonlinear equations
- Method for unconstrained optimization problems in n-dimensional spaces
- Requires derivatives of the objective function (gradient and hessian)
- Is based on the linearization of the objective function



Newton-Raphson

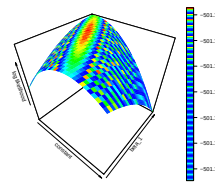
Problem

Minimize a function $f(x)$, $x \in \mathbb{R}^n$

$f \in C^2(\mathbb{R}^n, \mathbb{R})$, $n \geq 1$

Idea

Solve the minimization problem by application of a local linearization which can be solved easily



Newton-Raphson

Apply a Taylor expansion of second order near the optimum x^*:

$$f(x) \approx f(x^*) + \nabla f(x^*)'(x - x^*) + 0.5(x - x^*)'\nabla^2 f(x^*)(x - x^*)$$

This linearization can be differentiated w.r.t. x

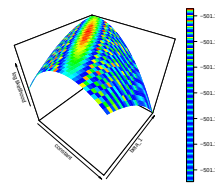
$$\nabla f(x) \approx \nabla f(x^*) + \nabla^2 f(x^*)(x - x^*)$$

A minimizer must satisfy $\nabla f(x) = 0$. Therefore we get

$$\nabla f(x^*) + \nabla^2 f(x^*)(x - x^*) = 0$$

This results after rearrangement (given a nonsingular hessian) in the following expression

$$x = x^* - [\nabla^2 f(x^*)]^{-1} \nabla f(x^*)$$

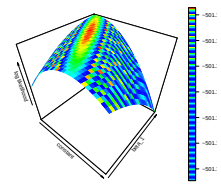


Newton-Raphson

Updater function

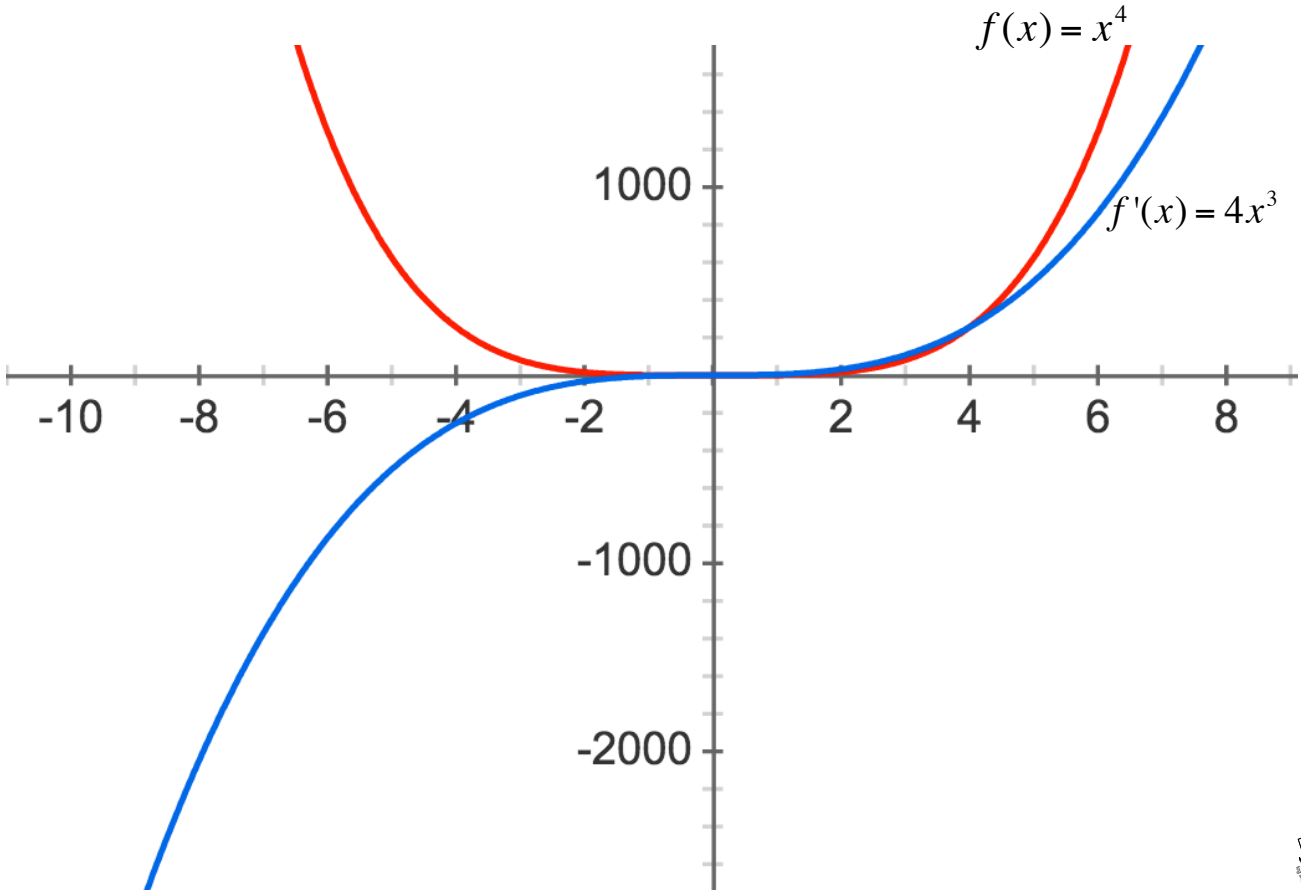
$$x^{n+1} = x^n - [\nabla^2 f(x^n)]^{-1} \nabla f(x^n)$$

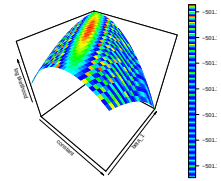
- Can be interpreted as gradient descent algorithm employing the hessian as learning parameter: $\alpha^n = [\nabla^2 f(x^n)]^{-1}$
- Very few iterations needed if $\nabla^2 f(x)$ is positive definite
- However: We must know the hessian and its computing might be very costly



Newton-Raphson - Example

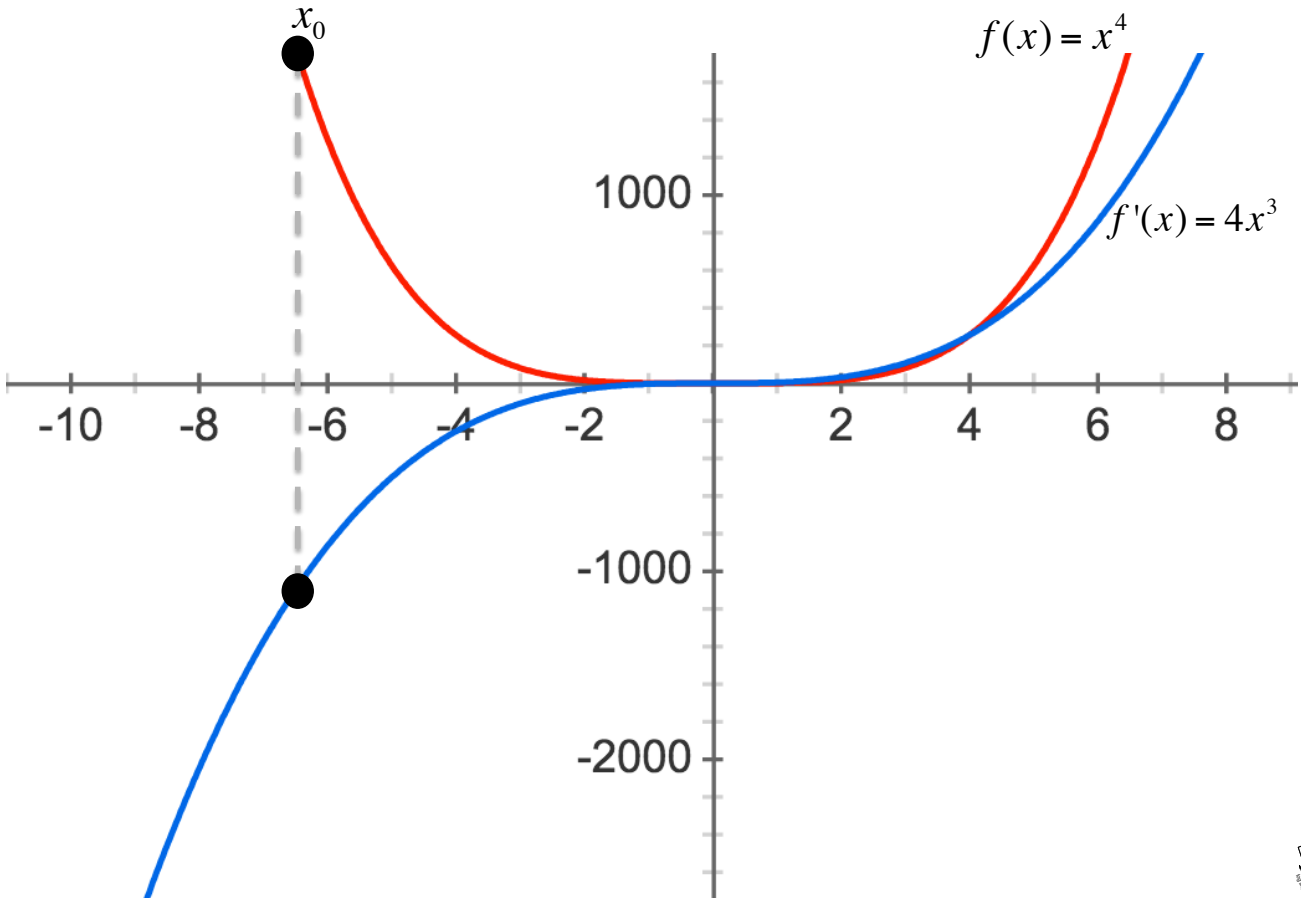
$f(x) = x^4$, $\nabla f(x) = 4x^3$, $\nabla^2 f(x) = 12x^2$

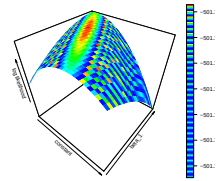




Newton-Raphson - Example

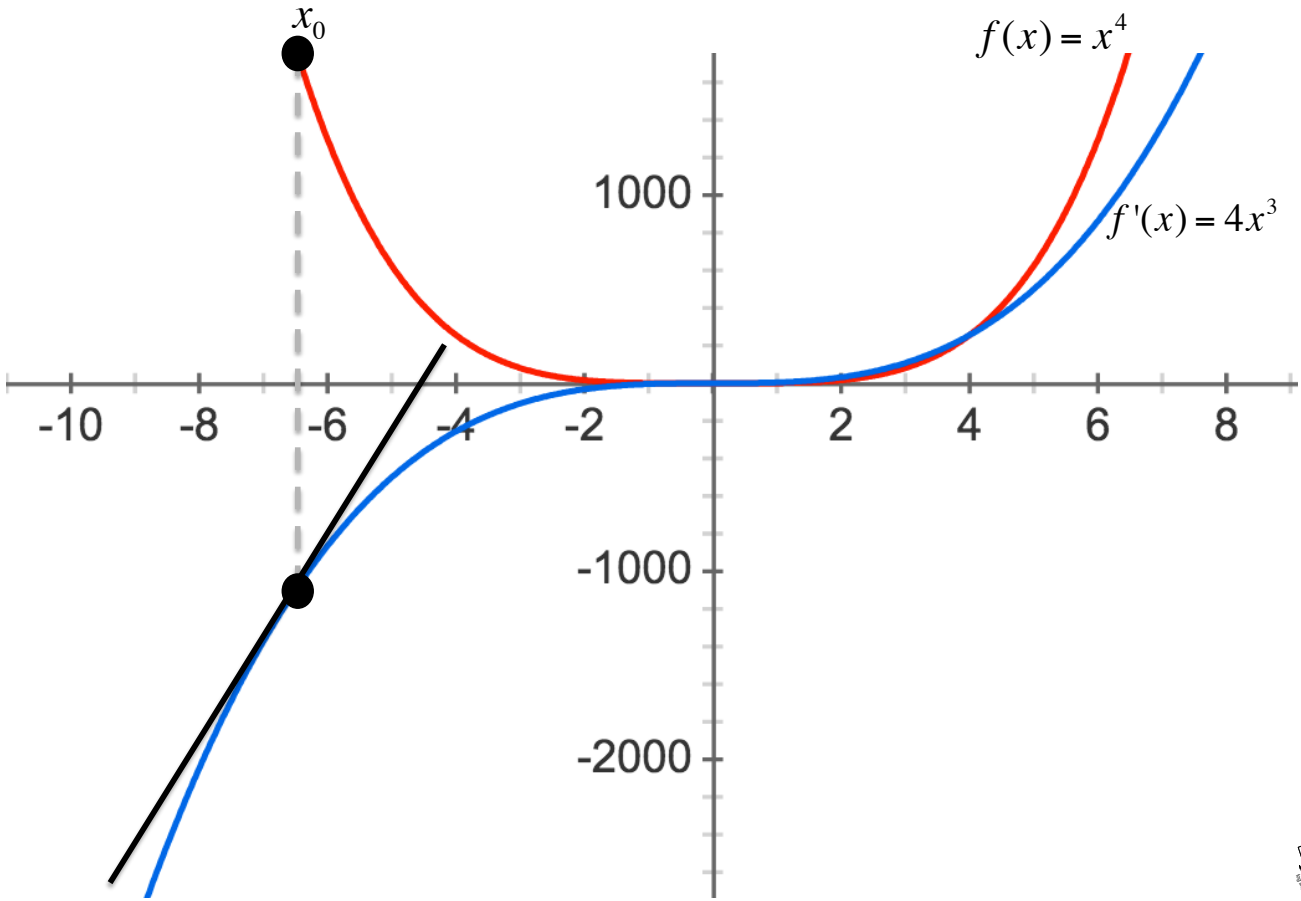
$f(x) = x^4$, $\nabla f(x) = 4x^3$, $\nabla^2 f(x) = 12x^2$

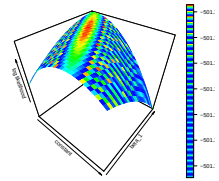




Newton-Raphson - Example

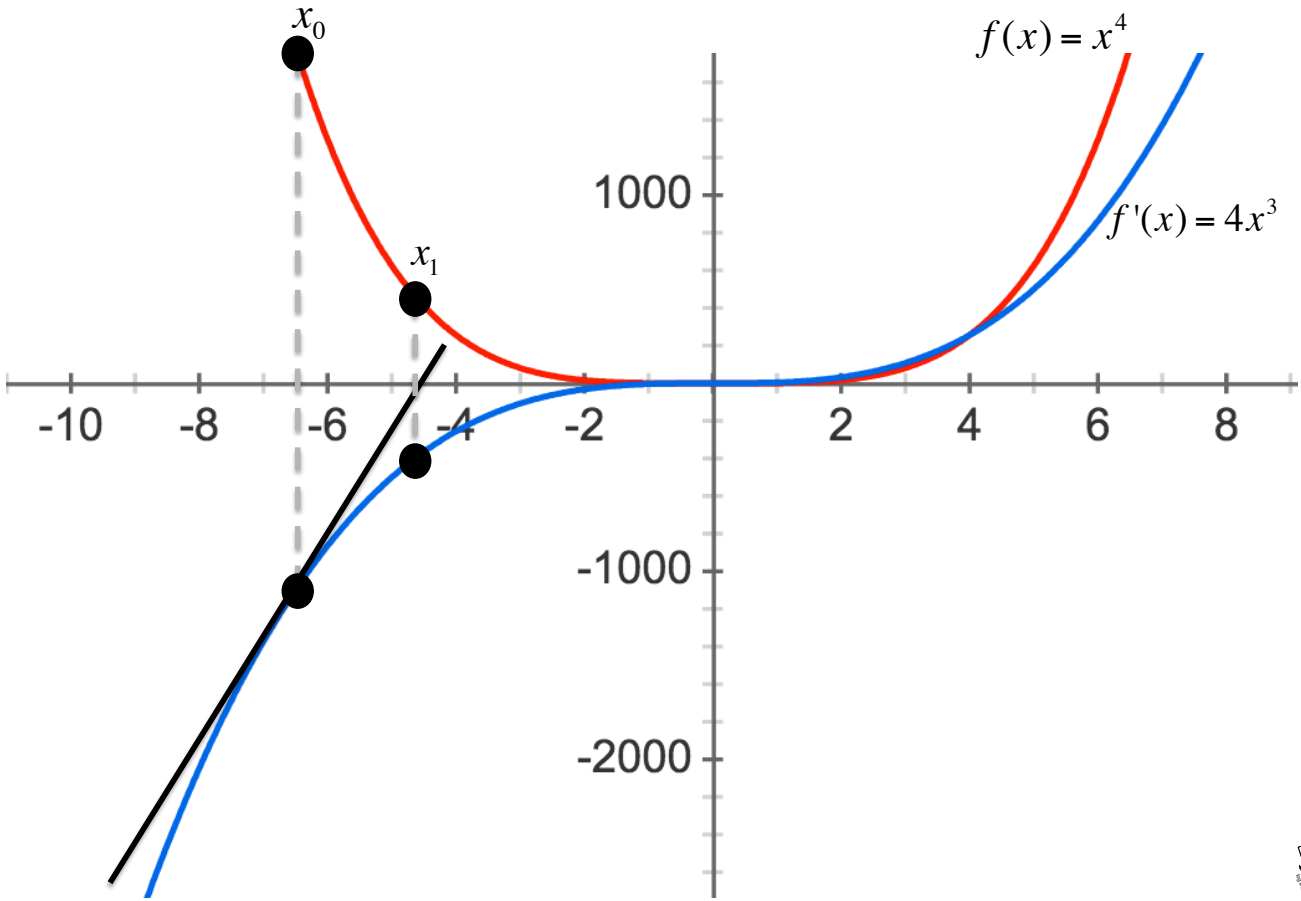
$f(x) = x^4$, $\nabla f(x) = 4x^3$, $\nabla^2 f(x) = 12x^2$

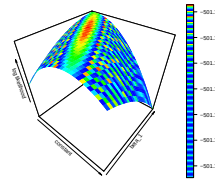




Newton-Raphson - Example

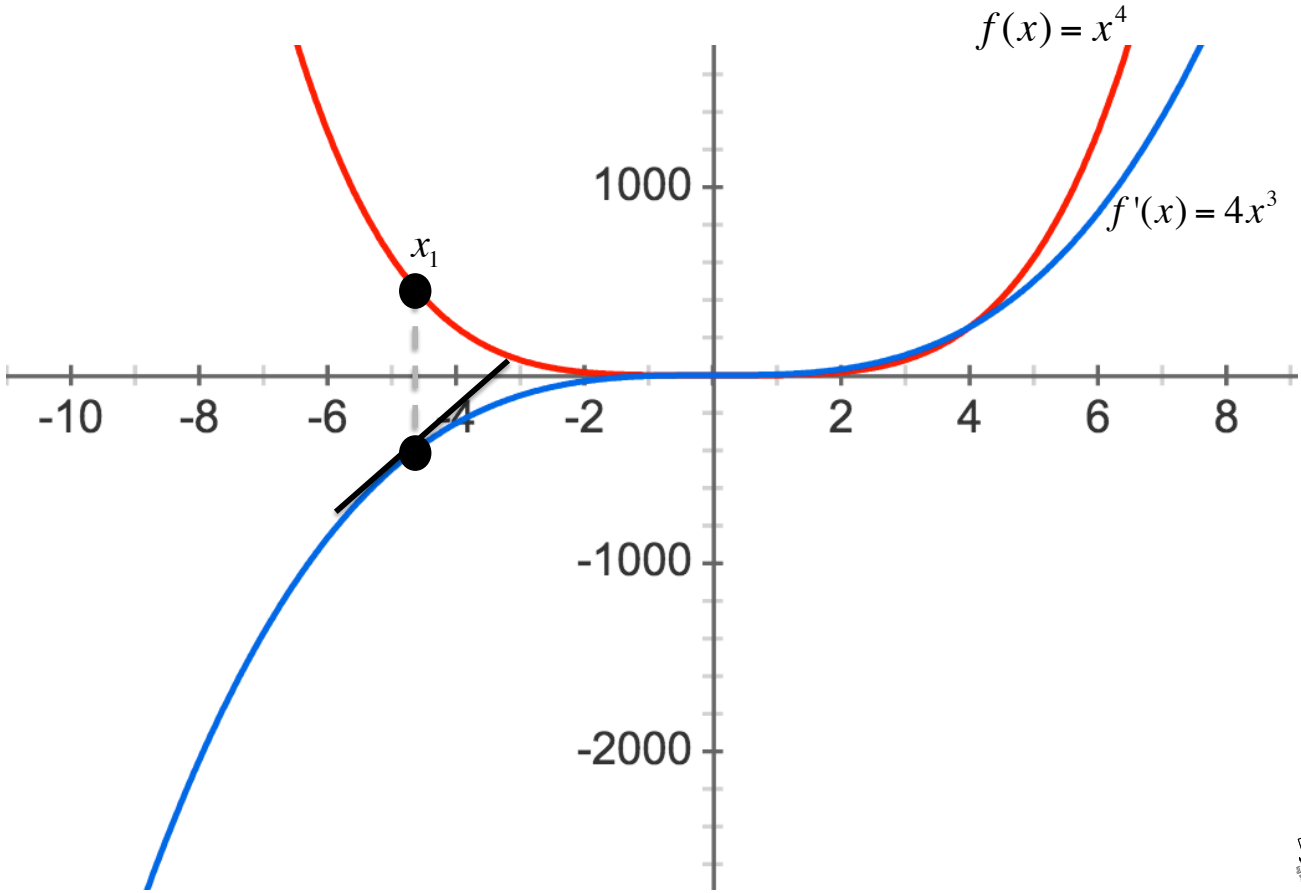
$f(x) = x^4$, $\nabla f(x) = 4x^3$, $\nabla^2 f(x) = 12x^2$

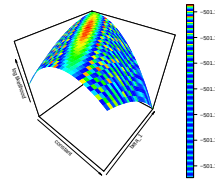




Newton-Raphson - Example

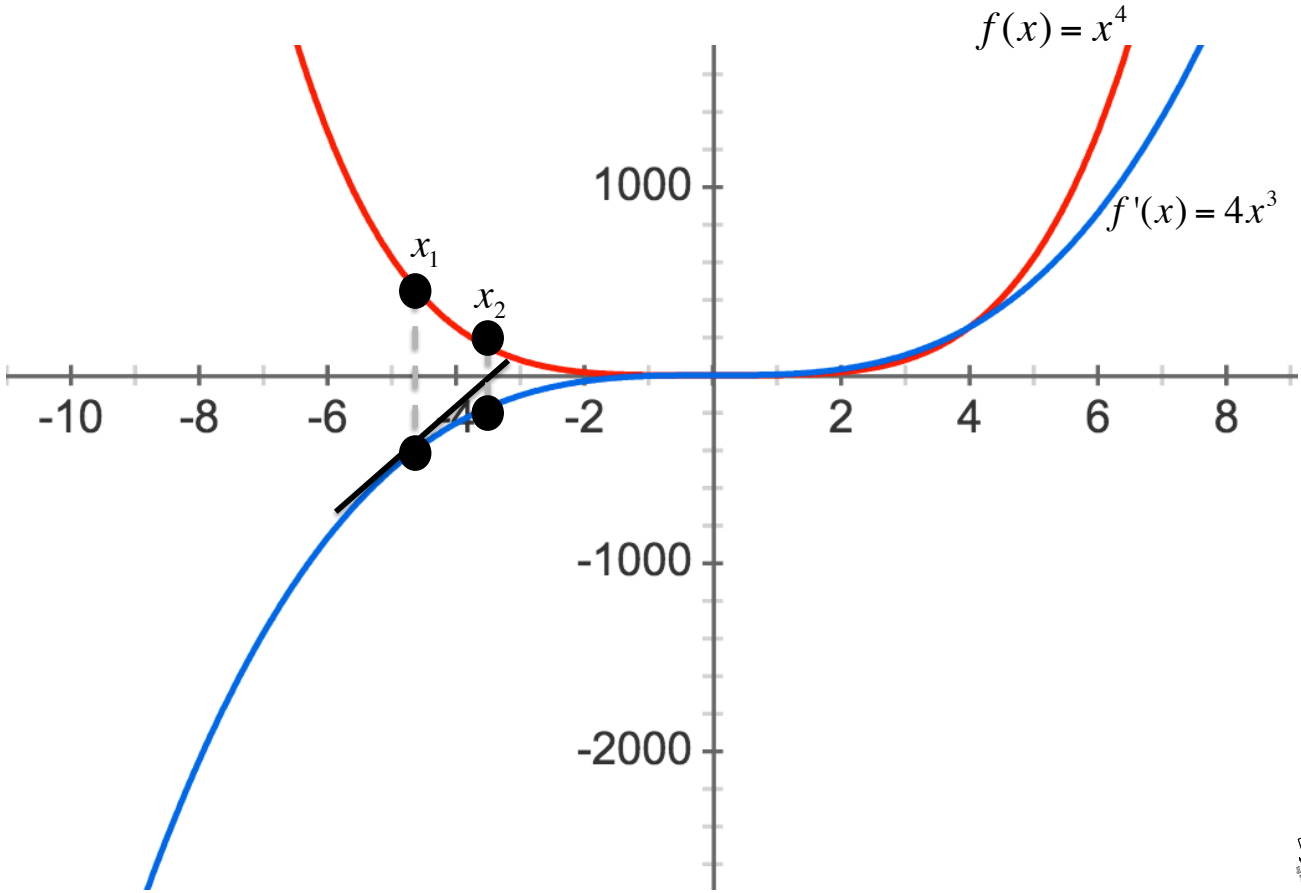
$f(x) = x^4$, $\nabla f(x) = 4x^3$, $\nabla^2 f(x) = 12x^2$

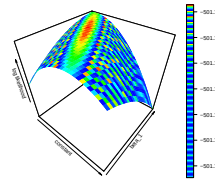




Newton-Raphson - Example

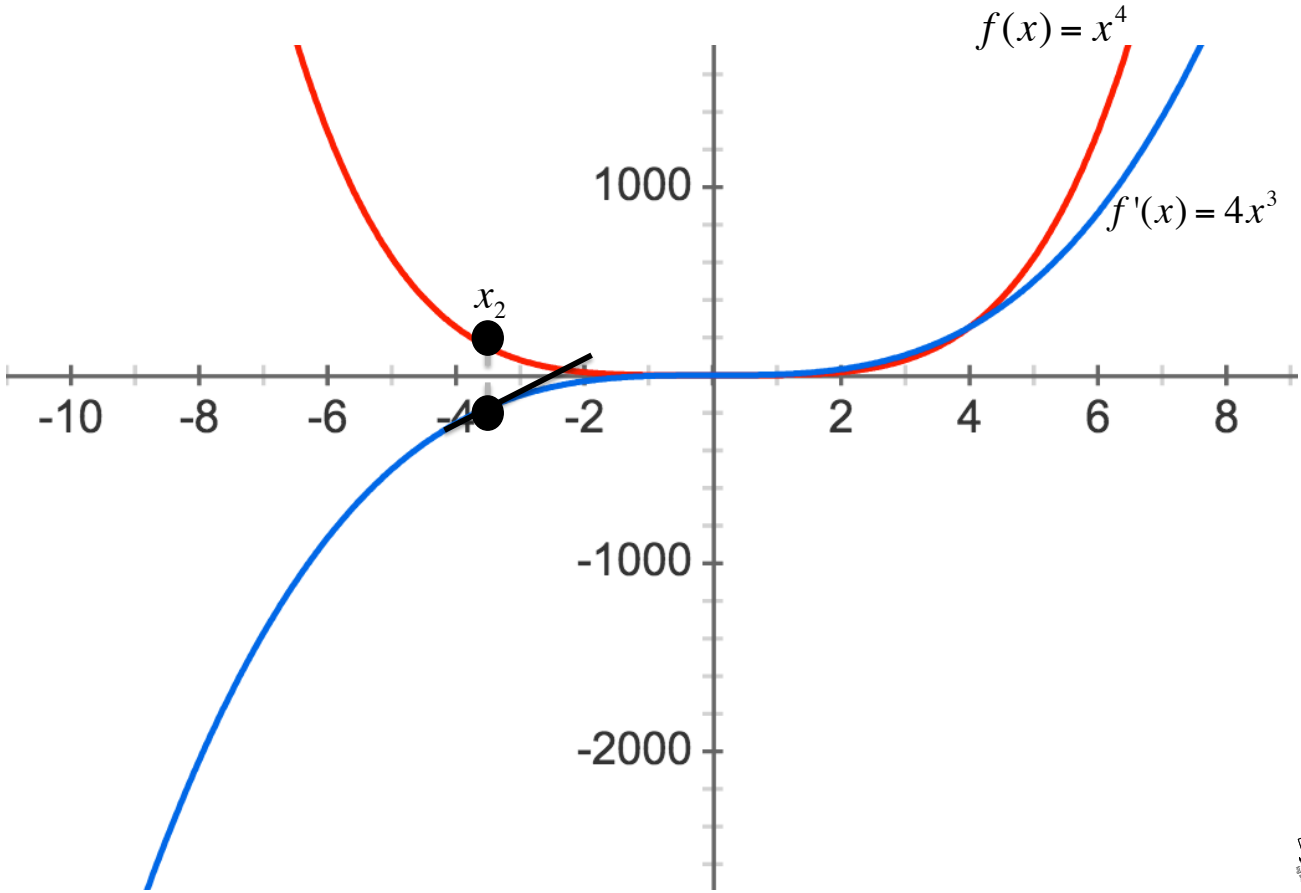
$f(x) = x^4$, $\nabla f(x) = 4x^3$, $\nabla^2 f(x) = 12x^2$

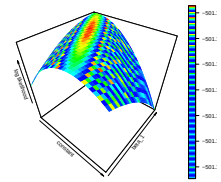




Newton-Raphson - Example

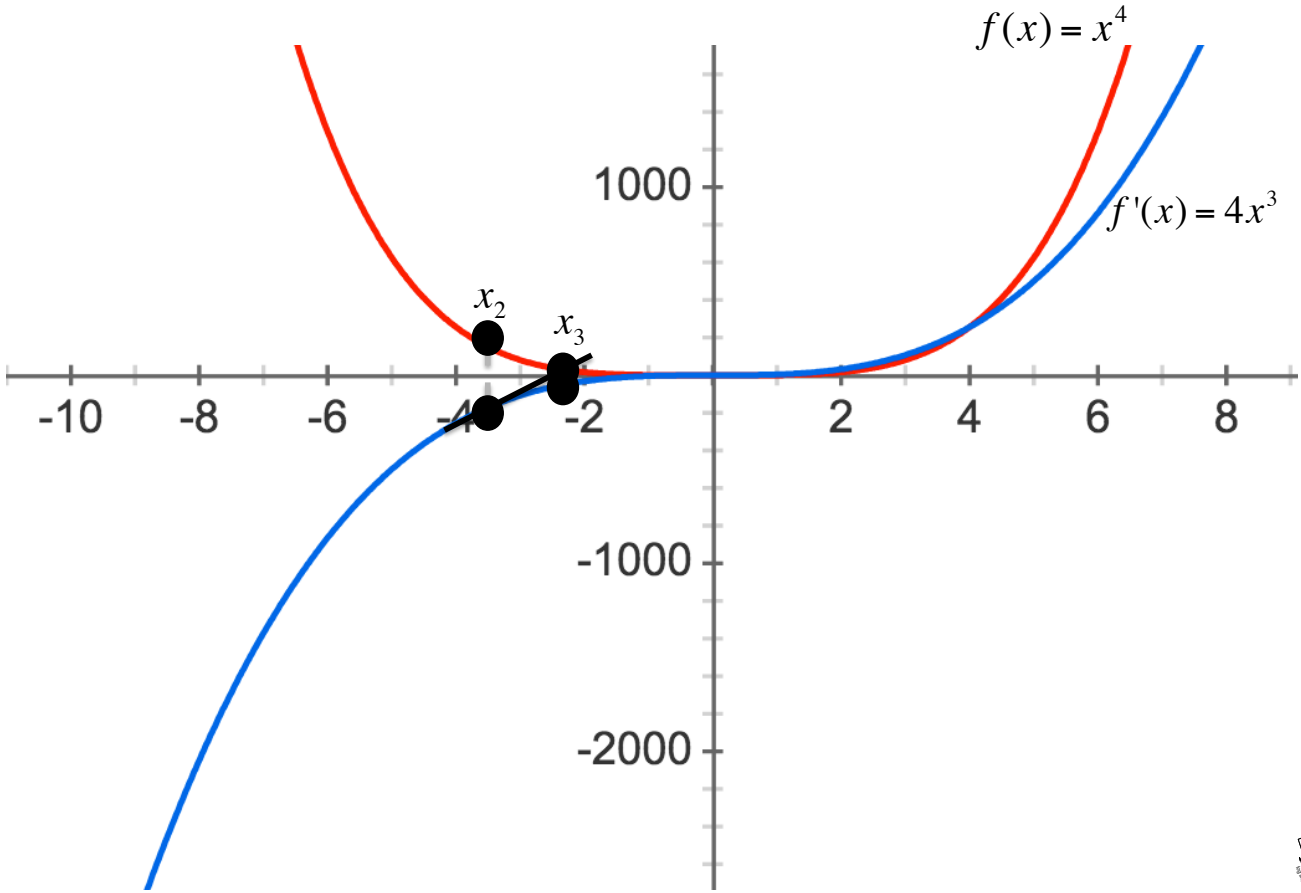
$f(x) = x^4$, $\nabla f(x) = 4x^3$,

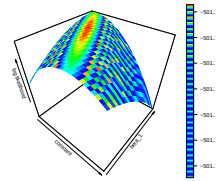




Newton-Raphson - Example

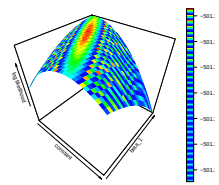
$f(x) = x^4$, $\nabla f(x) = 4x^3$,





Nelder-Mead method

- Proposed by John Nelder and Roger Mead in 1965
- Also known as *downhill simplex method*
- Method for unconstrained optimization problems in n-dimensional spaces
- Does not require derivatives of objective function (non gradient)
- Compare objective function on a set of parameters to identify optimal parameters



Nelder-Mead method

Problem

Minimize a function $f(x)$, $x \in \mathbb{R}^n$

$f \in C(\mathbb{R}^n, \mathbb{R})$, $n \geq 2$

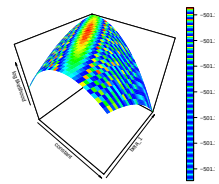
Simplex

A Simplex is a n dimensional polytope which is the convex hull of its $n+1$ vertices, more formally

$$\Delta = \{x^1, \cdots, x^{n+1}\} \subset \mathbb{R}^n$$

Examples

- A 0-Simplex is a point
- A 1-Simplex is a line
- A 2-Simplex is a triangle

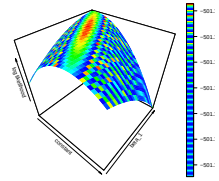


Nelder-Mead algorithm

Step 1 (ordering)

Choose an initial simplex and and evaluate the function at all x^i and identify indices $b, sw, w \in \{1, \ldots, n+1\}$, where

x^w	:=	$\operatorname{argmax}\{f(x) : x \in \Delta\}$	worst point
x^{sw}	:=	$\operatorname{argmax}\{f(x) : x \in \Delta, sw \neq w\}$	second worst point
x^b	:=	$\operatorname{argmin}\{f(x) : x \in \Delta, b \neq sw, w\}$	best point

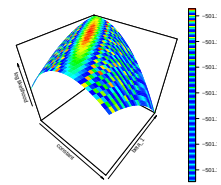


Nelder-Mead algorithm

Step 2 (centroid and reflection point)

Compute the centroid of the best n points: $z := \frac{1}{n} \sum_{i \neq w} x^i$

Compute a reflection point $r := z + \alpha(z - x^w)$, with $\alpha = 1$



Nelder-Mead algorithm

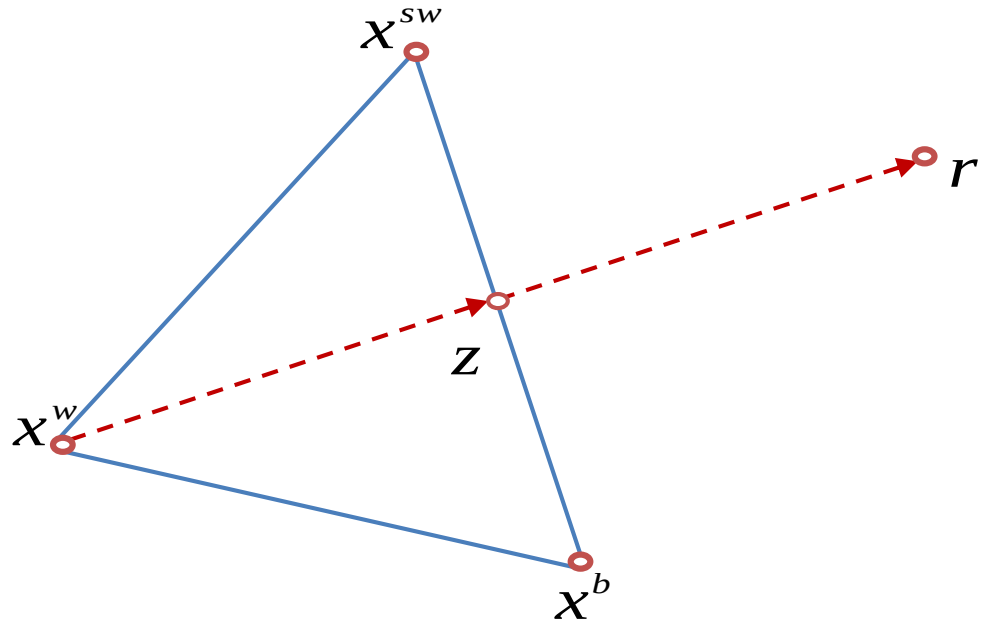
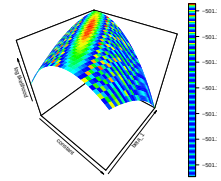
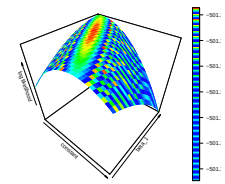


Figure 1: Reflection of worst point.



Nelder-Mead algorithm

Step 3 (reflection)
If $f(x^b) \leq f(r) \leq f(x^{sw})$, replace x^w by r



Nelder-Mead algorithm

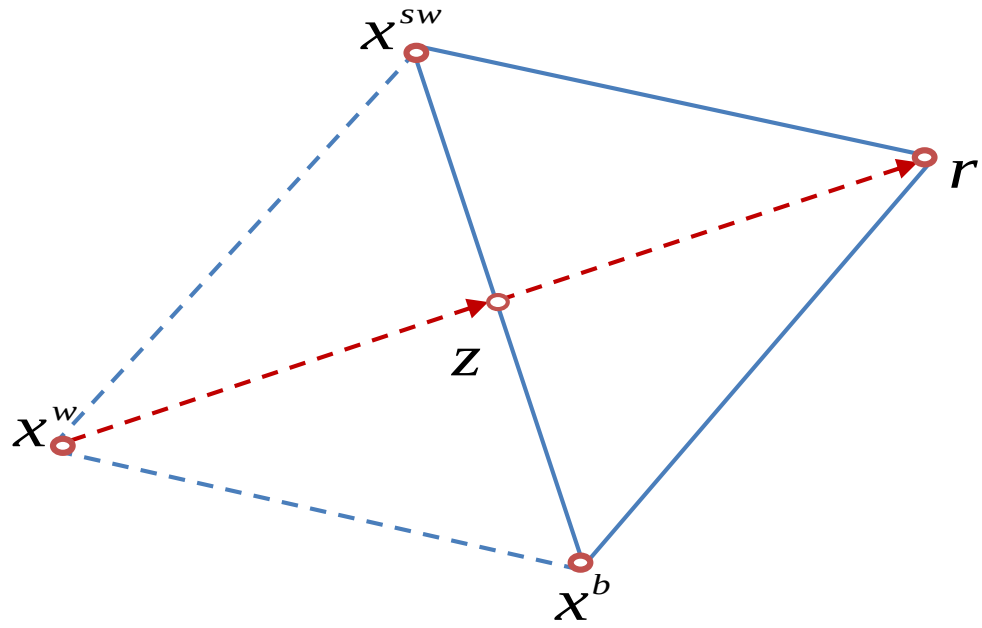
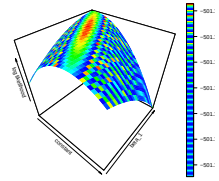
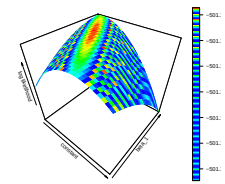


Figure 2: Simplex after replacing x^w by r.



Nelder-Mead algorithm

Step 4 (expansion)
If $f(r) < f(x^b)$, compute expansion point $e := z + \beta(z - x^w)$, with $\beta > \alpha$ (common practice: $\beta = 2$)
If $f(e) < f(r)$, replace x^w by e, else replace x^w by r



Nelder-Mead algorithm

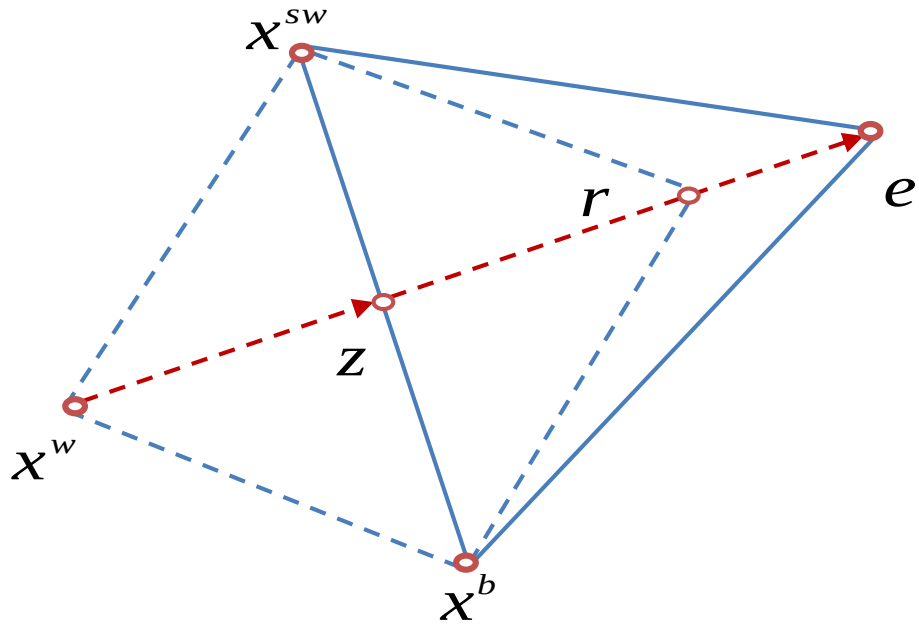
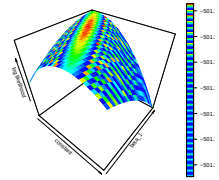


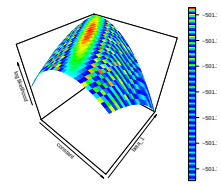
Figure 3: Simplex after replacing x^w by e.



Nelder-Mead algorithm

Step 5 (contraction)

- If $f(r) > f(x^{sw})$, compute contraction point k
  - If $f(r) > f(x^w)$ $k := z + \gamma(x^w - z)$
  - else $k := z + \gamma(z - x^w)$
  - with $0 < \gamma < \alpha$ (common practice: $\gamma = 0.5$)
- If $f(k) < f(x^w)$, replace x^w by k
- else replace contract simplex by x^b: $x^i := \frac{(x^i + x^b)}{2}$



Nelder-Mead algorithm

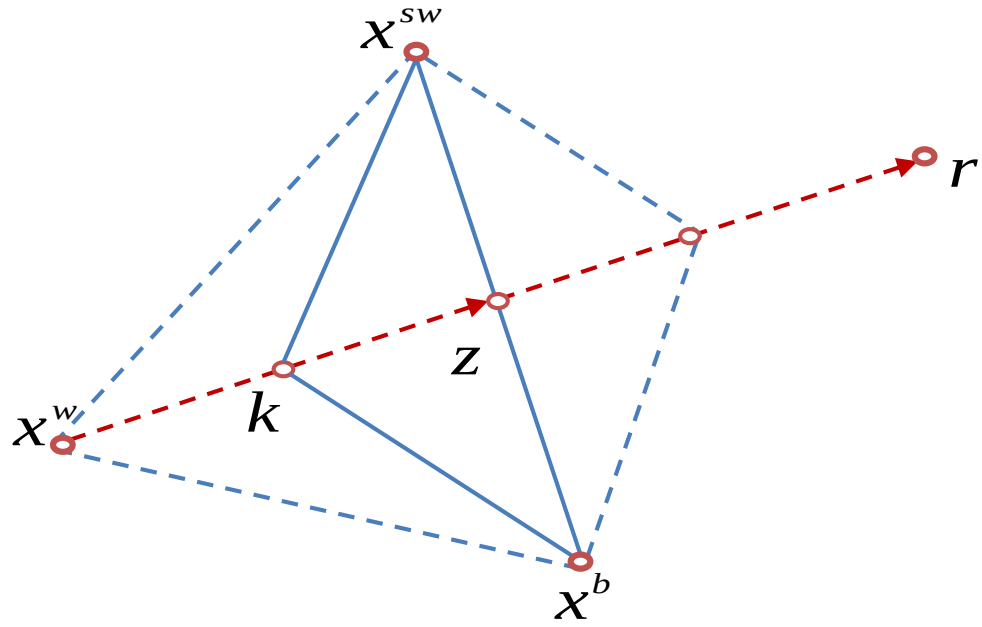
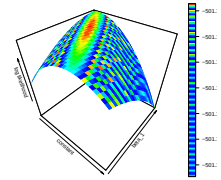


Figure 4: Simplex after replacing x^w by k.



Nelder-Mead algorithm

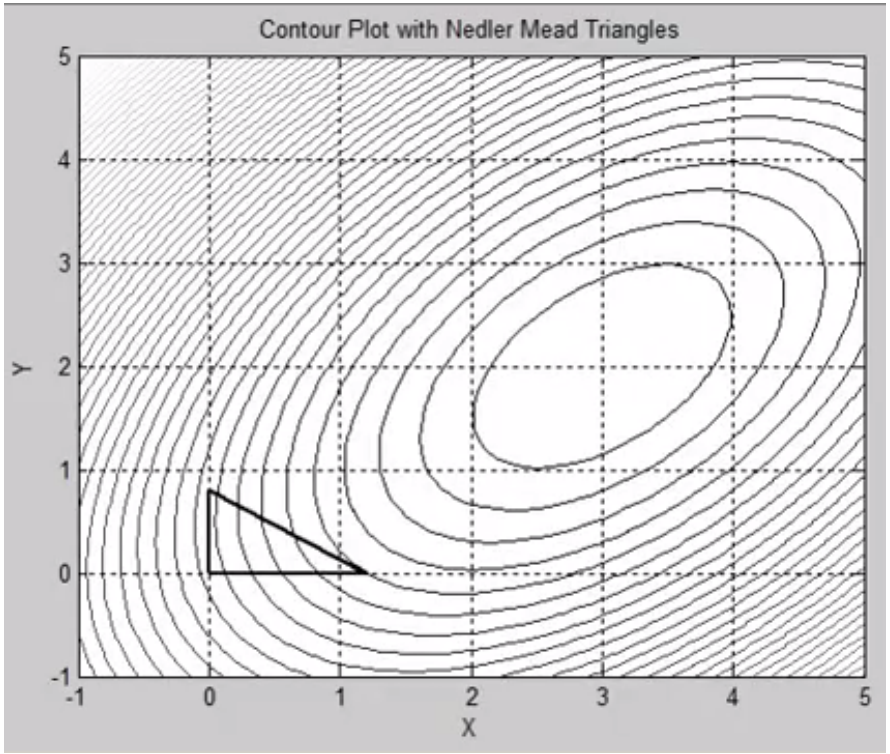
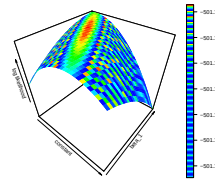


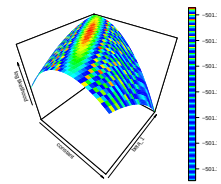
Figure 5: Visualization of Nelder-Mead approach.

Source: `https://www.youtube.com/watch?v=HUqLxHfxWqU`



Application of the algorithm to the data set Mroz.dta

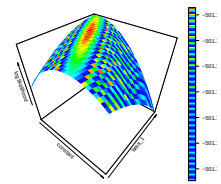
- Data from: Mroz (1987)
- Relationship of interest: Relationship between female labor force participation (inlf), education (educ) and the income of the husband (huswage).



Summary statistics

	inlf	educ	huswage
Description	labor force dummy	years of schooling	wage/h husband
Min.	0	5	0.412
1st Qu.	0	12	4.788
Median	1	12	6.976
Mean	0.568	12.290	7.482
3rd Qu.	1	13	9.167
Max.	1	17	40.510

Note: for better performance of the numerical methods, the data has been transformed by $x^* = \frac{x - min(x)}{max(x) - min(x)}$

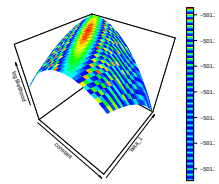


Application of the algorithm: The Logit Model

We model the relationship between the probability of being in the labor force and our explanatory variables as follows

$$P(infl_i = 1) = \Lambda(\beta_0 + \beta_1 educ_i + \beta_2 huswage_i) = \Lambda(x_i'\boldsymbol{\beta})$$

where $\Lambda(x)$ gives the logistic link function:
$\Lambda(x) = \exp(x)/(1 + \exp(x))$ and x_i is the vector of covariates for the observational unit i and $\boldsymbol{\beta} = (\beta_0 \ \beta_1 \ \beta_2)^T$



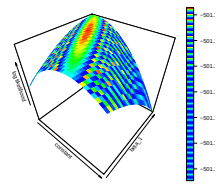
Application of the algorithm: The Logit Model

In order to maximize the likelihood function $L(\beta)$ w.r.t. to β we have to maximize the Likelihood-function

$$\mathrm{L}(\beta) = \prod_{i=1}^{N} \Lambda(x_i'\beta)^{y_i}\left[1 - \Lambda(x_i'\beta)\right]^{(1-y_i)}$$

Since ln(.) is a monotone transformation, this is equivalent to the maximization of the log-likelihood

$$\mathcal{L}(\beta) = \sum_{i=1}^{N}\left[y_i \ln\left[\Lambda(x_i'\beta)\right] + (1-y_i)\ln\left[1 - \Lambda(x_i'\beta)\right]\right]$$

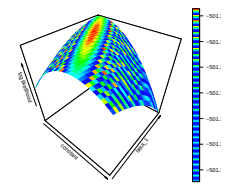


Application of the algorithm: The Logit Model

A maximum would be reached by setting the score $s(\beta)$ (gradient of $\mathcal{L}(\beta)$ w.r.t. to β) to zero:

$$s(\beta) = \sum_{i=1}^{N} (y_i - \Lambda(x_i'\beta)) x_i \stackrel{!}{=} 0$$

However, no closed form solution is available. Therefore we have to rely on numeric methods.



Visualization of a log-likelihood function

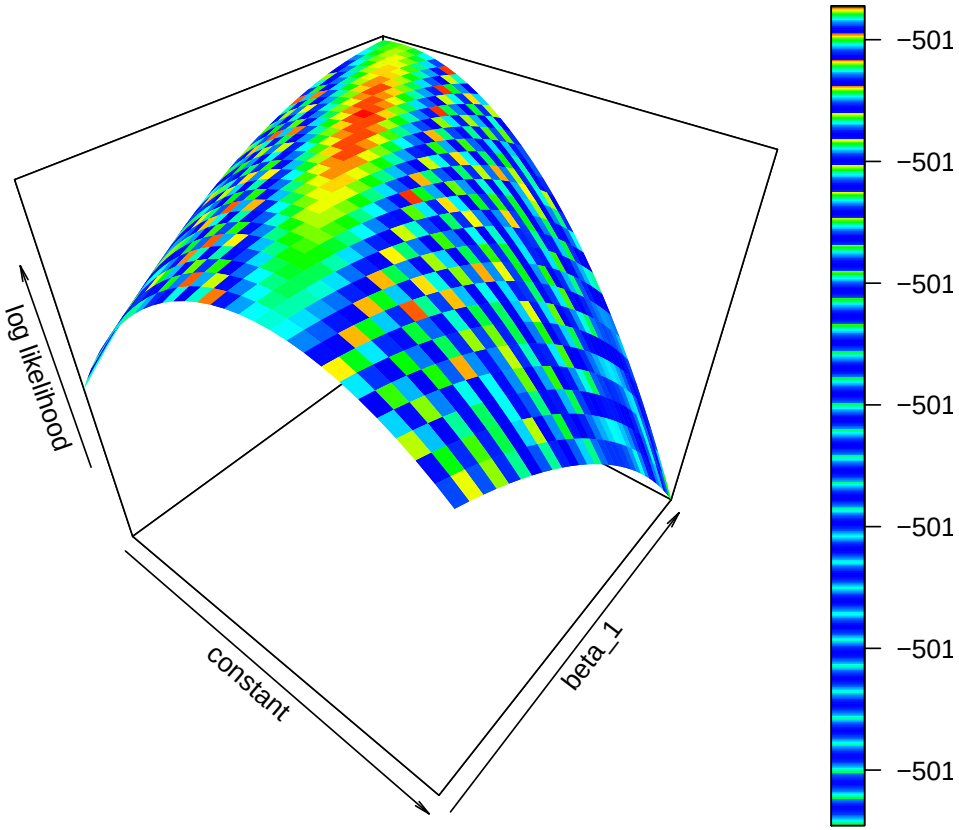


Figure 6: Visualization of a log-likelihood function.

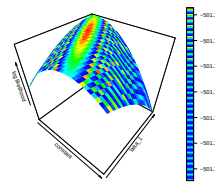
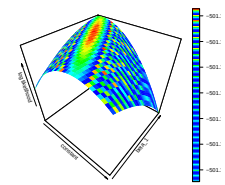
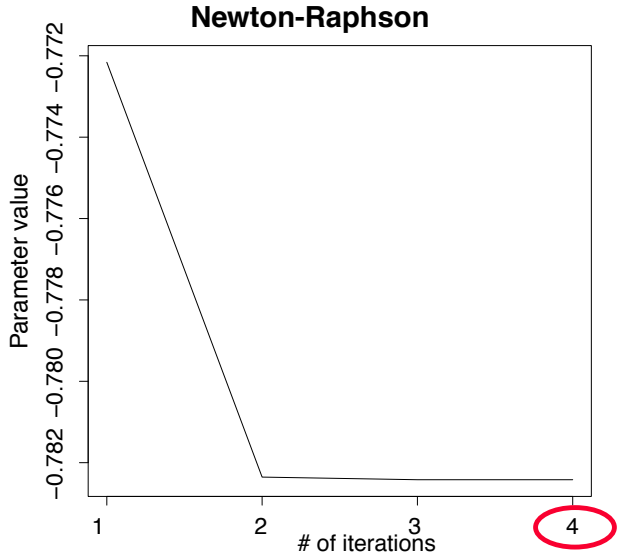


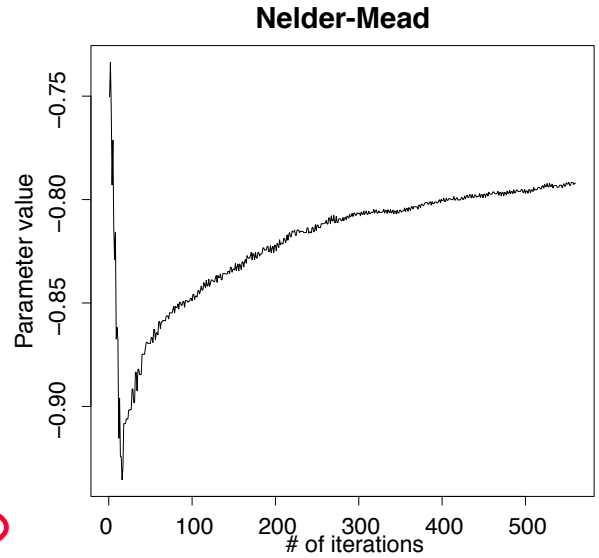
Table 1: Model Comparison

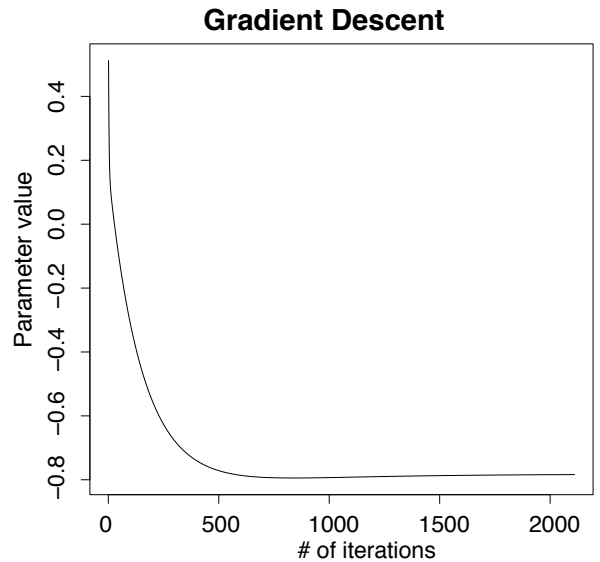
	β_0	β_1	β_2	Iterations	Log-Likelihood
`optim()`*	-0.783	2.571	-2.778	156	-494.798
Nelder-Mead	-0.783	2.569	-2.770	$1,575$	-494.798
Newton-R.	-0.782	2.570	-2.774	6	-494.798
Grad. des.**	-0.783	2.568	-2.769	$3,227$	-494.798
Grad. des.***	-0.782	2.570	-2.775	$1,191$	-494.798

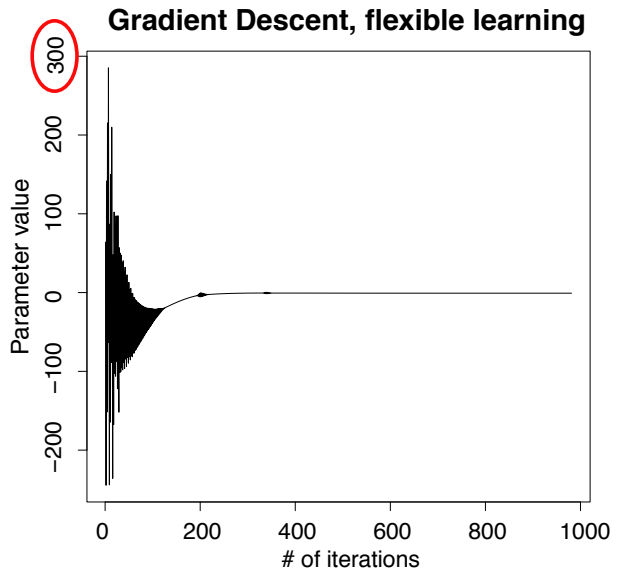
Note : * Nelder-Mead optimization, ** fixed learning parameter, *** flexible learning parameter

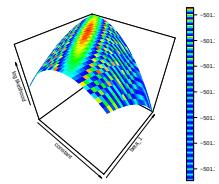



Newton-Raphson
Parameter value
−0.772
−0.774
−0.776
−0.778
−0.780
−0.782
1
2
3
4
# of iterations


Nelder-Mead
Parameter value
−0.75
−0.80
−0.85
−0.90
0
100
200
300
400
500
# of iterations

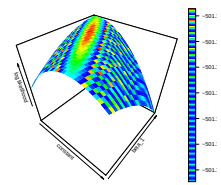

Gradient Descent
Parameter value
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0
500
1000
1500
2000
# of iterations


Gradient Descent, flexible learning
Parameter value
300
200
100
0
−100
−200
0
200
400
600
800
1000
# of iterations



Compare results

- `optim()` package performs best
- performance depend on information (gradient, hessian, only function)
- self implemented algorithms are able to identify almost the same parameter

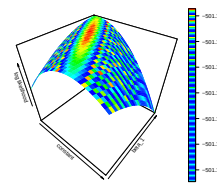


For further reading

Mroz, T.A. (1987)
The Sensitivity of an Empirical Model of Married Women's Hours of Work to Economic and Statistical Assumptions
Econometrica, 55, 765-799

Nelder, John A. and R. Mead (1965)
A simplex method for function minimization
Computer Journal 7, 308-313

Nocedal, Jorge, and Stephen Wright (2006)
Numerical optimization
Springer Science Business Media, 2006



For Further Reading

Winkelmann, Rainer, and Stefan Boes (2006)
Analysis of microdata
Springer Science Business Media, 2006