Roots

Introduction to Numerical Analysis

Riccardo Mannella

February 16, 2008

Rootfinding: art

Root finding is a very difficult chapter of numerical analysis.

Rootfinding: art

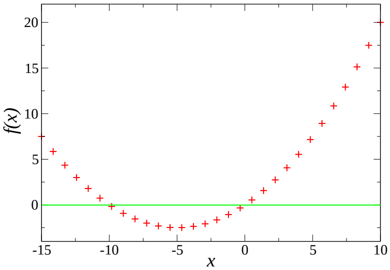
Root finding is a very difficult chapter of numerical analysis. The idea is to have a rough idea where the root is, and then compute it exactly (refine the root). The first step might be difficult, and it must be done sampling the appropriate range, if no analytical hint is available.

Rootfinding: art

Root finding is a very difficult chapter of numerical analysis. The idea is to have a rough idea where the root is, and then compute it exactly (refine the root). The first step might be difficult, and it must be done sampling the appropriate range, if no analytical hint is available.

$$f(x) = x + x^2/10$$

in the range $[-15, 10]$, with 30 points



Rootfinding: art/2

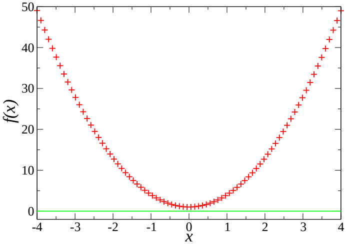
There are however some cases which may be hard to deal with numerically, almost pathologic ones (NumRec):

$$f(x) = 3x^2 + 1 + \frac{\ln\left[(\pi - x)^2\right]}{\pi^4}$$

Rootfinding: art/2

There are however some cases which may be hard to deal with numerically, almost pathologic ones (NumRec):

$$f(x) = 3x^2 + 1 + \frac{\ln\left[(\pi - x)^2\right]}{\pi^4}$$



80 points in the range $[-4, 4]$

Rootfinding: art/2

There are however some cases which may be hard to deal with numerically, almost pathologic ones (NumRec):

$$f(x) = 3x^2 + 1 + \frac{\ln\left[(\pi - x)^2\right]}{\pi^4}$$

There are two roots in $x \approx \pi \pm 10^{-667}$, although their rough location cannot be easily found numerically: but we know that the logarithm must go to some large negative value, hence the root is near π. In any case, we assume that we found some neighborhood of the root, and see now how to refine it.

Refining: bisection

We now have a region where there is a root. The simplest approach is to use the bisection method, based on the function changing sign at the root.

Refining: bisection

We now have a region where there is a root. The simplest approach is to use the bisection method, based on the function changing sign at the root.

Suppose we know that the function is positive in x_1 and negative in x_2, we take $x_m = (x_1 + x_2)/2$ (we bisect the initial range) and check the sign in x_m: if positive (negative) we replace $x_1(x_2)$ with x_m, and repeat until we reach the desired accuracy. If initially $\epsilon = |x_1 - x_2|$, after n steps the root has an accuracy which is order of

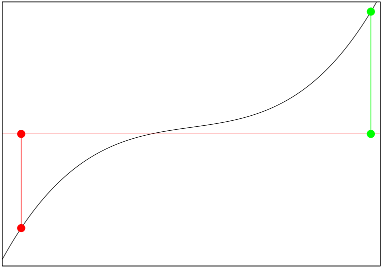
$$\delta \approx \frac{\epsilon}{2^n}$$

Let us see the method with some cartoons:

Refining: bisection

We now have a region where there is a root. The simplest approach is to use the bisection method, based on the function changing sign at the root.

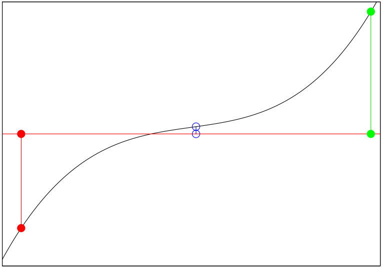
$f(x) = x^3/5 + x/5 + 0.1$



Refining: bisection

We now have a region where there is a root. The simplest approach is to use the bisection method, based on the function changing sign at the root.

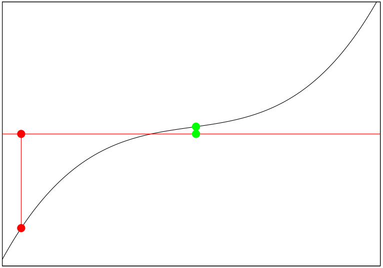
$f(x) = x^3/5 + x/5 + 0.1$



Refining: bisection

We now have a region where there is a root. The simplest approach is to use the bisection method, based on the function changing sign at the root.

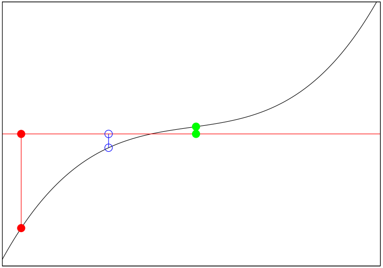
$f(x) = x^3/5 + x/5 + 0.1$



Refining: bisection

We now have a region where there is a root. The simplest approach is to use the bisection method, based on the function changing sign at the root.

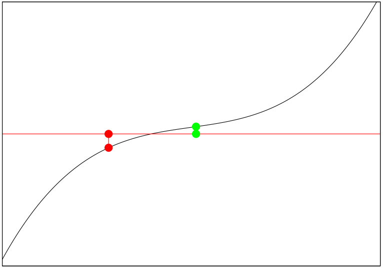
$f(x) = x^3/5 + x/5 + 0.1$



Refining: bisection

We now have a region where there is a root. The simplest approach is to use the bisection method, based on the function changing sign at the root.

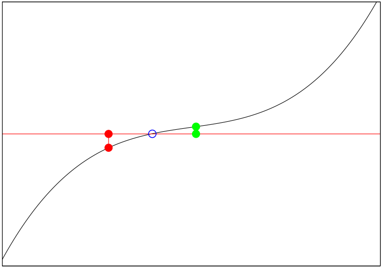
$f(x) = x^3/5 + x/5 + 0.1$



Refining: bisection

We now have a region where there is a root. The simplest approach is to use the bisection method, based on the function changing sign at the root.

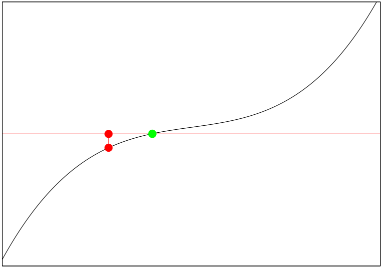
$f(x) = x^3/5 + x/5 + 0.1$



Refining: bisection

We now have a region where there is a root. The simplest approach is to use the bisection method, based on the function changing sign at the root.

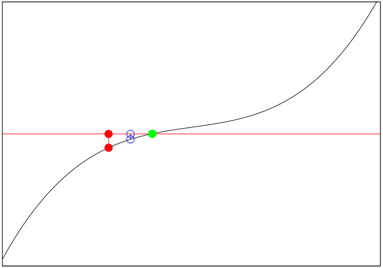
$f(x) = x^3/5 + x/5 + 0.1$



Refining: bisection

We now have a region where there is a root. The simplest approach is to use the bisection method, based on the function changing sign at the root.

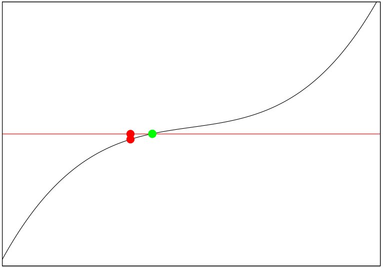
$f(x) = x^3/5 + x/5 + 0.1$



Refining: bisection

We now have a region where there is a root. The simplest approach is to use the bisection method, based on the function changing sign at the root.

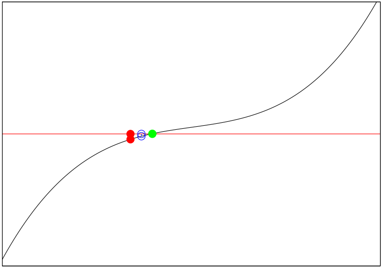
$f(x) = x^3/5 + x/5 + 0.1$



Refining: bisection

We now have a region where there is a root. The simplest approach is to use the bisection method, based on the function changing sign at the root.

$f(x) = x^3/5 + x/5 + 0.1$

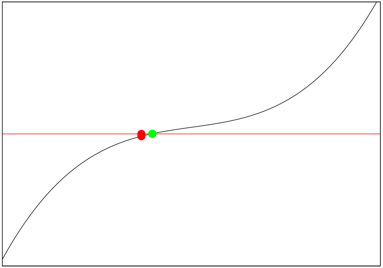


Refining: bisection

We now have a region where there is a root. The simplest approach is to use the bisection method, based on the function changing sign at the root.

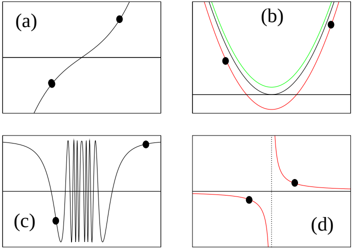
$f(x) = x^3/5 + x/5 + 0.1$

i	x_r	δ
1	-0.9	0.9
2	-0.45	0.45
3	-0.23	0.23
4	-0.338	0.11
5	-0.394	0.06
6	-0.422	0.03



Bisection: comments

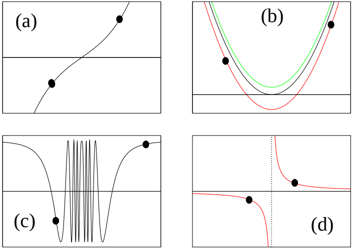
Cases which can happen:



(a) Normal case

Bisection: comments

Cases which can happen:

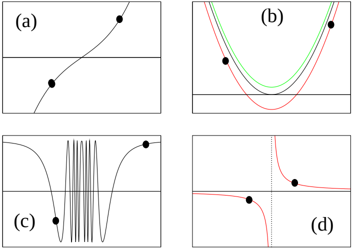


(a) Normal case

(b) Double roots

Bisection: comments

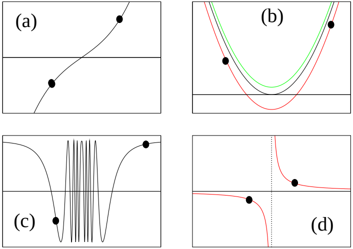
Cases which can happen:



(a) Normal case
(b) Double roots
(c) Very close, even infinite roots

Bisection: comments

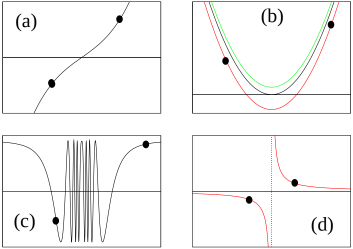
Cases which can happen:



(a) Normal case
(b) Double roots
(c) Very close, even infinite roots
(d) Discontinuities

Bisection: comments

Cases which can happen:



(a) Normal case
(b) Double roots
(c) Very close, even infinite roots
(d) Discontinuities

Bisection will always converge to something: in (d), one gets the discontinuity point. It requires bracketing of the root, the error goes like $\delta = \epsilon/2^n$, works in 1D.

Refining: Newton

We again have a region where there is a root.

Refining: Newton

We again have a region where there is a root.
The Newton-Raphson method starts from an analytical approximation of the function close to the root (x_r):

$$0 = f(x_r) \approx f(x) + (x_r - x)\frac{df(x)}{dx}$$

Solving for x_r, we have the relation

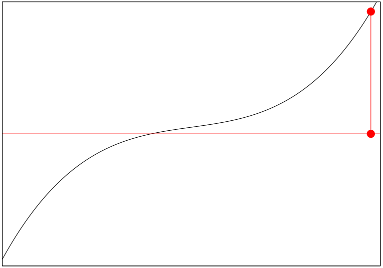
$$x_r = x - f(x)/f'(x)$$

which is iterated to find x_r from an initial guess x. Let us see the methods in cartoons:

Refining: Newton

We again have a region where there is a root. $x_r = x - f(x)/f'(x)$

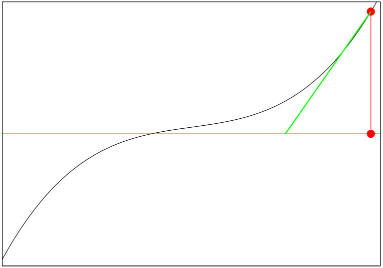
$f(x) = x^3/5 + x/5 + 0.1$



Refining: Newton

We again have a region where there is a root. $x_r = x - f(x)/f'(x)$

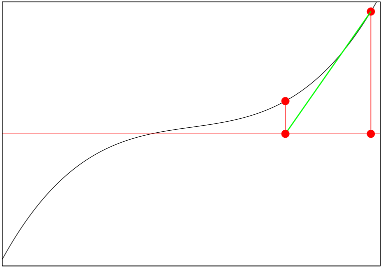
$f(x) = x^3/5 + x/5 + 0.1$



Refining: Newton

We again have a region where there is a root. $x_r = x - f(x)/f'(x)$

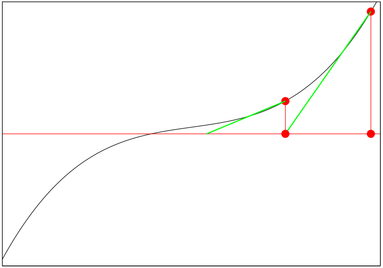
$f(x) = x^3/5 + x/5 + 0.1$



Refining: Newton

We again have a region where there is a root. $x_r = x - f(x)/f'(x)$

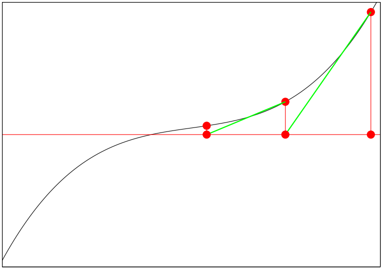
$f(x) = x^3/5 + x/5 + 0.1$



Refining: Newton

We again have a region where there is a root. $x_r = x - f(x)/f'(x)$

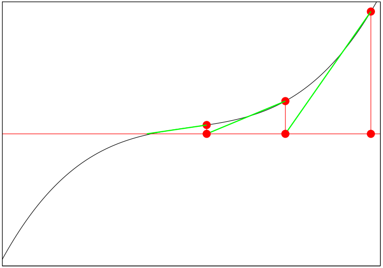
$f(x) = x^3/5 + x/5 + 0.1$



Refining: Newton

We again have a region where there is a root. $x_r = x - f(x)/f'(x)$

$f(x) = x^3/5 + x/5 + 0.1$

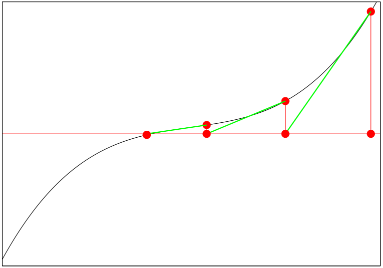


Refining: Newton

We again have a region where there is a root. $x_r = x - f(x)/f'(x)$

$f(x) = x^3/5 + x/5 + 0.1$

i	x_r	$f(x_r)$
0	1.9	
1	1.1173	0.60
2	0.4825	0.22
4	-0.4714	0.02
5	-0.4257	0.0006
7	-0.4239	$< 10^{-8}$



Newton: comments

The method is fast and accurate, exact for linear functions, works in many dimensions, does not require bracketing. The error on the root is order of the distance between the different approximations.

Newton: comments

The method is fast and accurate, exact for linear functions, works in many dimensions, does not require bracketing. The error on the root is order of the distance between the different approximations. Take $x = x_r + \epsilon$,

$$f(x+\epsilon) = f(x) + \epsilon f'(x) + \epsilon^2 f''(x)/2$$

$$x_{i+i} = x_i - f(x_i)/f'(x_i) \quad \epsilon_{i+i} = \epsilon_i - f(x_i)/f'(x_i)$$

use $f(x)$ around the root x_r

$$\epsilon_{i+1} = -\epsilon_i^2 \frac{f''(x_i)}{2f'(x_i)}$$

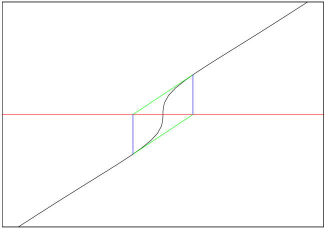
Newton: comments

The method is fast and accurate, exact for linear functions, works in many dimensions, does not require bracketing. The error on the root is order of the distance between the different approximations.

However, it breaks near a discontinuity and can get "stuck".

Newton: comments

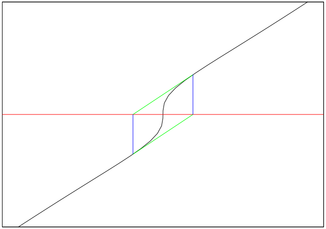
The method is fast and accurate, exact for linear functions, works in many dimensions, does not require bracketing. The error on the root is order of the distance between the different approximations.



However, it breaks near a discontinuity and can get "stuck".

Newton: comments

The method is fast and accurate, exact for linear functions, works in many dimensions, does not require bracketing. The error on the root is order of the distance between the different approximations.



However, it breaks near a discontinuity and can get "stuck".
It requires $f'(x)$ to be different from zero, but we can take higher order terms in the Taylor expansion.

Polynomials: sure fire techniques

Laguerre's method to find the root of a polynomials.

$$P_n(x) = (x - x_1)(x - x_2)...(x - x_n)$$

$$\ln|P_n(x)| = \sum_i^n \ln|x - x_i|$$

$$\frac{d\ln|P_n(x)|}{dx} = \sum_i \frac{1}{x - x_i} = \frac{P_n'}{P_n} = G$$

$$-\frac{d^2\ln|P_n(x)|}{dx^2} = \sum_i \frac{1}{(x - x_i)^2} = \left[\frac{P_n'}{P_n}\right]^2 - \frac{P_n''}{P_n} = H$$

Polynomials: sure fire techniques/2

We are seeking x_1, which is a away from x, current guess. Take all other roots to be b away. We can write

$$G = \frac{1}{a} + \frac{n-1}{b} \qquad H = \frac{1}{a^2} + \frac{n-1}{b^2}$$

and we get

$$a = \frac{n}{G \pm \sqrt{(n-1)(nH - G^2)}}$$

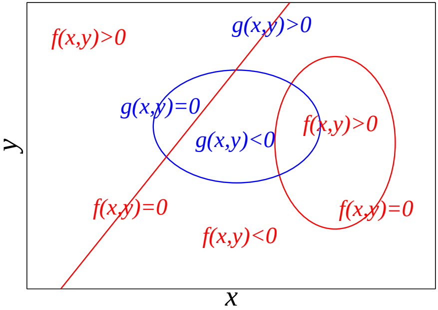
the sign is taken to give the largest magnitude in the denominator. Start with a trial x, compute a, then the new $x - a$ is the new trial value etc

Multidimensions

We face the same challenge of $1D$, but in many dimensions.

Multidimensions

We face the same challenge of $1D$, but in many dimensions.
Typical case:



Multidimensions

We face the same challenge of $1D$, but in many dimensions. Use Newton-Raphson: there are N relations $f_i(X) = 0$, where X is an N dimensional vector. Assume we are in X, and $X + \delta$ is the root:

$$f_i(X + \delta) = f_i(X) + \sum_j \frac{\partial f_i}{\partial x_j}\delta_j = f_i(X) + \sum_j a_{ij}\delta_j$$

$$\delta_j = -\sum_i (a_{ij})^{-1} f_i(X)$$

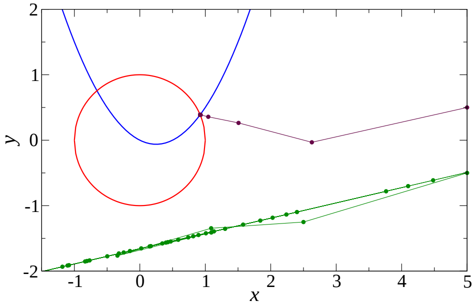
and iterate as

$$X_i(\text{new}) = X_i(\text{old}) + \delta_i$$

Multidimensions

We face the same challenge of $1D$, but in many dimensions.
Example:

$$f_1(x,y) = x^2 + y^2 - 1 \quad f_2(x,y) = y - x^2 + x/2$$



Roots: summary

Let us summarise:

- We need to have a rough idea of where the root is.

Roots: summary

Let us summarise:

- We need to have a rough idea of where the root is.
- Bisection: it normally works, but restricted to 1D, requires bracketing, it is slows.

Roots: summary

Let us summarise:

- We need to have a rough idea of where the root is.
- Bisection: it normally works, but restricted to 1D, requires bracketing, it is slows.
- Newton-Raphson: fast, does not require bracketing, but can get stuck, needs to start close to the root, fooled by extrema of the function.

Roots: summary

Let us summarise:

- We need to have a rough idea of where the root is.
- Bisection: it normally works, but restricted to 1D, requires bracketing, it is slows.
- Newton-Raphson: fast, does not require bracketing, but can get stuck, needs to start close to the root, fooled by extrema of the function.
- For polynamials, there are sure-fire techniques.

Roots: summary

Let us summarise:

- We need to have a rough idea of where the root is.
- Bisection: it normally works, but restricted to 1D, requires bracketing, it is slows.
- Newton-Raphson: fast, does not require bracketing, but can get stuck, needs to start close to the root, fooled by extrema of the function.
- For polynamials, there are sure-fire techniques.
- In multidimensions, Newton-Raphson is basically the only thing, but it is sensitive to the initial conditions.