Numerical Methods
5633

Lecture 3

Michaelmas Term 2018

Marina Krstic Marinkovic
mmarina@maths.tcd.ie

School of Mathematics
Trinity College Dublin

Root Finding

- Finding an argument of a function f(x) that makes y=f(x) zero
- We seek the value α, such that

$$f(\alpha)=0$$

- α - zero of the function f(x)
- α - root of the equation f(x)=0

- f(x) may be a scalar, or a vector-valued function of a vector-valued variable --> solving system of equations

  - Bisection Method
  - Newton's method
  - Secant Method

Bisection Method

1. Given an initial interval $[a_0,b_0]=[a,b]$, set $k=0$
2. Compute $c_{k+1}=a_k+(b_k-a_k)/2$
3. If $f(c_{k+1})f(a_k)<0$ then set $a_{k+1}=a_k$, $b_{k+1}=c_{k+1}$
4. If $f(c_{k+1})f(b_k)<0$ then set $a_{k+1}=c_{k+1}$, $b_{k+1}=b_k$
5. Update k and go to Step 2.

- ➡ Each step is decreasing an upper bound on the absolute error by a factor of 2
- ➡ Programming hint: For numerical stability, we want to replace (a+b)/2 with a+(b-a)/2. This is because large values of a,b may lead to the computational overflow in (a+b)/2.

Bisection Convergence and Error Theorem

Let **$[a_0,b_0]=[a,b]$** be the initial interval, with **$f(a)f(b)<0$.** If we define an approximate root as **$X_n=c_n=(a_{n-1} + b_{n-1})/2$,** then there exists a root $\alpha \in [a,b]$ s.t.

$$|\alpha - x_n| \leq \frac{1}{2^n}(b-a)$$

Moreover, to achieve accuracy of

$$|\alpha - x_n| \leq \epsilon$$

it suffices to take

$$n \geq \frac{\log(b-a) - \log \epsilon}{\log 2}$$

Bisection Method for root-finding

```
# Bisection method for root-finding

# f - user defined function
# a - start of an interval
# b - end of an interval
# nmax - maximal number of steps in the bisection method (divisions of the interval [a,b])
# eps - required precision for the root

BisectionMethod <- function(f, a, b, eps, nmax) {
    fa <- f(a)                                  # check if a or b are the root of f(x)
    if (fa == 0.0) {
        return(a)
    }
    fb <- f(b)
    if (fb == 0.0) {
        return(b)
    }

    k=1                                         # iteration nr. counter
    while ((abs(a-b)>eps)&&(k<nmax))
    {
    x0 <- a+(b-a)/2                             # finding midpoint of the interval
    if ((f(a) * f(x0)) < 0)
        b<-x0
    else
        a<-x0
    k<-k+1
    }

    if (k<nmax) {
        print('The found root on the interval [a,b] is:')
        return(x0)
    }
    else
        print('Maximal number of iterations reached and solution not yet found.')
}
```

Bisection Method for root-finding

```
f<-function(x)
{
    x**3-2
}

a<-1
b<-2
eps<-1e-10
n<-1000

source('BisectionMethod.R')
NewtonMethod(f,a,b,eps,n)
```

Bisection Method

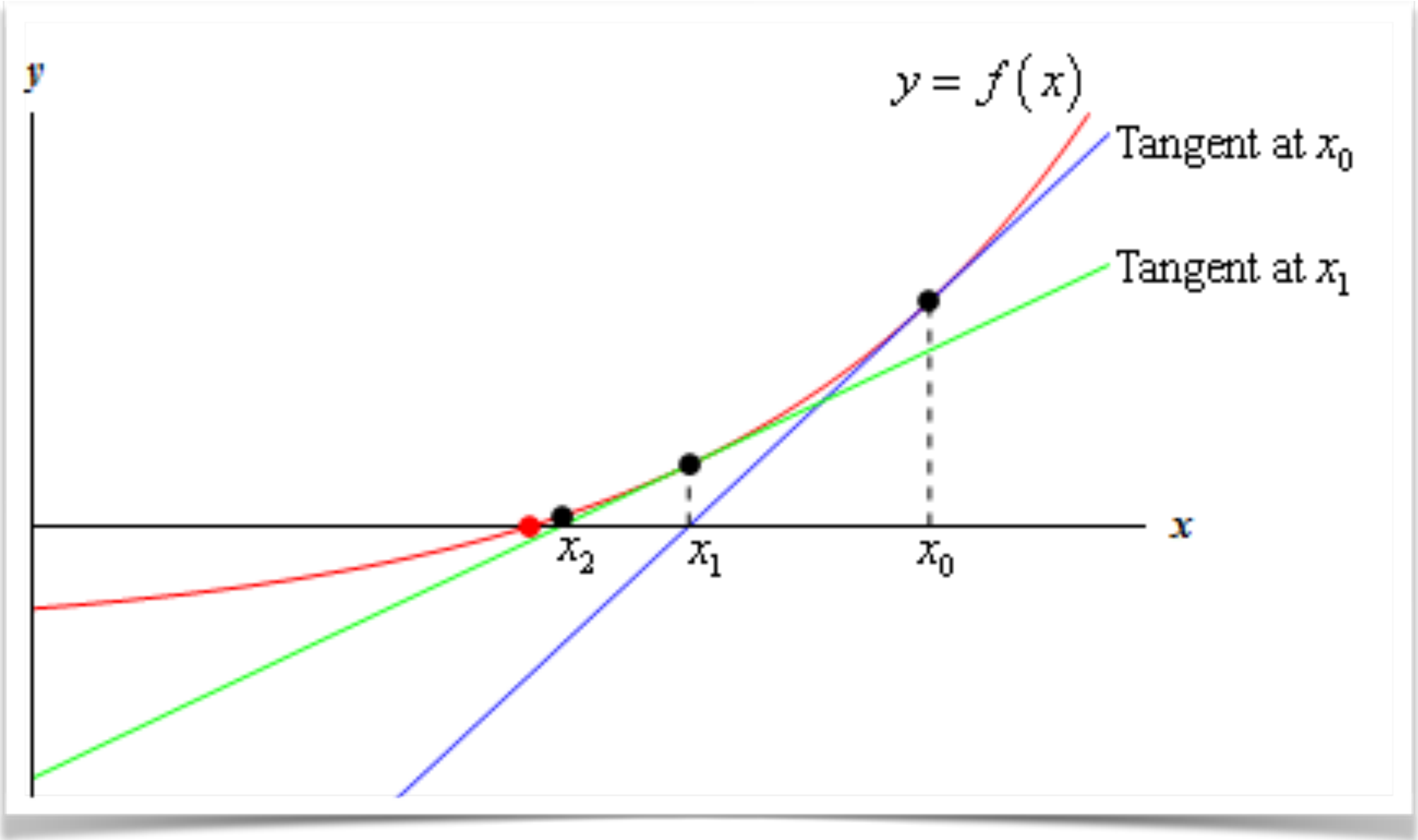
- **Globally convergent method:** it always converges no matter how far from the actual root we start, assuming that the root is "bracketed" (**f(a)f(b)<0**)

- Disadvantages:
  - cannot be used when the function is tangent to the axis and does not pass through the axis (e.g. $f(x)=x^2$)
  - converges slowly compared to other methods

- **How many iterations is needed in bisection method in order to decrease the initial error by a factor of ~1000?**

Newton's Method

- Historically - first used by Newton in 1669
- Babylonians also had a method for approximating sqrt(x)
- Assume we want to find a root of y=f(x) given an initial guess x_0
- **Newton's method uses tangent line approximation to f at (x_0,f(x_0))**



Newton's Method

➡ Tangent line approximation to f at $(x_0, f(x_0))$

$$\frac{y - y_0}{x - x0} = f'(x_0)$$

➡ Finding where this tangent line crosses the x-axis (y=0)

$$x = x_0 - \frac{f(x_0)}{f'(x_0)} \equiv x_1$$

➡ Continue the process with another tangent line through $f(x_1)$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

...

➡ And so on - until the new tangent line through $f(x_n)$

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

Newton's Method

➡ **Example:** $f(x)=2-e^x$, choosing $x_0=0$

$$x_1 = x_0 - \frac{2-e^{x_0}}{-e^{x_0}} = -\frac{2-1}{-1} = 1$$

$$x_2 = x_1 - \frac{2-e^{x_1}}{-e^{x_1}} = 1 - \frac{2-e}{-e} = 0.7357588823$$

$$x_3 = x_2 - \frac{2-e^{x_2}}{-e^{x_2}} = 0.6940422999$$

➡ **The convergence is much more rapid then the for the bisection**

n	x_n	$\alpha - x_n$	$\log_{10}(\alpha - x_n)$
0	0.0000000	0.6931472	-0.1591
1	1.000000	0.3068528	-0.5131
2	0.7357589	0.0426117	-1.3705
3	0.6940423	0.0008951	-3.0481
4	0.6931476	0.0000004	-6.3974
5	0.6931472	0.0000000	-13.0553

Convergence of Newton's Method

- **Study other examples:**

- $f(x)=4/3e^{2-x/2}(1+x^{-1}\log(x))$, Application of Newton's method will be problematic unless the initial guess x_0 is chosen very carefully. What happens in case $x_0 \in [0.8,1.2]$?

- How about $f(x)=\arctan(x)$ - try to apply Newton's method here, with initial guess $x_0=1.39174520027$

> If f, f' anf f" are continuous near the root, and if f' does not equal 0 at the root, then Newton's method will converge whenever the initial guess is sufficiently close to the root.

- This convergence will be very rapid (see example on previous page) –> number of correct digits doubling in every iteration

Newton's Method

- **Locally convergent method:** we have to start the iteration with a "good enough" approximation to the root, otherwise the method will not converge
- Stopping criterium for the Newton's method:

$$5\ |x_{n+1} - x_n| \leq \epsilon$$

- Warning! If $f'(x_n)$ is very large compared to $f(x_n)$, it is possible to have $|x_{n+1} - x_n|$ small and yet not have x_{n+1} very close to α
  - common to add a term to the error check:

$$|f(x_n)| + |x_n - x_{n-1}| \leq \epsilon/5$$

- Disadvantages:
  - requires a formula for the derivative of f(x)!

Newton’s Method for root-finding

```
# Note: other implementation of num. derivative is possible (e.g. backward, symmetric or some predefined function
in R could be used)

# f - user defined function
# a - start of an interval
# b - end of an interval
# h - step used in the numerical integration
# eps - required precision for the root
# n - maximal number of iterations

NumDerivative <- function (f, x0,dx)           # computing numerical derivative of a function f
{
    (f(x0+dx)-f(x0))/dx
}
NewtonMethod <- function(f, a, b, eps, n) {
  x0 <- a                                      # setting start value to the interval lower bound
  fa <- f(a)                                   # check if a or b are the root of f(x)
  if (fa == 0.0) {
    return(a)
  }
  fb <- f(b)
  if (fb == 0.0) {
    return(b)
  }
  for (k in 1:n) {
    fprime= NumDerivative(f, x0, dx)           # f'(x0)
    x1 = x0 - (f(x0) / fprime)                 # calculate next value x1
    if (abs(x1 - x0) < eps) {                  # check if required precision reached
      print('The found root on the interval [a,b] is:')
      return(x1)
    }
    x0 = x1                                    #continue Newton's method until convergence or max #iter. reached
  }
  print('Maximal number of iterations reached and solution not yet found.')
}
```

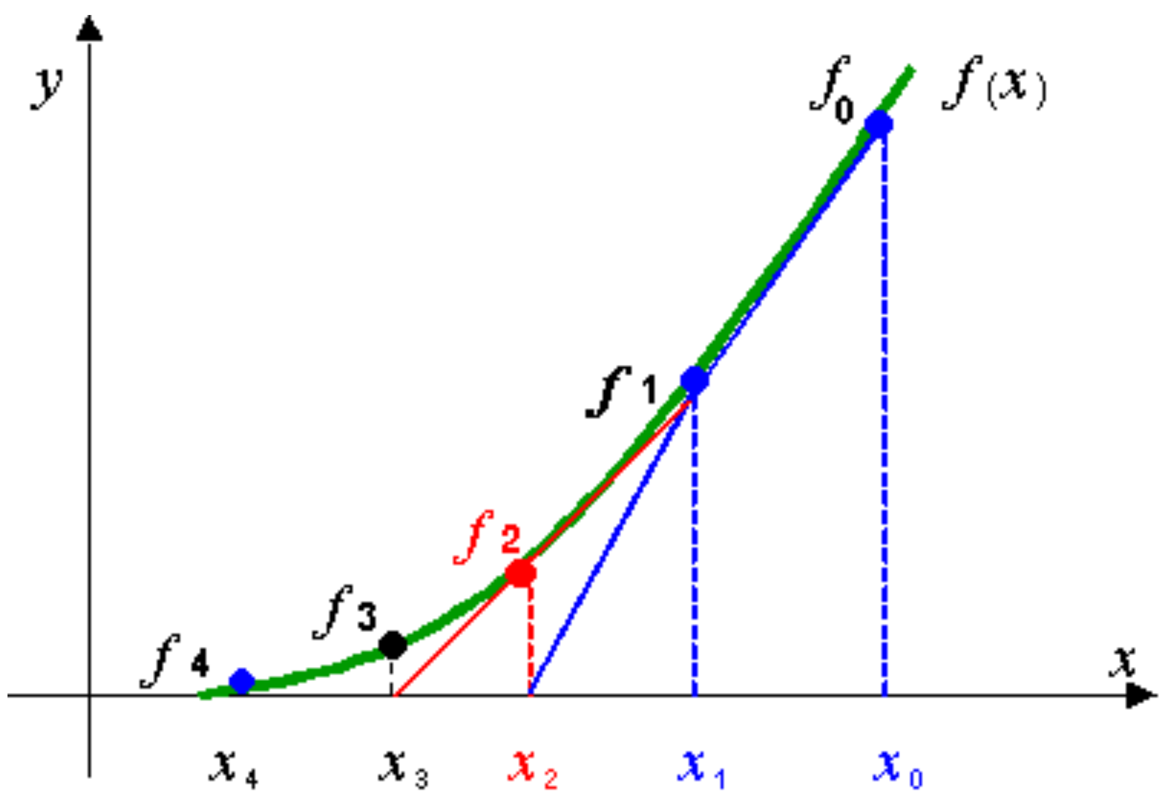
Newton's Method for root-finding

```
f<-function(x)
{
    x**3-2
}

a<-1
b<-2
dx<-1e-7
eps<-1e-10
n<-1000

source('NewtonMethod.R')
NewtonMethod(f,a,b,eps,n)
```

Secant Method

- Disadvantage of the Newton's Method:
  - requires a formula for the derivative of f(x)!

- Approximate the derivative with "secant line"
- two points in the curve needed (instead of just one for the tangent)
- Assume two initial guesses are given: x_0, x_1; $f(x_0)=f_0$ $f(x_1)=f_1$



Secant Method

- Assume two initial guesses are given: x_0, x_1
- The secant line

$$\frac{y - f(x_1)}{x - x_1} = \frac{f(x_1) - f(x_0)}{x_1 - x_0}$$

- Setting y=0 and solving for $x=x_2$:

$$x_2 = x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}$$

- Or, generally:

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

- Consistent with Newton's method, with approximation:

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

Secant Method

1. Given two initial guesses x_0, x_1, and $f(x_0)=f_0$ $f(x_1)=f_1$ set **k=1**
2. Compute $X_{k+1}=x_1-f_1*(x_1-x_0)/(f_1-f_0)$
3. Compute $f(X_{k+1})$; Assign: $x_0=x_1$, $x_1=X_{k+1}$; $f_0=f_1=f(x_0)$, $f_1=f(X_{k+1})$
4. If $|x_1-x_0|<tol$ then set **root**$=x_1$, exit the loop.

- Less costly than the Newton's method (one function call per iteration)
- Similar convergence propreties as Newton's method
- Error formula for secant method:

$$\alpha - x_{n+1} = \frac{1}{2}(\alpha - x_n)(\alpha - x_{n-1})\frac{f''(\zeta_n)}{f''(\eta_n)} \qquad \min\{\alpha, x_n, x_{n-1}\} \leq \zeta_n, \eta_n \leq \max\{\alpha, x_n, x_{n-1}\}$$

- Therefore:

$$|\alpha - x_{n+1}| = C|\alpha - x_n||\alpha - x_{n-1}|, \qquad \text{for} \quad x_n \approx \alpha, x_{n+1} \approx \alpha, x_{n-1} \approx \alpha$$

- Programming hint: avoid unnecessary function calls in your code!