Computational Numerical Statistics
PROJECT 2

Group G1 – TP2

Ana Mendes, N 57144
Emanuele Vivoli N 57284
Joao Camacho N 56861
Lia Schmid N 57629

Simao Goncalves N 54896

First Semester 2019–2020

FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE NOVA DE LISBOA

References

Project task 1

Random variable generation

Let $X \sim Beta(\alpha, 1)$, which has p.d.f.

$$f(x;\alpha) = \frac{x^{\alpha-1}}{B(\alpha,1)}, \alpha > 0, x \in [0,1]$$

Let

```
0.5409477 0.8184872 0.7848854 0.9850439 0.8963032 0.6089008
0.9549606 0.6795304 0.8451902 0.5613979 0.4029634 0.2741569
0.3996693 0.6371445 0.7521881
```

be an observed sample from X.

Project task 1

Likelihood, log-likelihood and score functions

- Likelihood function:

$$L(\alpha) = \prod_{i=1}^{n} f(x_i|\alpha) = \alpha^n \prod_{i=1}^{n} x_i^{\alpha-1}$$

- Log-likelihood function:

$$l(\alpha) = logL(\alpha) = nlog(\alpha) + (\alpha - 1) \sum_{i=1}^{n} log(x_i)$$

- Score function:

$$s(\alpha) = l'(\alpha) = \frac{n}{\alpha} - \sum_{i=1}^{n} log(x_i)$$

Project task 1

Maximum likelihood estimation of α

The MLE of α is obtained by maximizing L(α)

1. Solve the equation $s(\alpha) = 0$

$$s(\alpha) = 0 \Leftrightarrow \frac{n}{\alpha} + \sum_{i=1}^{n} log(x_i) = 0 \Leftrightarrow$$

$$\Leftrightarrow \frac{n}{\alpha} = -\sum_{i=1}^{n} log(x_i) \Leftrightarrow \alpha = -\frac{n}{\sum_{i=1}^{n} log(x_i)}$$

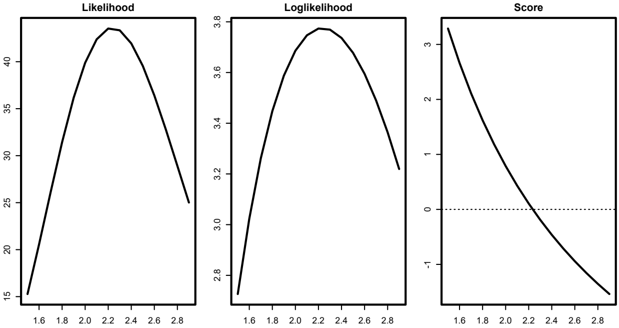
2. Confirm that $s'(\alpha) < 0$

$$s'(\alpha; x) = l''(\alpha) = \left(\frac{n}{\alpha} - \sum_{i=1}^{n} log(x_i)\right)' = -\frac{n}{\alpha^2} < 0$$

- The maximum likelihood estimator of α is $\hat{\alpha} = -\frac{n}{\sum_{i=1}^{n} log(x_i)}$
- Calculate MLE for sample from , $\hat{\alpha} = 2.235083$

Project task 1

Maximum likelihood estimation of α



Project task 1

Maximum likelihood estimation of α

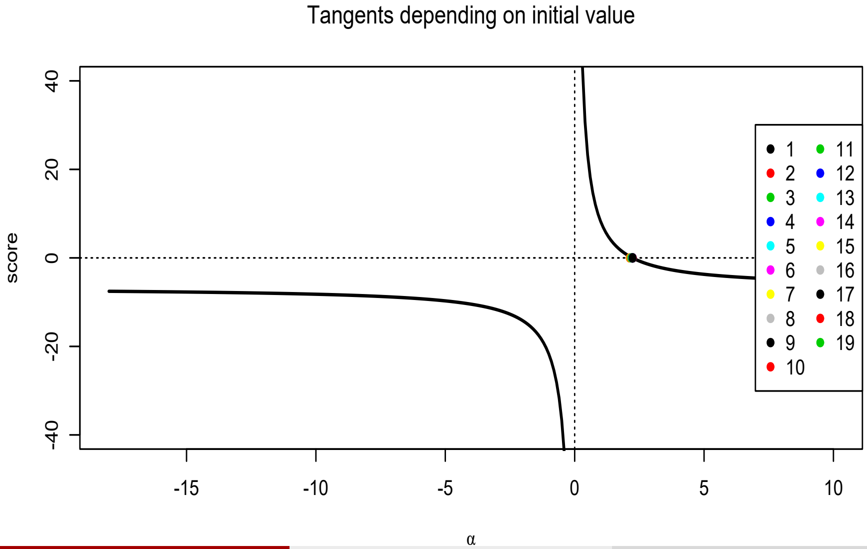
- Approximation of the ML estimate α using the R function maxLik()

```
maxLik(logLik=loglik.b,start=mme.graph.b)

maxLik(logLik=loglik, start=mme.graphical)
#Maximum Likelihood estimation
#Newton-Raphson maximisation, 3 iterations
#Return code 1: gradient close to zero
#Log-Likelihood: 3.775335 (1 free parameter)
#Estimate(s): 2.235083
```

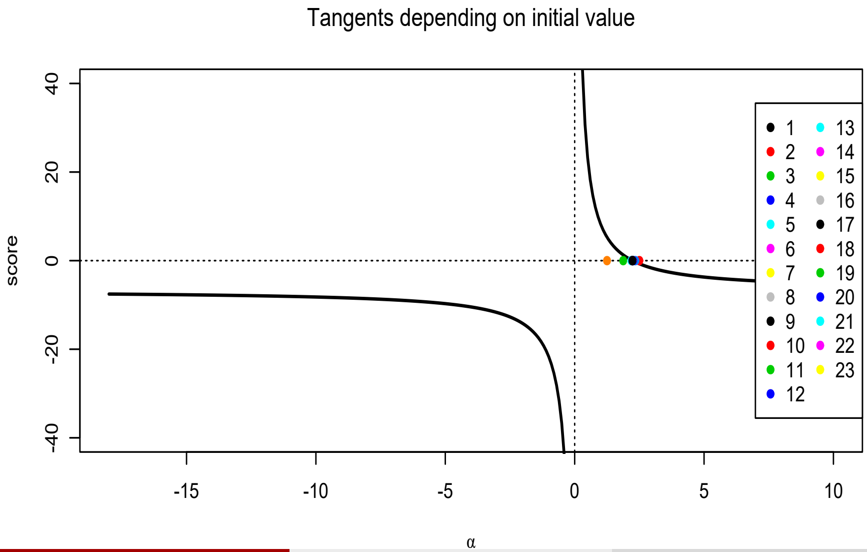
Project task 1

Bisection method, a = 2, b = 2.5



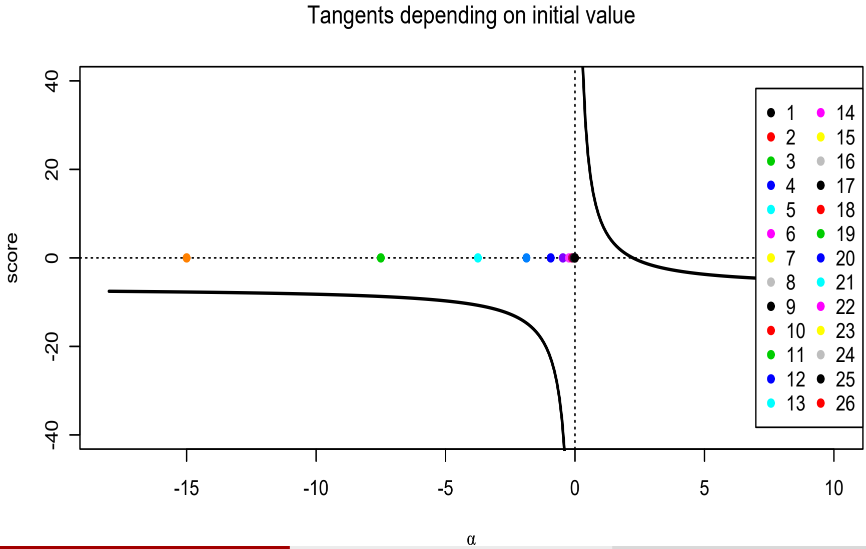
Project task 1

Bisection method, a = epsilon, b = 5



Project task 1

Bisection method, a = -30, b = 30



Theorem (Bolzano theorem)

Let f be a continuous function in the limited interval [a, b] $\in \mathbb{R}$ *such that:*

$$f(a)f(b) \leq 0$$

then f has at least one root $x^* \in\,]a, b[$.

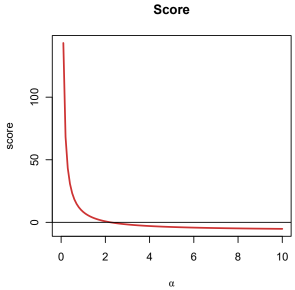
Project task 1

Bisection method

We can observe that s function is strictly descending, because

$$s'(\alpha; x) = -\frac{n}{\alpha^2} < 0$$

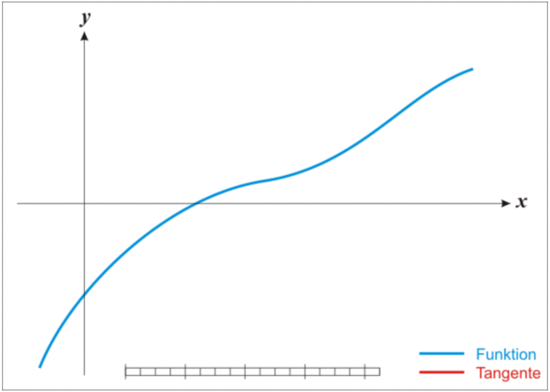
So, the function has one root in $\mathbb{R}^+$.



Therefore, we can conclude, whatever range that verifies the Bolzano theorem, this method will always converge.

Project task 1

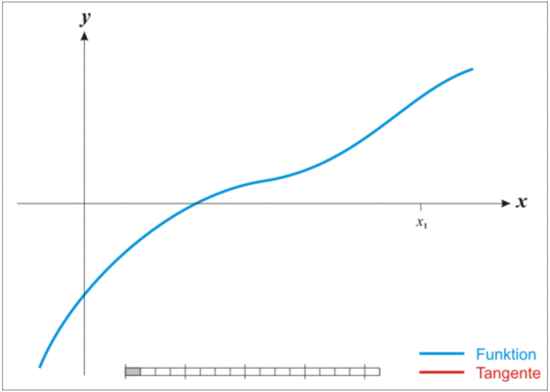
Random variable generation



INSERT Source of the image.

Project task 1

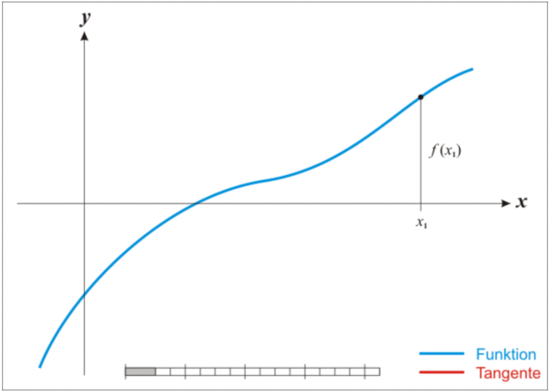
Random variable generation



INSERT Source of the image.

Project task 1

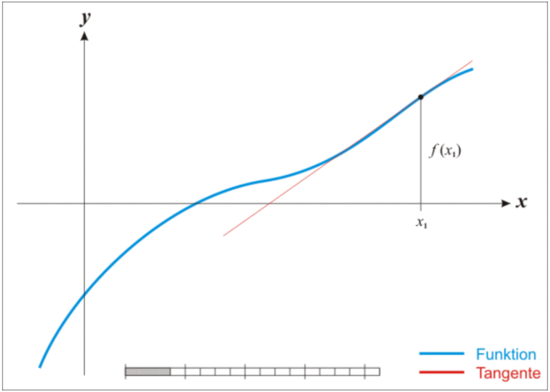
Random variable generation



INSERT Source of the image.

Project task 1

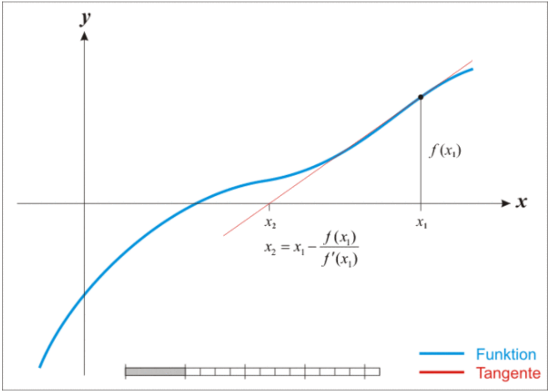
Random variable generation



INSERT Source of the image.

Project task 1

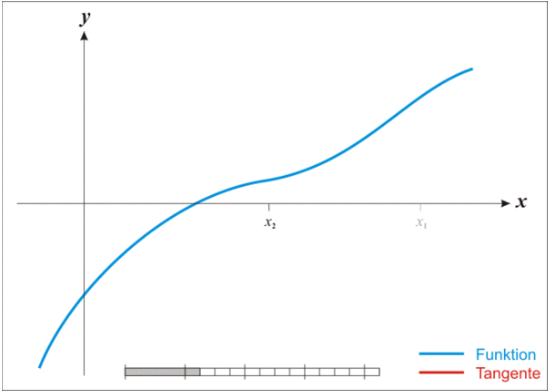
Random variable generation



INSERT Source of the image.

Project task 1

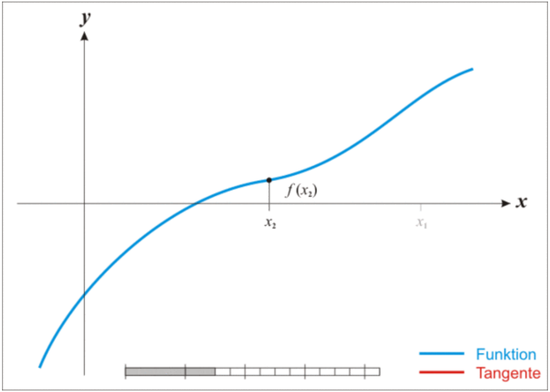
Random variable generation



INSERT Source of the image.

Project task 1

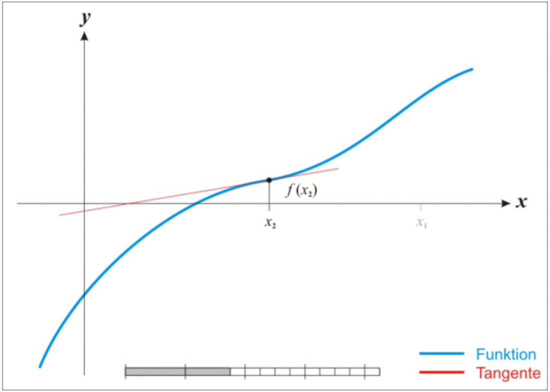
Random variable generation



INSERT Source of the image.

Project task 1

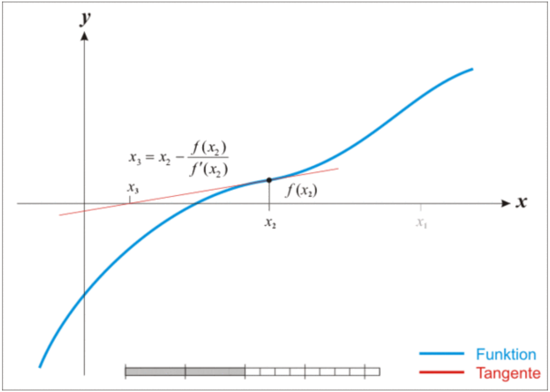
Random variable generation



INSERT Source of the image.

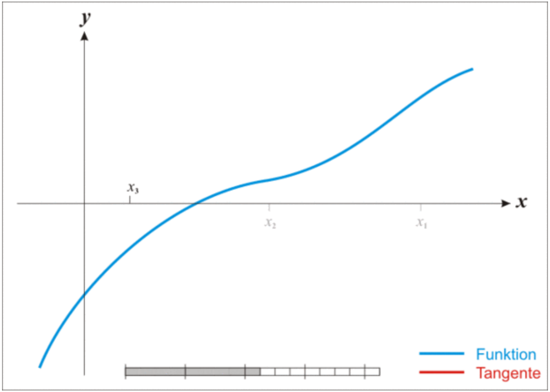
Project task 1

Random variable generation



INSERT Source of the image.

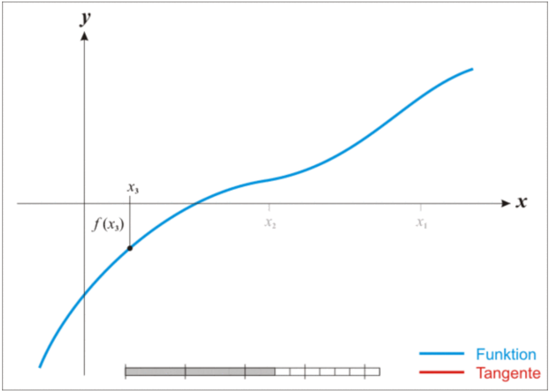
Project task 1
Random variable generation



INSERT Source of the image.

Project task 1

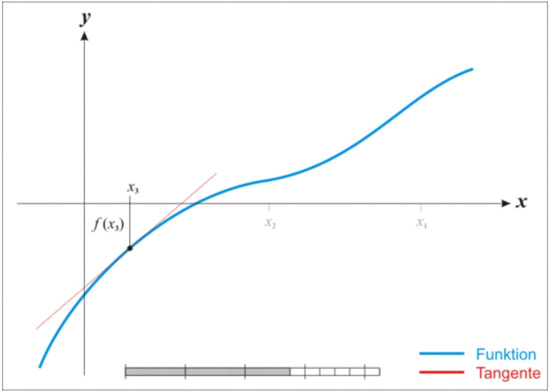
Random variable generation



INSERT Source of the image.

Project task 1

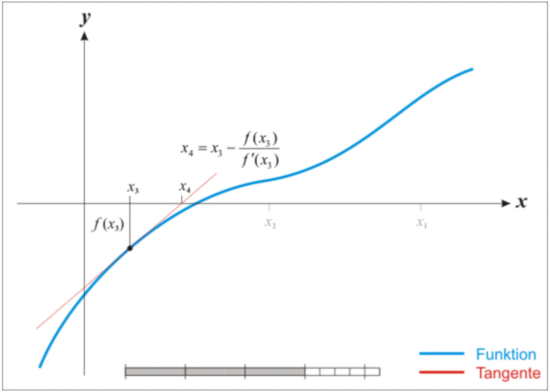
Random variable generation



INSERT Source of the image.

Project task 1

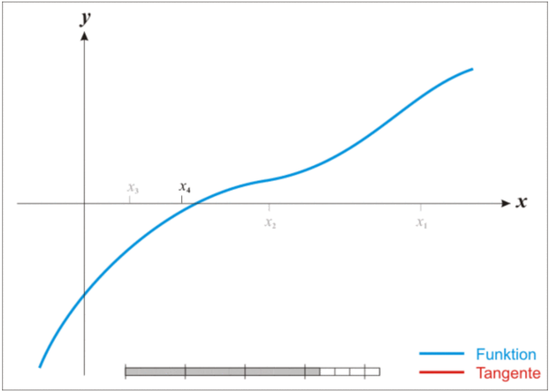
Random variable generation



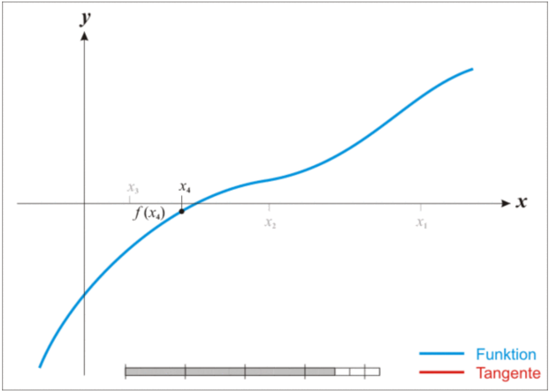
INSERT Source of the image.

Project task 1

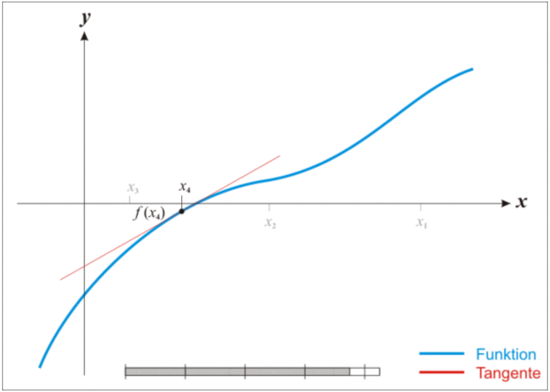
Random variable generation



INSERT Source of the image.



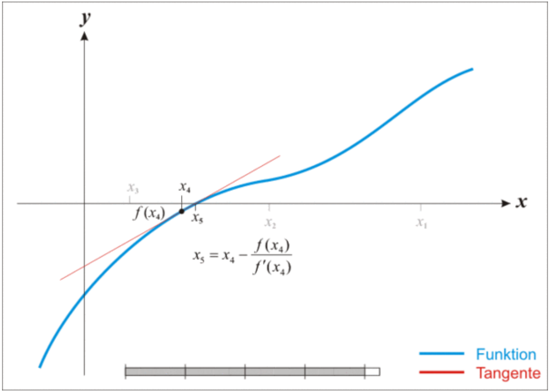
INSERT Source of the image.



INSERT Source of the image.

Project task 1

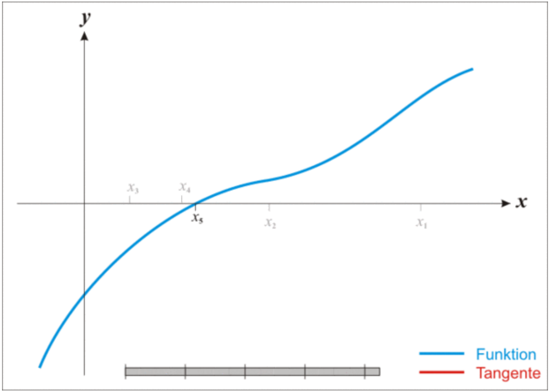
Random variable generation



INSERT Source of the image.

Project task 1

Random variable generation



INSERT Source of the image.

- Fisher Scoring Algorithm:

$$x_{n+1} = x_n + \frac{s(x_n)}{\mathcal{I}_n} = x_n + \frac{s(x_n)}{\frac{n}{\alpha^2}}$$

Where,

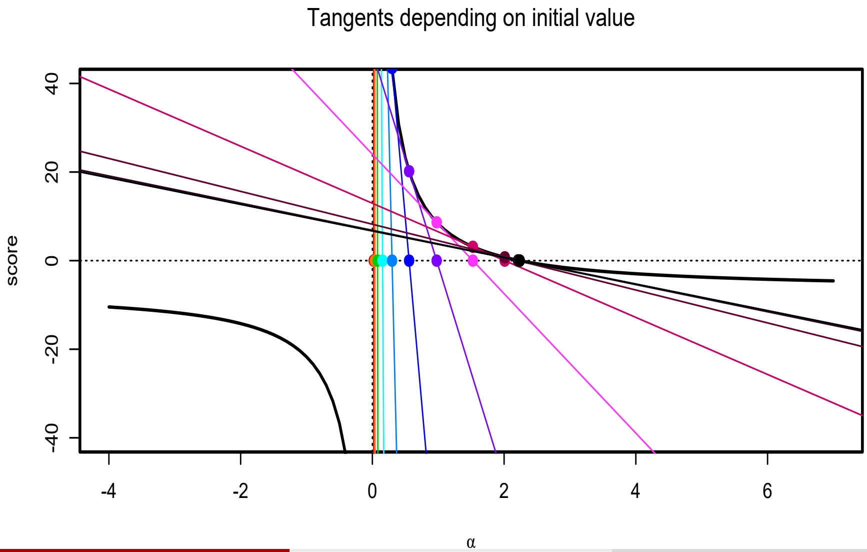
$$\mathcal{I}(\alpha) = -E\left[s'(\alpha; x)\right] = -E\left[-\frac{n}{\alpha^2}\right] = \frac{n}{\alpha^2}$$

- Newton Raphson Method:

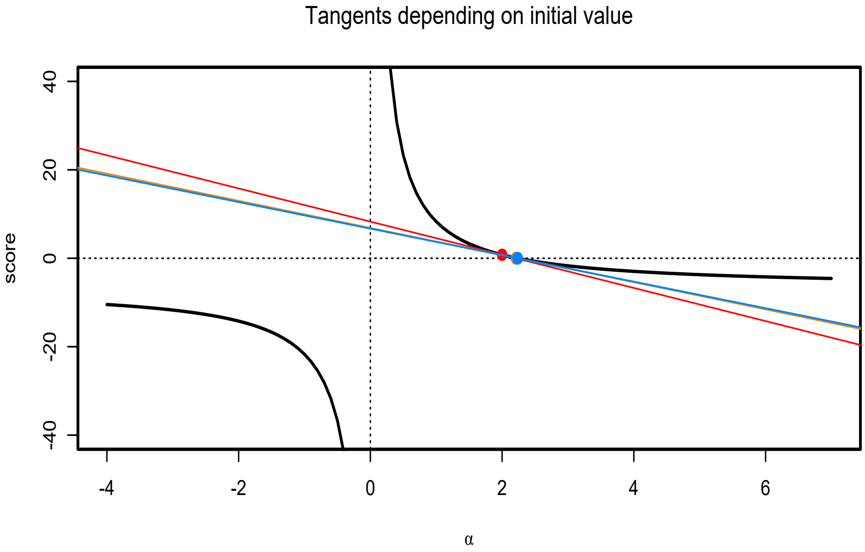
$$x_{n+1} = x_n - \frac{s(x_n)}{s'(x_n)} = x_n - \frac{s(x_n)}{-\frac{n}{\alpha^2}} = x_n + \frac{s(x_n)}{\frac{n}{\alpha^2}} \tag{1}$$

$\alpha \approx \epsilon$

Remember: $\alpha > 0$

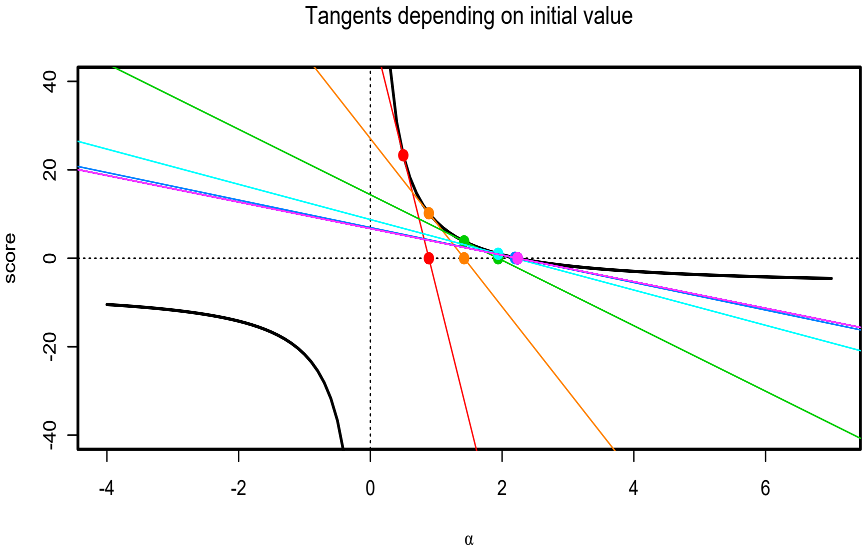


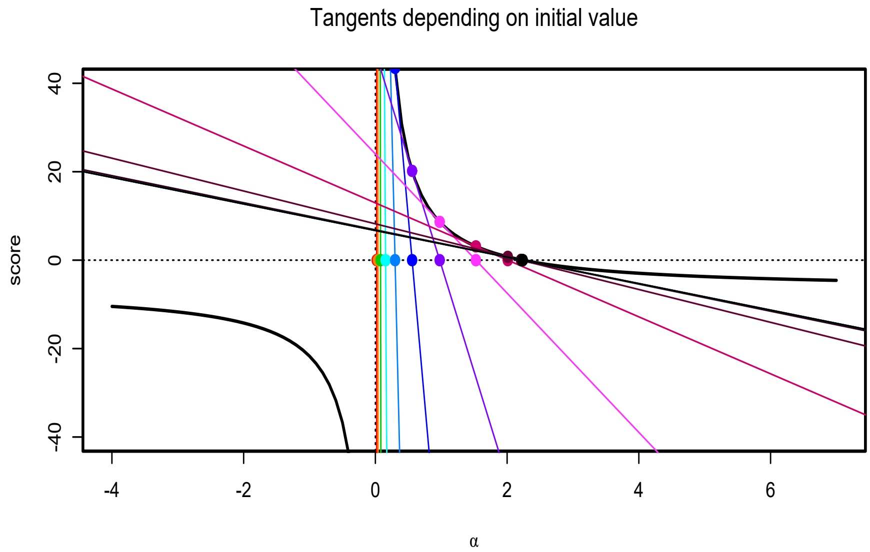
alpha = 2



Tangents depending on initial value
score
40
20
0
-20
-40
-4
-2
0
2
4
6
α

alpha = 0.5


Tangents depending on initial value
score
α
40
20
0
-20
-40
-4
-2
0
2
4
6



Click on picture to play the movie

Project task 1

Random variable generation

1. Let X be a discrete random variable with probability mass function (p.m.f.) f proportional to $g(x) = 1 - x + x^2$, $x = 0, 1, 2, 3, 4$.
   a) Identify the p.m.f. f
   b) Derive the cumulative distribution function (c.d.f.) F of X.
   c) Describe and implement the **inverse-transform** method in `R` for generating a sample from f. Call your routine `sim.itm()` and let it receive as input a generic sample size m. Provide both algorithm and **R** code. Finally, use `sim.itm()` to generate a sample of size $m = 10000$ of f.
   d) The same as in (c) but now using the **acceptance-rejection** method. Call you new simulation routine `sim.arm()`. Compute the rejection rate.
   e) Plot the discrete histograms from (c) and (d) with the true p.m.f. superimposed.
   f) Display the the hit-and-miss plot referring to `sim.arm(10)`.

Project task 1

Random variable generation

present problem solving here

```
paste you R code inside boxes like this one, e.g.,

# function that computes the mean of a trimmed sample
  trimmean=function(x,p){
          n=length(x); x=sort(x); k=n*p/100
          trimmean=sum(x[(k+1):(n-k)])/(n-2*k)
          return(trimmean)
  }
```

see template_report.tex for other tips

include your plots centered in the slide

logo_compridoFCTUNL.png