Lookup tables and the Newton-Raphson method

Nicola Seriani

The Abdus Salam International Centre for Theoretical Physics,
Strada Costiera 11, 34151 Trieste, Italy

Introduction

- Up to now we have seen: spline interpolation, Taylor polynomials, Pade` approximants
- The strategy was to look for simpler ways to compute the function of interest
- Now we consider a different strategy: to store the data and minimize (or get rid of) computations

Look-up tables

- Now we consider a different strategy: to store the data and minimize (or get rid of) computations
- Historically, people would use tables with values of functions
- Oldest examples: chord tables by Hipparchus, 190-120 BC; sine tables by Aryabhata, from the Kerala school, year 499 CE

Look-up tables

- Now we consider a different strategy: to store the data and minimize (or get rid of) computations
- We consider a set of points x_i (i=1, N), and we store the values of the function f in an array $f_i = f(x_i)$
- Retrieving a value for memory can be faster than computing it

Look-up tables

- Retrieving a value for memory can be faster than computing it
- This is particularly true for complex functions, like sin(x)
- Values of the function are computed on an array of evenly distributed points $\{x_i\}$, and store them
- When the value of the function at a certain X is needed, the table is used
- Different strategies can be employed: if the $\{x_i\}$ array is fine and the function is very smooth, one can simply take the value of the x_i nearest to X
- This will not work at the desired accuracy in many cases

Look-up tables: calculating sin(x)

- Example: computing sin(x)

- One can create an array to store sin(x) evaluated on 2000 points:

```
real array sine_table[-1000..1000]
for x from -1000 to 1000
    sine_table[x] := sine(π * x / 1000)
```

and then

```
function lookup_sine(x)
    return sine_table[round(1000 * x / π)]
```

Look-up tables: calculating sin(x)

- *real array* sine_table[-1000..1000]

This takes up some memory

To reduce memory requirements:

1) Reduce number of points, and use interpolation

OR

2) Use symmetry properties of sine and cosine to reduce interval of definition: then, sine can be defined in [0; π/2]

Look-up tables: calculating sin(x)

2) Use symmetry properties of sine and cosine to reduce interval of definition: then, sine can be defined in [0; π/2]

This needs some additional operation to reduce everything to sin(x'), with x' in [0; π/2]

$\sin(x) = \sin(\pi - x)$

$\cos(x) = -\cos(\pi - x)$

...

Look-up tables: calculating sin(x)

1) Reduce number of points, and use interpolation

   a) Linear interpolation

   b) Newton-Raphson method

Non-uniform sampling

Look-up tables

- Different strategies can be employed: if the $\{x_i\}$ array is fine and the function is very smooth, one can simply take the value of the x_i nearest to X
- This will not work at the desired accuracy in many cases
- Then one has to use **interpolation**
- For example, linear interpolation (for simple cases) or the Newton-Raphson method
- It works fast e.g. for approximation of the square root.

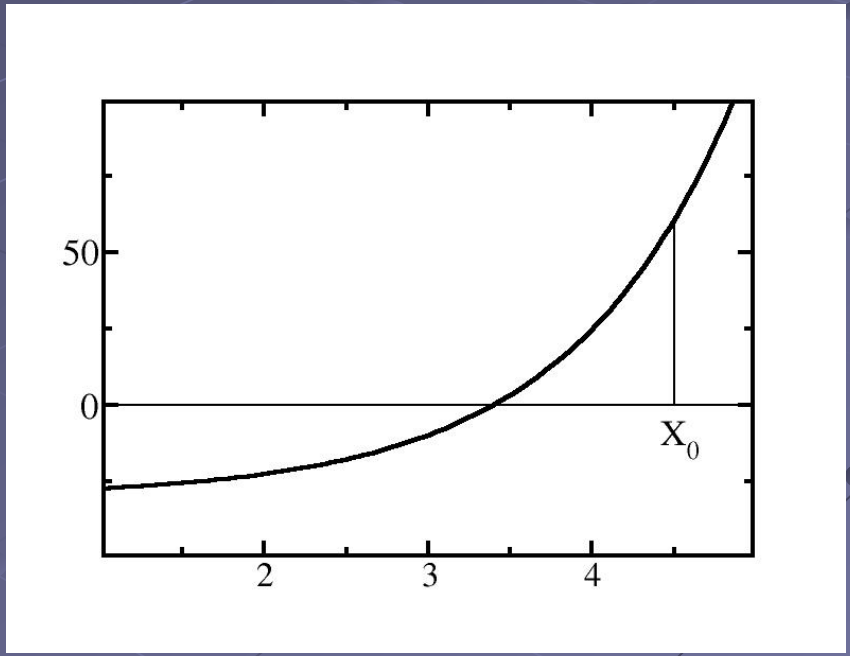
Small detour: Newton-Raphson method

- Newton-Raphson method is a method to find zero's of a differentiable function, i.e. x's such that f(x) = 0
- Useful to solve trascendental equations, like $x = e^x$
  In this case $f(x) = x - e^x = 0$
- It is an iterative method, using the derivative of the function to deliver a sequence of approximate solutions

Newton-Raphson method

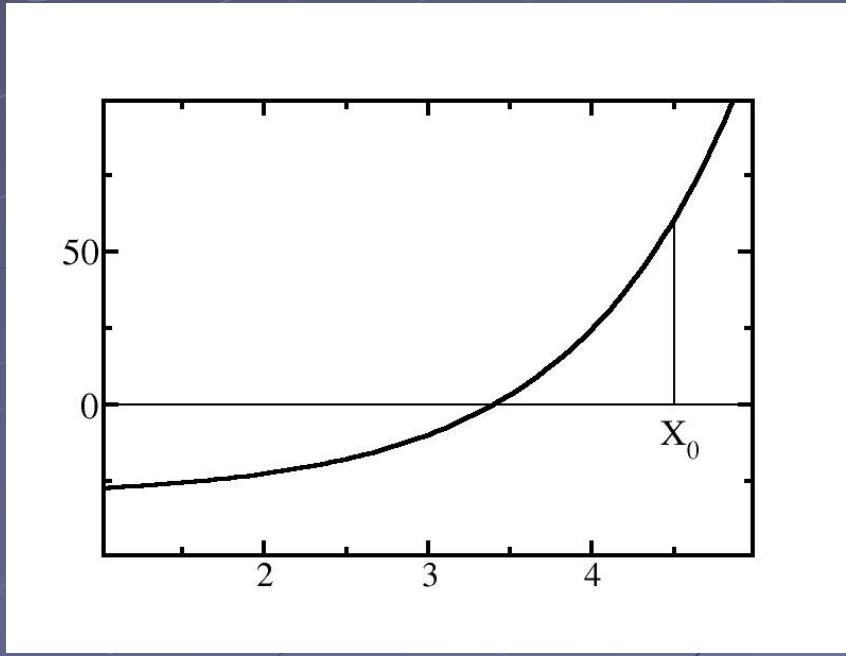
- It is an iterative method, using the derivative of the function to deliver a sequence of approximate solutions

- We need to know that there is only one zero in an interval [a, b], and we need an initial guess of our zero, x_0, in that interval



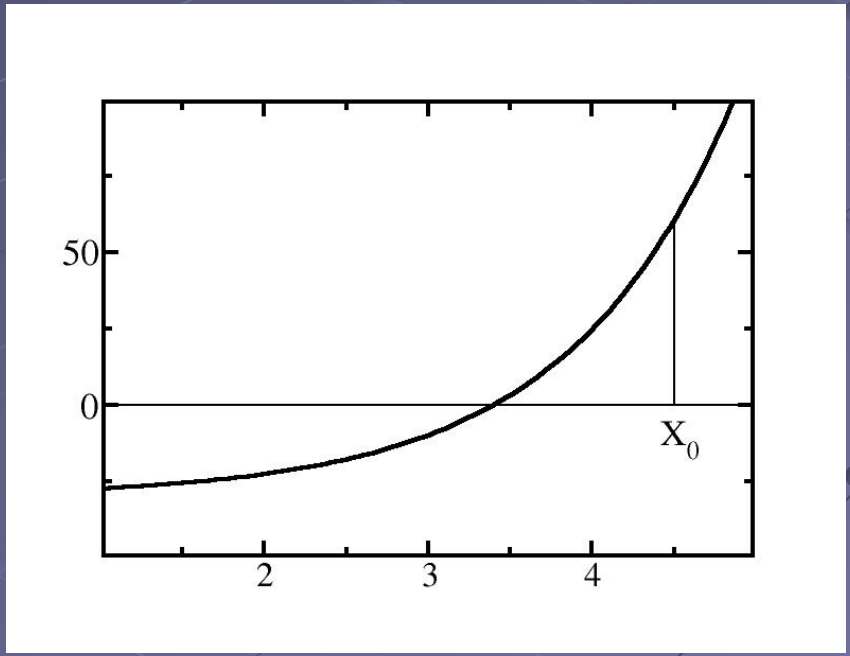
Newton-Raphson method

- We need to know that there is only one zero in an interval [a, b], and we need an initial guess of our zero, x_0, in that interval
- Given the estimate x_0, a new estimate is produced linearizing the function: $y = f(x_0)+f'(x_0)\,(x-x_0)$



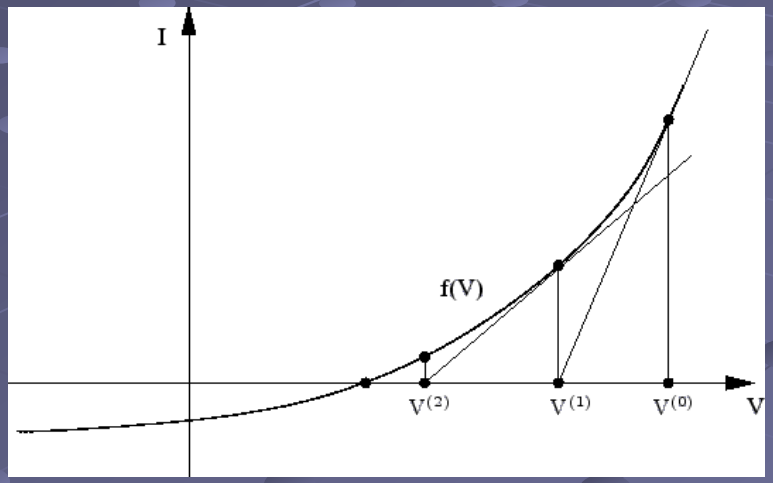
Newton-Raphson method

- $y = f(x_0)+f'(x_0)\,(x-x_0)$
- The new estimate for the zero is the x_1 such that $y = 0$
- $0 = f(x_0)+f'(x_0)\,(x_1-x_0)$
- $x_1 = x_0 - f(x_0)/f'(x_0)$



Newton-Raphson method

- It is an iterative method, using the derivative of the function



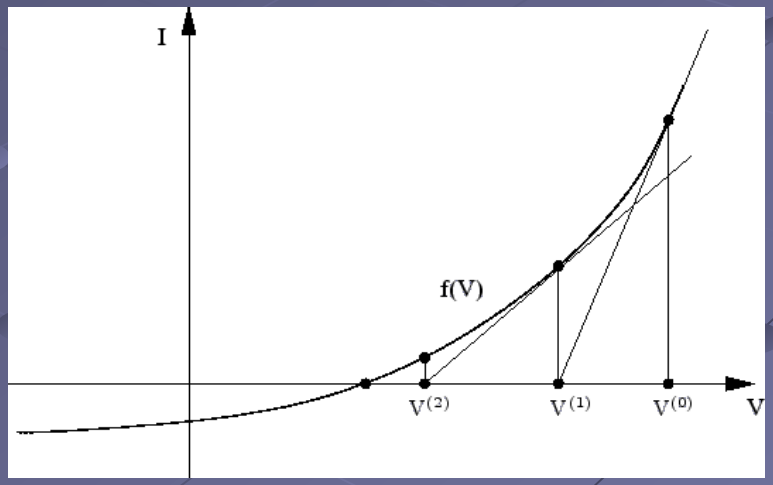
http://qucs.sourceforge.net/tech/node16.html

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Newton-Raphson method

- Newton-Raphson method often converges fast, and can be used to approximate functions

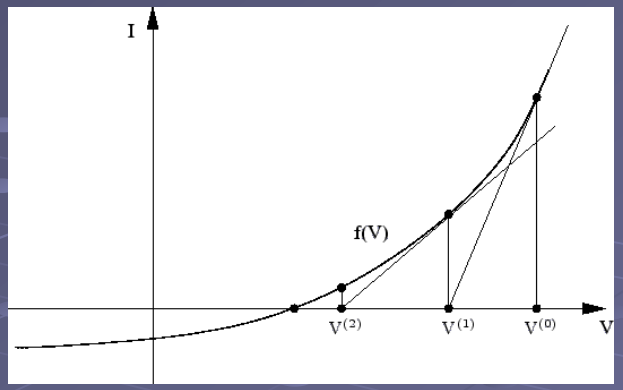
- In the example x_{i+1} is a better approximation than x_i for every i; this is because f' is never zero, and the function is convex



http://qucs.sourceforge.net/tech/node16.html

Newton-Raphson method: a convergence theorem

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$



- Theorem: differentiable f in [a, b], such that:

  a) There is one and only one r in [a, b] for which f(r) = 0

  b) For every x, f'(x)≠0 and f''(x) > 0 or f''(x)<0

  Then the sequence

  i) x_0 = a if f(a)f''(a) > 0

  ii) x_0 = b if f(b)f''(b) > 0

  and $x_n = x_{n-1} - f(x_{n-1})/f'(x_{n-1})$

  converges to r for n –> ∞

Newton-Raphson method: a convergence theorem

- Theorem: differentiable f in [a, b], such that:

  a) There is one and only one r in [a, b] for which f(r) = 0

  b) For every x, $f'(x) \neq 0$ and $f''(x) > 0$ or $f''(x) < 0$

  Then the sequence

  i) $x_0 = a$ if $f(a)f''(a) > 0$

  ii) $x_0 = b$ if $f(b)f''(b) > 0$

  and $x_n = x_{n-1} - f(x_{n-1})/f'(x_{n-1})$

  converges to r for $n \rightarrow \infty$

Proof (case $f(a) > 0$, $f''(a) > 0$): therefore $f''(x) > 0$ and $f'(x) < 0$ for every x. Therefore $x_{n-1} < x_n < r$. The sequence $\{x_n\}$ is monotonous and limited, it has therefore a limit $L \leq r$. At the limit, $L = L - f(L)/f'(L)$, i.e. $f(L) = 0$. QED

Newton-Raphson method: when is it convenient?

- $f(x) = 0 \Rightarrow x = f^{-1}(0)$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

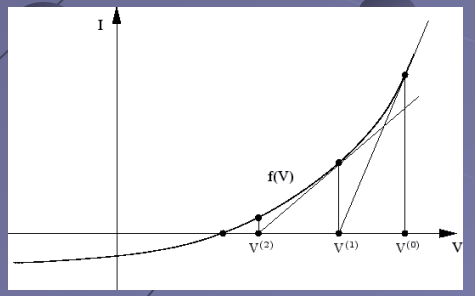
- Requirement: f easier to compute than $f^{-1}(x)$
- Requirement: f' easy to compute
- Requirement: initial guess near to the solution
- Typical example: calculate $y = \sqrt{A}$; it is equivalent to find the zero of $f(x) = x^2 - A$; much easier to calculate x^2 than $\sqrt{}$

Newton-Raphson method

- Newton-Raphson method converges fast, and can be used to approximate functions

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

- It is typically used to calculate the square root of numbers; imagine we want to find the square root of a number, e.g. 389. Then we want to solve $x^2 = 389$
- This corresponds to finding the zero of the function $f(x) = x^2 - 389$



Newton-Raphson method

- $f(x) = x^2 - 389$
- $f'(x) = 2x$

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

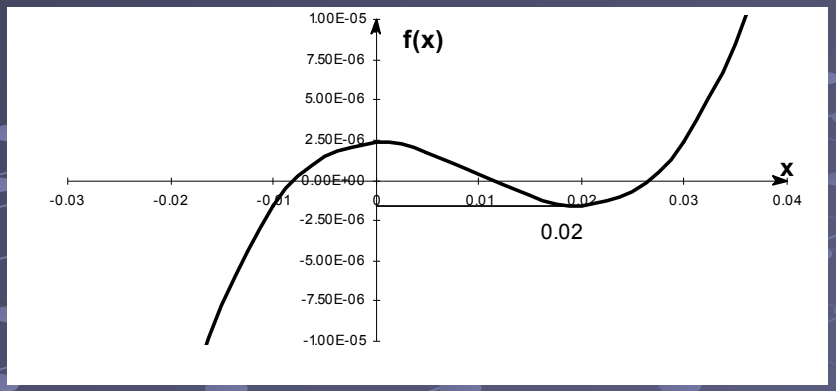
- Let's start from an initial guess: $x_0 = 20$
- $x_1 = 20 - (400-389)/(2*20) = 19.725000000$
- $x_2 = 19.725000000 - (389.075625000-389)/(2*19.725000000) = 19.723083016476552598 23$
- $x_3 = \ldots = 19.723082923316020188 09$
- $x_4 = \ldots = 19.723082923316019968 07$
- Near a root, Newton-Raphson converges fast

Disadvantages of the Newton-Raphson method

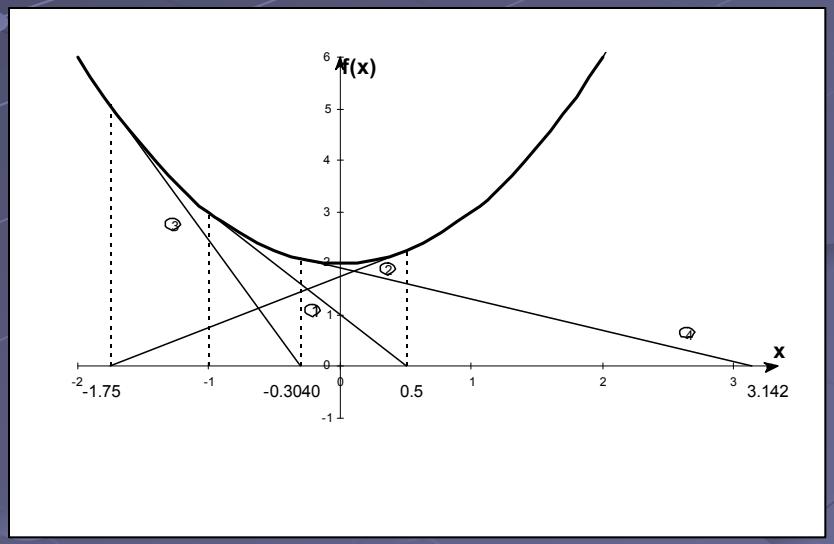
- Must find derivative
- Poor global convergence properties
- Dependence on initial guess: too far from root, zero derivative, may loop indefinitely

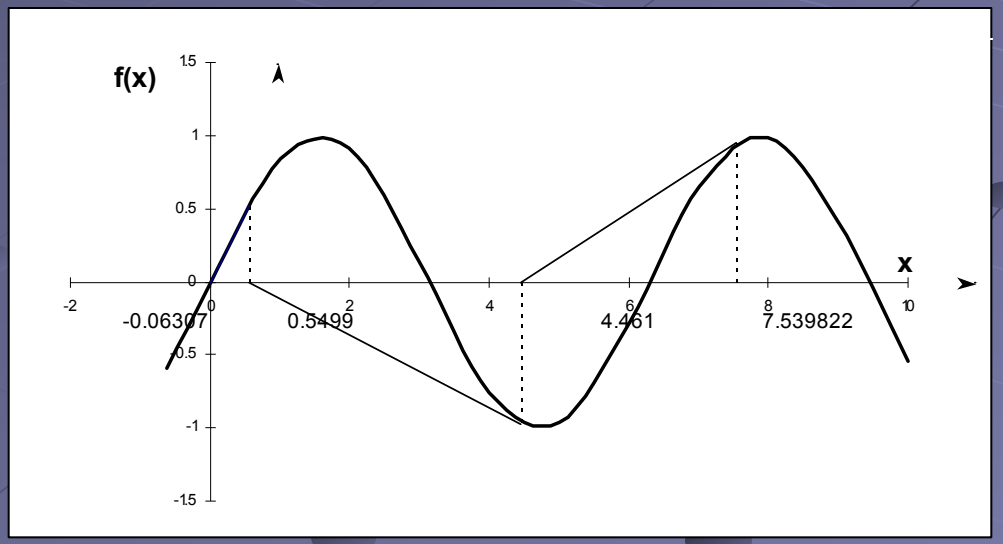
$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Disadvantages of the Newton-Raphson method



$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$





http://nm.mathforcollege.com/mws/gen/03nle/mws_gen_nle_txt_newton.pdf

Combining look-up tables and the Newton-Raphson method

- One can employ the Newton-Raphson method using a look-up table to provide the initial guess.

- It combines the strengths of both methods: a reasonable guess from the lookup table, a quick and accurate refining from Newton-Raphson