Computational statistics, lecture 1

Frank Miller, Department of Computer and Information Science, Linköping University

frank.miller@liu.se

October 31, 2023

Course schedule

- Lecture 1: **Unidimensional optimization, computer arithmetic**
- Math-lecture 1: Basic matrix algebra, analytical optimization, determinants
- Lecture 2: **Multidimensional optimization**
- Math-lecture 2: Density, cumulative distribution function, integration
- Lecture 3: **Random number generation**
- Lecture 4: **Monte Carlo methods**
- Lecture 5: **Model selection and hypothesis testing**
- Lecture 6: **EM algorithm, stochastic optimization**

Teaching group: Krzysztof Bartoszek, L5-L6; Frank Miller, examiner, L1-L4, LM1-LM2; Bayu Brahmantio, Héctor Rodriguez Déniz, teaching assistants

Course homepage: https://www.ida.liu.se/~732A90/index.en.shtml; includes schedule, reading material, lecture notes, assignments

Computer labs: For each lecture; exercises to hand-in in **groups of 2**

Evaluation of last course (HT2022)

- 23 students of around 50 submitted the evaluation; average grade 1.65
- Changes to HT2023:
  - Changes in content by deepening central topics (e.g. optimization) and instead removing some topics (e.g. reduced computer arithmetics); lab questions adjusted accordingly
  - Givens and Hoeting textbook slightly more used as basis for course layout (however, Gentle textbook covers topics as well)
  - New lecture slides for large parts of the course
  - Mathematical lectures moved inside the course (after L1 and after L2)

Computational statistics

- When large or huge datasets should be analyzed and/or complex models are used, **statistics depends on effective computational methods**
- We will **learn in this course several algorithms** for optimization, randomization, Monte Carlo integration and **methods to use them**

Today’s content

- Optimization
  - Why?
  - Analytic univariate optimisation
  - Bi-section, Newton, and secant methods (univariate)
  - On convergence speed

- Computer arithmetics

Optimization in statistics

- Maximum Likelihood
- Minimizing risk in (Bayesian) decision theory
- Minimizing sum of squares (Least Squares Estimate)
- Maximizing information in experimental design
- Machine learning

- Common problem in these examples:
  - $\boldsymbol{x}$ p-dimensional vector, $g\colon \mathbb{R}^p \to \mathbb{R}$ function
  - We search $\boldsymbol{x}^*$ with $g(\boldsymbol{x}^*) = \max g(\boldsymbol{x})$
- Minimization problem turns into maximization by considering $-g$

Least squares estimation (LSE)

- We search a Least Squares estimate $\widehat{\boldsymbol{\beta}}$ for $\boldsymbol{\beta}$ minimising the distance $g(\widehat{\boldsymbol{\beta}}) = \|\widehat{\boldsymbol{y}} - \boldsymbol{y}\|^2$ from $\widehat{\boldsymbol{y}} = \boldsymbol{X}\widehat{\boldsymbol{\beta}}$ to $\mathbf{y}=\mathbf{X}\,\boldsymbol{\beta} + \boldsymbol{\varepsilon}$
- $g(\widehat{\boldsymbol{\beta}}) = \left\|\boldsymbol{X}\widehat{\boldsymbol{\beta}} - \boldsymbol{y}\right\|^2 = \left(\boldsymbol{X}\widehat{\boldsymbol{\beta}} - \boldsymbol{y}\right)^T\left(\boldsymbol{X}\widehat{\boldsymbol{\beta}} - \boldsymbol{y}\right) = \widehat{\boldsymbol{\beta}}^T\boldsymbol{X}^T\boldsymbol{X}\widehat{\boldsymbol{\beta}} - \mathbf{2}\widehat{\boldsymbol{\beta}}^T\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{y}^{\boldsymbol{T}}\boldsymbol{y}$
- Setting the derivative to 0 ($\frac{\partial f}{\partial \widehat{\beta}} = 2\boldsymbol{X}^T\boldsymbol{X}\widehat{\boldsymbol{\beta}} - \mathbf{2}\boldsymbol{X}^T\boldsymbol{y} = 0$), we get $\widehat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$

- Optimization problem:
  - $\widehat{\boldsymbol{\beta}}$ p-dimensional vector, $g\colon\ \mathbb{R}^p \to \mathbb{R}$ function
  - We search $\widehat{\boldsymbol{\beta}}$ with $g(\widehat{\boldsymbol{\beta}}) = \min g(\boldsymbol{b})$
- Here, we do not need to iteratively compute this minimum since we have an algebraic solution $\widehat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$

Variations of least squares estimation

- Algebraic solution exists for the LSE, but not if we vary the problem
- Lasso estimate: $g(\widehat{\boldsymbol{\beta}}) = \left\|\boldsymbol{X}\widehat{\boldsymbol{\beta}} - \boldsymbol{y}\right\|^2 + \lambda\left\|\widehat{\boldsymbol{\beta}}\right\|_1$
- L_1-estimation: $g(\widehat{\boldsymbol{\beta}}) = \left\|\boldsymbol{X}\widehat{\boldsymbol{\beta}} - \boldsymbol{y}\right\|_1$
- Many further variations of estimates have been considered
- In all cases, we search $\widehat{\boldsymbol{\beta}}$ with $g(\widehat{\boldsymbol{\beta}}) = \min g(\boldsymbol{b})$

- Recall: Norms for $\boldsymbol{x} = (x_1, \dots, x_p)^T$: $\|\boldsymbol{x}\| = \|\boldsymbol{x}\|_2 = \sqrt{x_1^2 + \cdots + x_p^2}$ (Euclid), $\|\boldsymbol{x}\|_1 = |x_1| + \cdots + |x_p|$, $\|\boldsymbol{x}\|_\infty = \max\{|x_1|, \dots, |x_p|\}$ (max-norm)

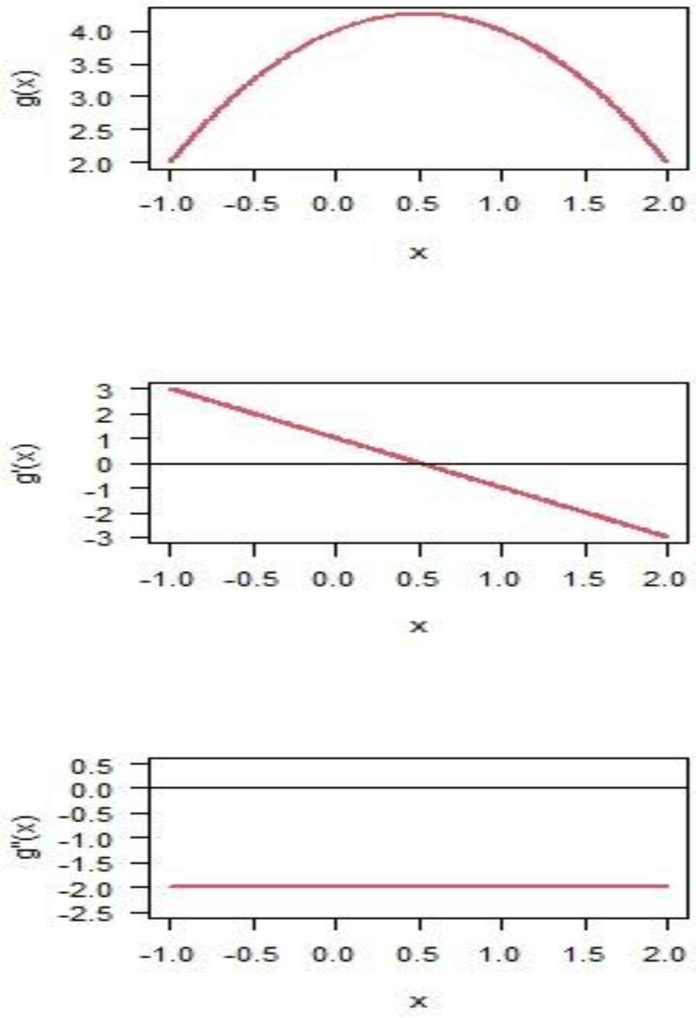
Univariate optimization

- x real number, $g\colon \mathbb{R} \to \mathbb{R}$ continuously differentiable function
- We search x^* with $g(x^*) = \max g(x)$

- Compute $g'(x)$ and search x^* with $g'(x^*) = 0$
- One has then to check if the result is maximum, minimum, possibly local optimum…

Univariate optimization – analytic solution

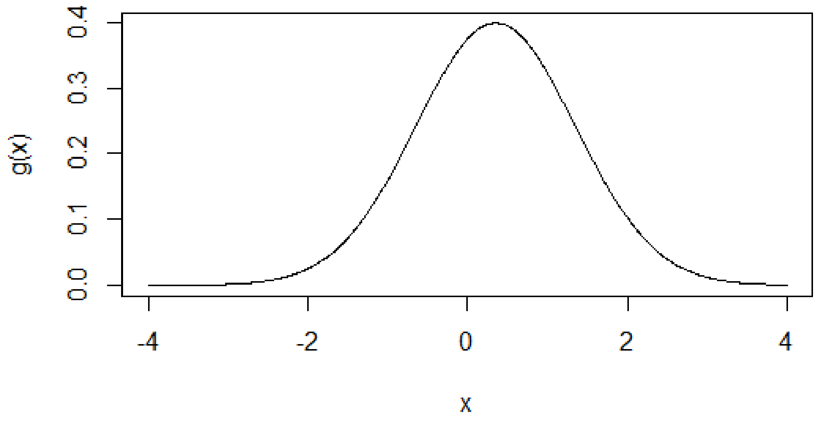
- Compute $g'(x)$ and search x^* with $g'(x^*) = 0$
- Example, where analytic optimization possible:
- $g(x) = 4 + x - x^2$
  $g'(x) = 1 - 2x$
  $g'(x) = 0$ if and only if $x = 1/2$
  $g''(x) = -2, \quad g''(1/2) = -2 < 0$
- Therefore, g has local maximum at $x = 1/2$

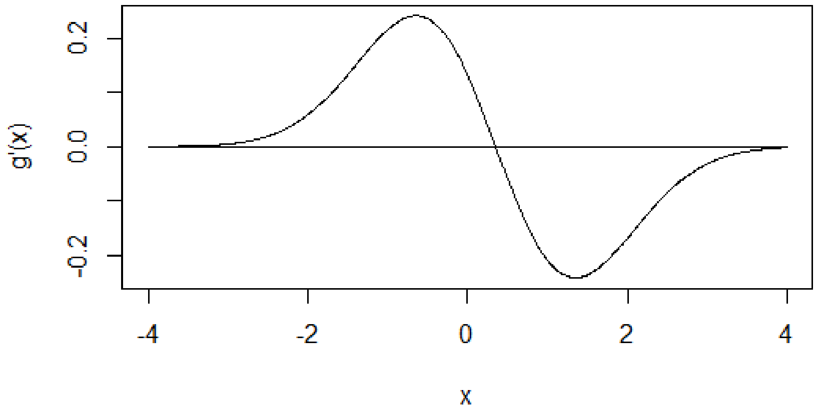


- Now, we have cases in mind were the **analytic solution** is **not possible**, and we need **iterative methods**

Univariate optimization: bisection

- Search x^* with $g'(x^*) = 0$:
  1) Start with interval $[a_0, b_0]$ such that $g'(a_0) \cdot g'(b_0) < 0, t = 0$
  2) Set $x^{(t)} = (a_t + b_t)/2$
  3) Define next interval $[a_t, b_t]$ by
     $[a_t, x^{(t)}]$ if $g'(a_t) \cdot g'\left(x^{(t)}\right) \leq 0$,
     $[x^{(t)}, b_t]$ if $g'\left(x^{(t)}\right) \cdot g'(b_t) < 0$
  4) Set t to t+1 and go to 2)
- See [video on course homepage]()
- **Iteratively** improve approximations for x^*:
  $x^{(0)}$ -> $x^{(1)}$ -> $x^{(2)}$ -> ...





Optimization: convergence criterion

- Compare $x^{(t)}$ and $x^{(t+1)}$ and stop if they are “close enough”
- Absolute convergence criterion:

$$\left|x^{(t+1)} - x^{(t)}\right| < \epsilon$$

- Relative convergence criterion:

$$\frac{\left|x^{(t+1)} - x^{(t)}\right|}{\left|x^{(t)}\right|} < \epsilon$$

Univariate Newton(-Raphson)

- x real number, $g: \mathbb{R} \to \mathbb{R}$ twice differentiable function
- Search x^* with $g(x^*) = \max g(x)$ by searching x^* with $g'(x^*) = 0$

- Taylor expansion around x^* motivates:

$$0 = g'(x^*) \approx g'\left(x^{(t)}\right) + \left(x^* - x^{(t)}\right) g''\left(x^{(t)}\right)$$

$$-\left(x^* - x^{(t)}\right) g''\left(x^{(t)}\right) \approx g'\left(x^{(t)}\right)$$

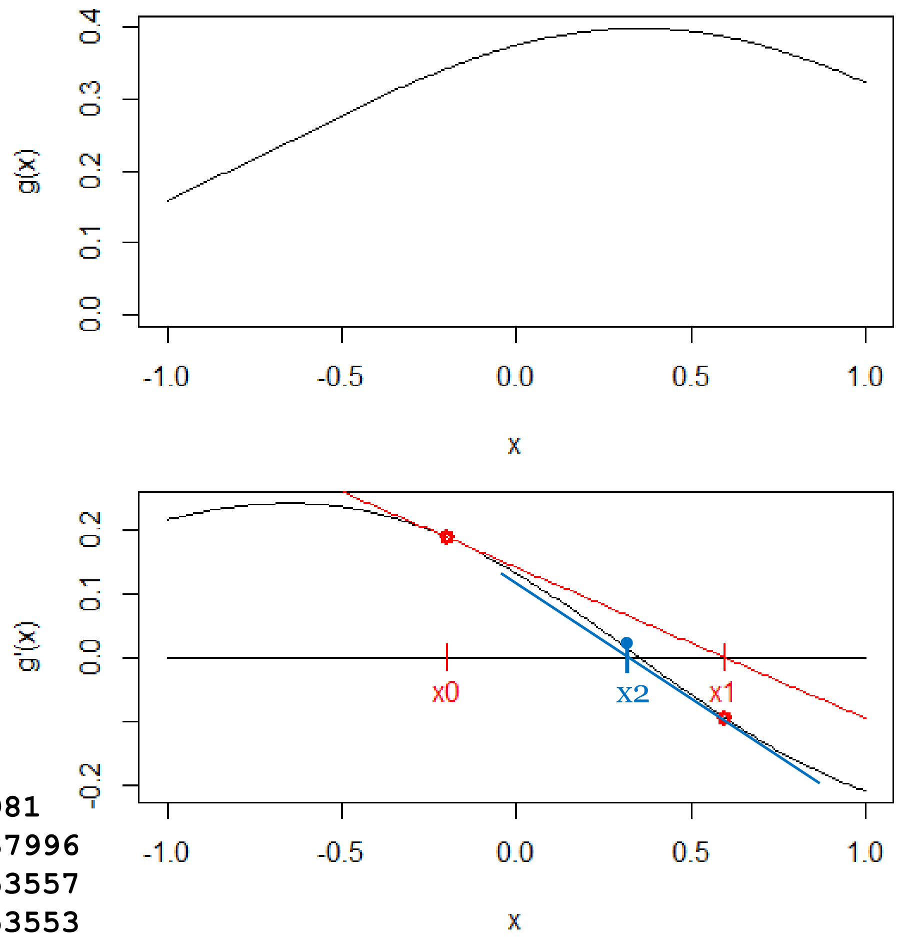
$$x^* \approx x^{(t)} - g'\left(x^{(t)}\right) / g''(x^{(t)})$$

- Therefore, the Newton-iteration works as:

$$x^{(t+1)} = x^{(t)} - g'\left(x^{(t)}\right) / g''(x^{(t)})$$

Univariate Newton(-Raphson)

- $x^{(t+1)} = x^{(t)} - g'(x^{(t)})/g''(x^{(t)})$
- Start with a $x^{(0)}$
- Tangent in $(x^{(0)}, g'(x^{(0)}))$ determines $x^{(1)}$
- Tangent in $(x^{(1)}, g'(x^{(1)}))$ determines $x^{(2)}$
- ...
- until convergence criterion met



+Newton method is fast

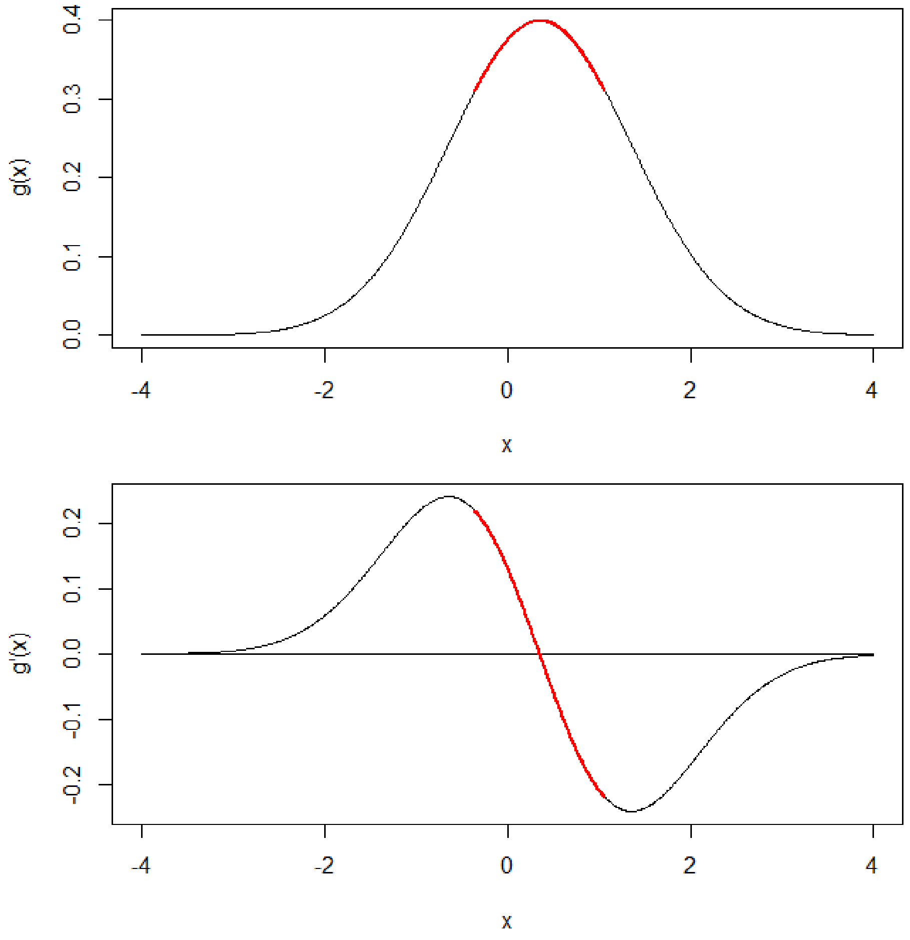
\- Requires existence and computation of g''

```
x0  -0.2
x1   0.5981
x2   0.337996
x3   0.353557
x4   0.353553
x5   0.353553
STOP
```

Univariate Newton(-Raphson)

- $x^{(t+1)} = x^{(t)} - g'\left(x^{(t)}\right)/g''(x^{(t)})$

- What about the starting value $x^{(0)}$?



Univariate secant method

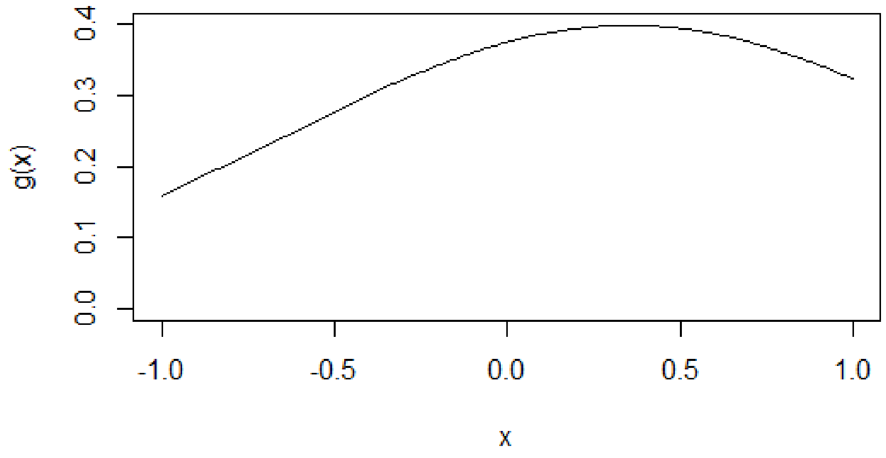
- x real number, $g: \mathbb{R} \to \mathbb{R}$ <u>once</u> differentiable function
- Search x^* with $g(x^*) = \max g(x)$ by searching x^* with $g'(x^*) = 0$
- Recall: The Newton-iteration works as:

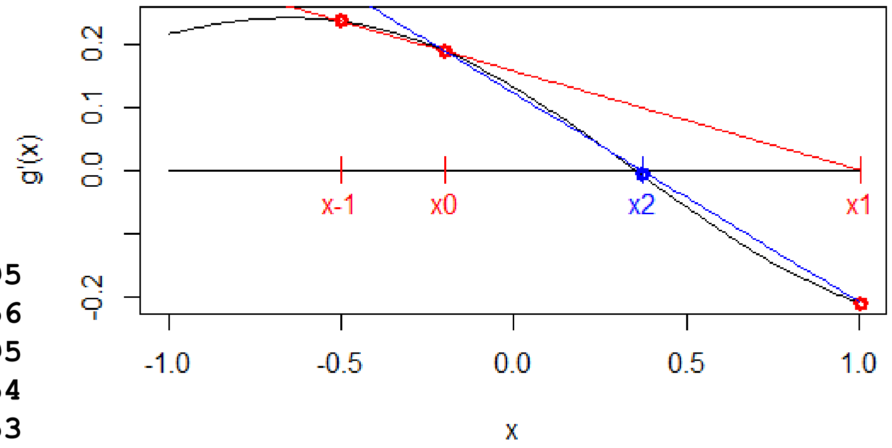
$$x^{(t+1)} = x^{(t)} - g'\left(x^{(t)}\right)/g''(x^{(t)})$$

- Need to compute g'' which might be difficult. Instead:
- Approximate $g''(x^{(t)})$ by $\left[g'\left(x^{(t)}\right) - g'(x^{(t-1)})\right]/\left(x^{(t)} - x^{(t-1)}\right)$

Univariate secant method

- $x^{(t+1)} = x^{(t)} - g'\left(x^{(t)}\right)\frac{x^{(t)}-x^{(t-1)}}{g'\left(x^{(t)}\right)-g'\left(x^{(t-1)}\right)}$
- Start with $x^{(0)}$ and $x^{(-1)}$
- Secant through $x^{(0)}$ and $x^{(-1)}$ determines $x^{(1)}$
- Secant through $x^{(1)}$ and $x^{(0)}$ determines $x^{(2)}$
- …
- until stopping crit. fulfilled





- Quite fast
- No 2^{nd} derivative necessary

```
x0   -0.2
x1    1.006995
x2    0.371656
x3    0.349095
x4    0.353554
x5    0.353553
x6    0.353553
STOP
```

Convergence speed of optimization algorithms

Convergence order

- Convergence speed can be quantified by q and c as follows:

Convergence rate

  - Let $\varepsilon^{(t)} = x^{(t)} - x^*$,
  - Find q and c such that $\lim_{t\to\infty} \varepsilon^{(t+1)}/(\varepsilon^{(t)})^q = c$

Intuitively, $\varepsilon^{(t+1)} \approx c \cdot (\varepsilon^{(t)})^q$

- ε = 1, 0.5, 0.25, 0.125, 0.063, 0.031, … ➡ q=1, c=0.5,
- ε = 1, 0.1, 0.01, 0.001, 0.0001, … ➡ q=1, c=0.1,
- If q=1, we say that convergence is ”linear”
- ε = 1, 0.5, 0.125, 0.008, 0.00003, … ➡ q=2, c=0.5.
- If q=2, we say that convergence is ”quadratic”

Determine empirically convergence rate (and order) of optimization algorithms

- You have a given optimization algorithm and you have determined or know the maximiser $\boldsymbol{x}^*$. To check convergence speed in an optimization-run, you can calculate

$$D^{(t)} = \frac{\left|x^{(t)} - x^*\right|}{\left|x^{(t-1)} - x^*\right|}$$

(see Givens and Hoeting, 2013, page 101/102, for an example)

- If $D^{(t)} \to 1$, there is not even linear convergence (bad, order q<1),
  If $D^{(t)} \to c \in (0,1)$, linear convergence (order q=1) with rate c,
  If $D^{(t)} \to 0$, better than linear convergence (order q>1).

Comparison of univariate optimization methods

Bisection	Secant	Newton
g' required	g' required	g'' required
finds always an optimum between a_0 and b_0 (but could be local)	converges only when the two starting values ”close” to optimum	converges only when starting value ”close” to optimum
slow $q=1$	$q = \frac{1+\sqrt{5}}{2} = 1.62$	fast $q=2$

- There are also algorithms not needing g'
- **`R`**-function **`optimize`** uses such an algorithm (q=1.324)

Computer arithmetics

- Numbers are represented as binary numbers ($17 = 1 * 2^4 + 1 * 2^1 = "1001"$)
- Rational numbers are also represented based on the binary system:

$$\pm 0.d_1 d_2 \dots d_p * 2^e,$$
$$e = \pm e_1 e_2 \dots e_q$$

- E.g. $p = 52, q = 10$, two signs ⇒ one number needs 64 bits in the computer
- Limits in representation depending on p and q

```
> 2^1023
[1] 8.988466e+307
> 2^1024
[1] Inf
```

(overflow)

```
> 2^-1074
[1] 4.940656e-324
> 2^-1075
[1] 0
```

(underflow)

```
> 3/5-2/5-1/5
[1] -5.551115e-17
> if (3/5-2/5==1/5)
>   print("yes") else
>   print("no")
[1] "no"
```

Computer arithmetics

- Good to have limitations of computer arithmetics in mind!
- Example: Binomial coefficient (avoid overflow)

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{(k+1)(k+2)\cdots(n-1)n}{(n-k)!}$$

$$\binom{200}{2} = \frac{200!}{2!198!} = \frac{3*4*\cdots*199*200}{1*2*\cdots*197*198} = \frac{199*200}{1*2} = 19900$$

```
> n <- 200
> k <- 2
> prod(1:n) / (prod(1:k)*prod(1:(n-k)))
[1] NaN
> prod(((k+1):n) / (1:(n-k)))
[1] 19900
```

Course material, lab, seminar, exam

- Homepage: https://www.ida.liu.se/~732A90/index.en.shtml
  - Lecture notes, lab- and seminar info, exam info
- Submission of 6 labs via LISAM – all need to be passed – groups of 2
  - First lab: Oct 31 to Nov 7
- Mandatory attendance at 3 seminars and 1 presentation or opposition
- Computer exam: Jan 9, 2024. Own document of 100 pages can be used.
- 10 points to pass (E); 12 or more: D; ≥14: C; ≥16: B; ≥18: A

Literature:

- Gentle JE (2009). *Computational Statistics*, Springer
- Givens GH, Hoeting JA (2013). *Computational Statistics*, Wiley