Nonlinear Equations and Continuous Optimization

Sanzheng Qiao

Department of Computing and Software
McMaster University

November, 2012

Outline

Outline

Problem setting

Find roots

$$f(x) = 0$$

Often, methods are iterative (roots cannot be found in finite number of steps).

Example

Compute square roots

$$x^2 - A = 0$$

Problem setting

Find roots

$$f(x) = 0$$

Often, methods are iterative (roots cannot be found in finite number of steps).

Example

Compute square roots

$$x^2 - A = 0$$

Find the side of the square whose area is A

Compute square roots

Start with a rectangle whose one side is x_c, then the other side is A/x_c so that its area is A.
Make the rectangle “more square” by setting the new side:

$$x_+ = \frac{1}{2}\left(x_c + \frac{A}{x_c}\right)$$

Then $x_c = x_+$ and iterate.

Compute square roots

Start with a rectangle whose one side is x_c, then the other side is A/x_c so that its area is A.
Make the rectangle "more square" by setting the new side:

$$x_+ = \frac{1}{2}\left(x_c + \frac{A}{x_c}\right)$$

Then $x_c = x_+$ and iterate.

A better form

$$x_+ = x_c - \frac{1}{2}\left(x_c - \frac{A}{x_c}\right)$$

Compute square roots

Three issues to be addressed

- Initialization (x_0)
- Convergence ($x_k \to x_*$?) and rate (how fast?)
- Termination

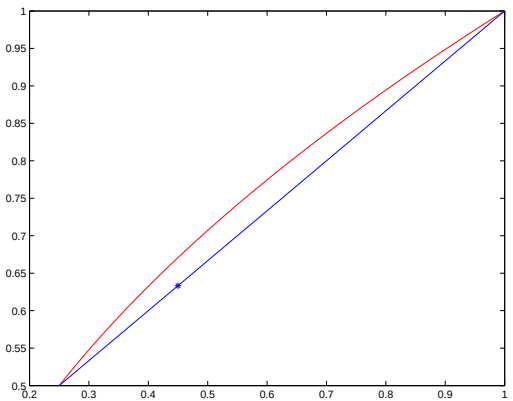
Initialization

Write A in base 4:

$$A = m \times 4^e, \qquad 0.25 \le m < 1$$

then $\sqrt{A} = \sqrt{m} \times 2^e$.
Now we can assume $4^{-1} \le A < 1$.
Linear interpolation of $f(A) = \sqrt{A}$ at $A = 0.25, 1.0$:

$$p(A) = (1 + 2A)/3.$$



Initialization (cont.)

Initial error bound:
Differentiating

$$\sqrt{A} - \frac{1 + 2A}{3}$$

with respect to A and then setting the derivative to zero to find the maximum, it can be shown that

$$\left| \sqrt{A} - (1 + 2A)/3 \right| \leq 0.05$$

Initialization (cont.)

Initial error bound:
Differentiating

$$\sqrt{A} - \frac{1 + 2A}{3}$$

with respect to A and then setting the derivative to zero to find the maximum, it can be shown that

$$\left| \sqrt{A} - (1 + 2A)/3 \right| \leq 0.05$$

Initial value: $x_0 = (1 + 2A)/3$
Initial error: $e_0 \leq 0.05$

Convergence

A relation between x_{k+1} and x_k:

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{A}{x_k}\right)$$

Denote the error $e_k = |x_k - \sqrt{A}|$, then the relation between e_{k+1} and e_k:

$$\begin{aligned} e_{k+1} &= |x_{k+1} - \sqrt{A}| = \frac{1}{2}\left(\frac{x_k - \sqrt{A}}{\sqrt{x_k}}\right)^2 \\ &= \frac{1}{2|x_k|}e_k^2 \end{aligned}$$

Convergence

A relation between x_{k+1} and x_k:

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{A}{x_k}\right)$$

Denote the error $e_k = |x_k - \sqrt{A}|$, then the relation between e_{k+1} and e_k:

$$\begin{aligned} e_{k+1} &= |x_{k+1} - \sqrt{A}| = \frac{1}{2}\left(\frac{x_k - \sqrt{A}}{\sqrt{x_k}}\right)^2 \\ &= \frac{1}{2|x_k|}e_k^2 \end{aligned}$$

It can be shown that $0.5 \leq x_k \leq 1.0$.

Convergence (cont.)

Since the initial error $e_0 \leq 0.05$,

$$e_k \leq e_{k-1}^2 \leq \cdots \leq e_0^{2^k} \leq (0.05)^{2^k}$$

We have shown the convergence ($e_k \to 0$ as $k \to \infty$).

Convergence (cont.)

Since the initial error $e_0 \leq 0.05$,

$$e_k \leq e_{k-1}^2 \leq \cdots \leq e_0^{2^k} \leq (0.05)^{2^k}$$

We have shown the convergence ($e_k \to 0$ as $k \to \infty$).

How fast?
Rate: quadratic $e_{k+1} \leq ce_k^2$, each iteration doubles the accuracy.

Termination

Recall: $e_k \leq (0.05)^{2^k} < 10^{-2^k}$.

Termination

Recall: $e_k \leq (0.05)^{2^k} < 10^{-2^k}$.

When $k = 3$, $e_k < 10^{-8}$.
When $k = 4$, $e_k < 10^{-16}$.

Termination

Recall: $e_k \leq (0.05)^{2^k} < 10^{-2^k}$.

When $k = 3$, $e_k < 10^{-8}$.
When $k = 4$, $e_k < 10^{-16}$.

Three iterations are enough for IEEE single precision (2^{-24}).

Four iterations are enough for IEEE double precision (2^{-53}).

Example

Compute $\sqrt{3}$

Example

Compute $\sqrt{3}$

Scale: $3 = 0.75 \times 4$

Example

Compute $\sqrt{3}$

Scale: $3 = 0.75 \times 4$

Initial: $x_0 = (1 + 2 \times 0.75)/3 = 2.5/3$

Example

Compute $\sqrt{3}$

Scale: $3 = 0.75 \times 4$

Initial: $x_0 = (1 + 2 \times 0.75)/3 = 2.5/3$

Iterate: $x_{n+1} = x_n - (x_n - 0.75/x_n)/2$

n	x_n	error
0	0.8333...	3.3×10^{-2}
1	0.8667...	6.4×10^{-4}
2	0.8660...	2.4×10^{-7}
3	0.8660...	3.2×10^{-14}
4	0.8660...	$< 10^{-16}$

$x_5 = x_4$.

Example

Compute $\sqrt{3}$

Scale: $3 = 0.75 \times 4$

Initial: $x_0 = (1 + 2 \times 0.75)/3 = 2.5/3$

Iterate: $x_{n+1} = x_n - (x_n - 0.75/x_n)/2$

n	x_n	error
0	0.8333...	3.3×10^{-2}
1	0.8667...	6.4×10^{-4}
2	0.8660...	2.4×10^{-7}
3	0.8660...	3.2×10^{-14}
4	0.8660...	$< 10^{-16}$

$x_5 = x_4$.

Scale back: $2x_4$

Outline

Generic algorithm

If $f(a) * f(b) \leq 0$ and $f(x)$ is continuous on $[a, b]$, then $f(x)$ has a root on $[a, b]$.

```
while (b-a)>tol
    m = (a+b)/2;
    if f(a)*f(m)<=0
        b = m;
    else
        a = m;
    end;
end;
r = (a + b)/2;
```

Generic algorithm

Two problems in the generic algorithm:

The while-loop may not terminate.

Generic algorithm

Two problems in the generic algorithm:

The while-loop may not terminate.

When `a` and `b` are two neighboring floating-point numbers and `(b-a)>tol`, `(a+b)/2` is rounded to either `a` or `b`.

Generic algorithm

Two problems in the generic algorithm:

The while-loop may not terminate.

When `a` and `b` are two neighboring floating-point numbers and `(b-a)>tol`, `(a+b)/2` is rounded to either `a` or `b`.

Redundant function evaluations.

An improved algorithm

```
fa = f(a);
while (b-a)>tol + eps*max(|a|,|b|)
   m = (a + b)/2;
   fm = f(m);
   if fa*fm<=0
      b = m;
   else
      a = m; fa = fm;
   end;
end;
r = (a + b)/2;
```

An improved algorithm

```
fa = f(a);
while (b-a)>tol + eps*max(|a|,|b|)
   m = (a + b)/2;
   fm = f(m);
   if fa*fm<=0
      b = m;
   else
      a = m; fa = fm;
   end;
end;
r = (a + b)/2;
```

Note: `eps*max(|a|,|b|)` is about the distance between two consecutive floating-point numbers near `max(|a|,|b|)`. (ulp)

Convergence

Since $b_k - a_k \leq (b-a)/2^k$, $x_* \in [a_k, b_k]$, and $x_k = (a_k + b_k)/2$, we have

$$e_k = |x_k - x_*| \leq \frac{b_k - a_k}{2} = \frac{b-a}{2^{k+1}} \rightarrow 0$$

In this case, $e_{k+1} \leq 0.5 e_k$.
Improve accuracy by 1 bit per iteration or 1 decimal digit for every three or so iterations.

Convergence

In general, linear convergence rate:

$$e_{k+1} \leq ce_k$$

for some constant $c < 1$.

Convergence

In general, linear convergence rate:

$$e_{k+1} \leq ce_k$$

for some constant $c < 1$.

Difficulty: Locate the interval $[a, b]$.

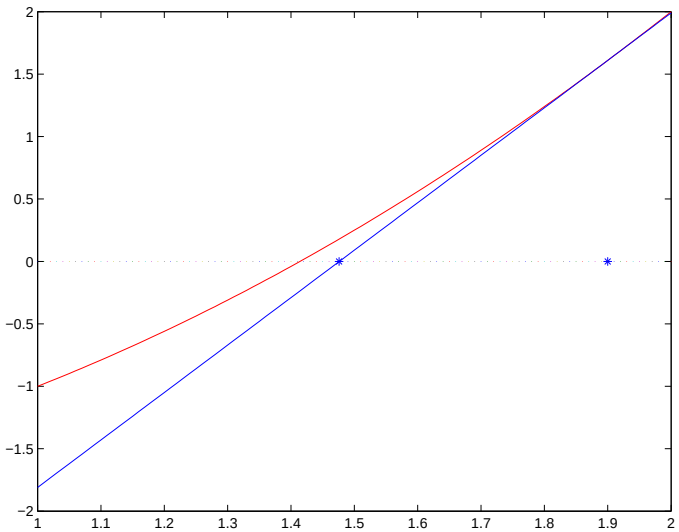
Outline

Idea

The tangent line of $f(x)$ at x_c:

$$y = f(x_c) + (x - x_c)f'(x_c)$$

Set $y = 0$



Newton's method

Newton's method

$$x_+ = x_c - \frac{f(x_c)}{f'(x_c)}$$

Newton's method

Newton's method

$$x_+ = x_c - \frac{f(x_c)}{f'(x_c)}$$

Example.

Square root problem revisited, find a zero of $f(x) = x^2 - A$

$$x_+ = x_c - \frac{x_c^2 - A}{2x_c} = x_c - \frac{1}{2}\left(x_c - \frac{A}{x_c}\right)$$

Complex case

Example

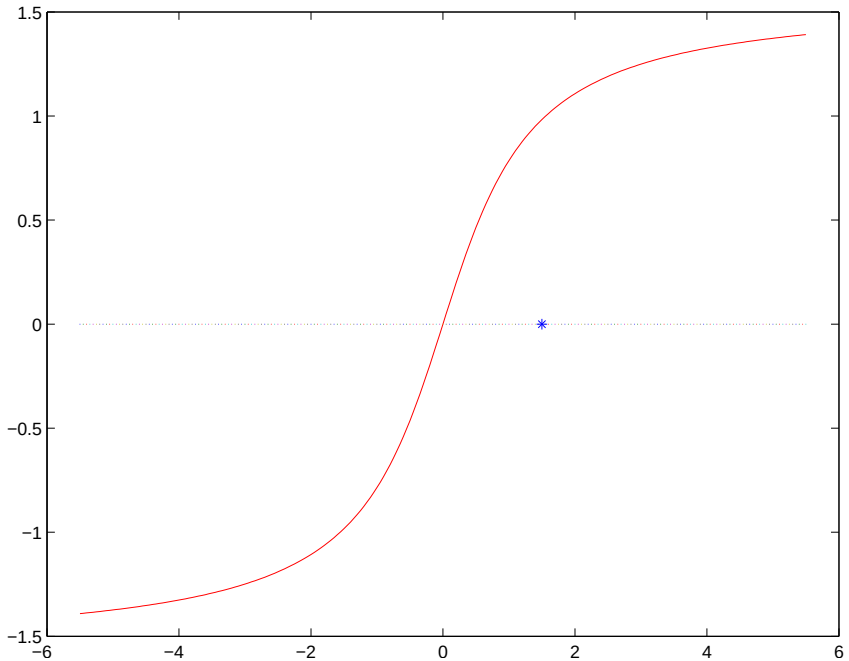
$f(x) = x^2 + x + 1$ (zeros $(-1 \pm i\sqrt{3})/2$)

i	x_i	error
0	i	5.2×10^{-1}
1	$-0.40000 + 0.80000i$	1.2×10^{-1}
2	$-0.50769 + 0.86154i$	8.9×10^{-3}
3	$-0.49996 + 0.86600i$	4.6×10^{-5}
4	$-0.50000 + 0.86603i$	1.2×10^{-9}
5	$-0.50000 + 0.86603i$	converge

$x_6 = x_5$

Convergence

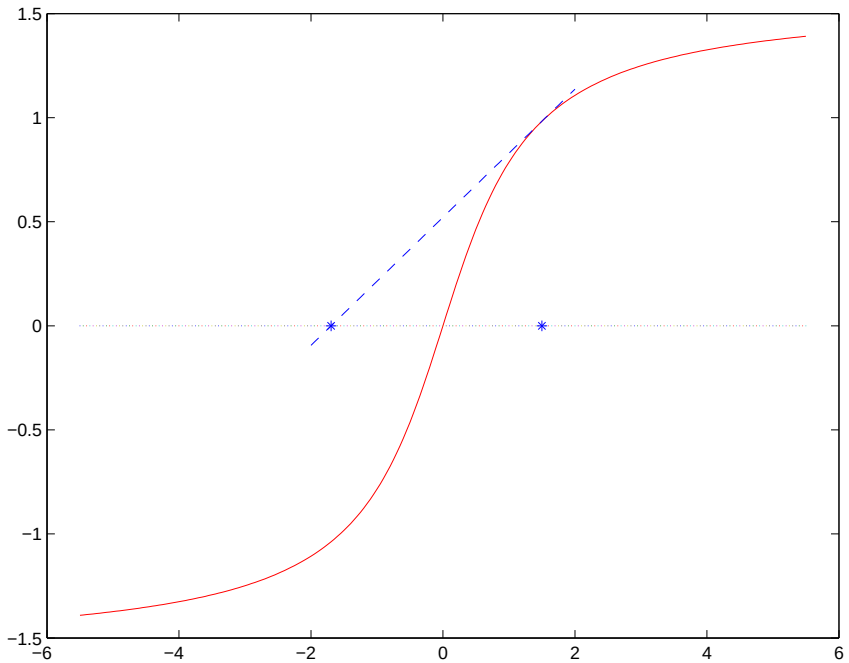
No guarantee of convergence (unlike bisection).
For example, $f(x) = \text{atan}(x)$, $x_+ = x_c - (1 + x_c^2)\text{atan}(x_c)$
$x_0 = 1.5$ (> 1.3917)



Convergence

No guarantee of convergence (unlike bisection).
For example, $f(x) = \text{atan}(x)$, $x_+ = x_c - (1 + x_c^2)\text{atan}(x_c)$

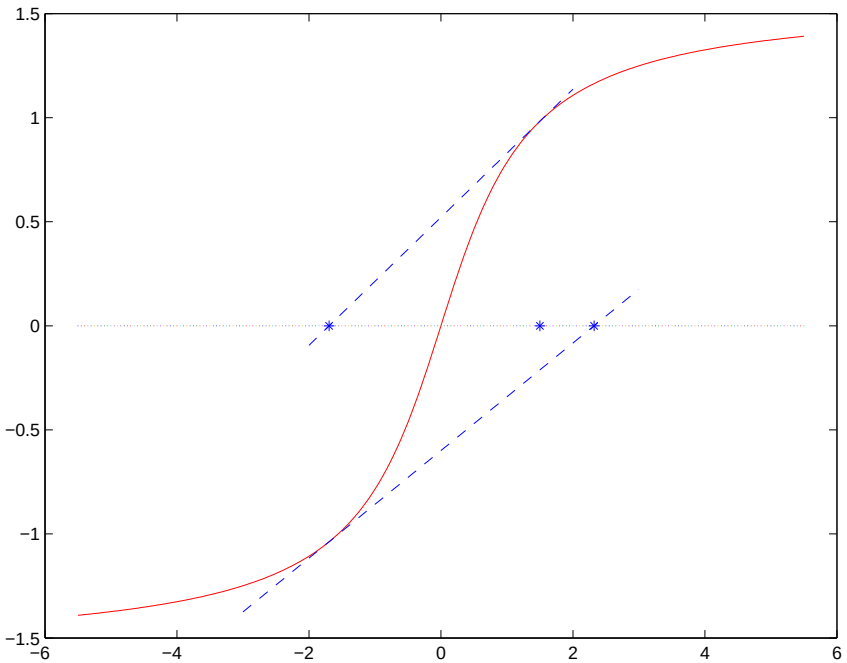
$x_1 = -1.6941$



Convergence

No guarantee of convergence (unlike bisection).
For example, $f(x) = \text{atan}(x)$, $x_+ = x_c - (1 + x_c^2)\text{atan}(x_c)$

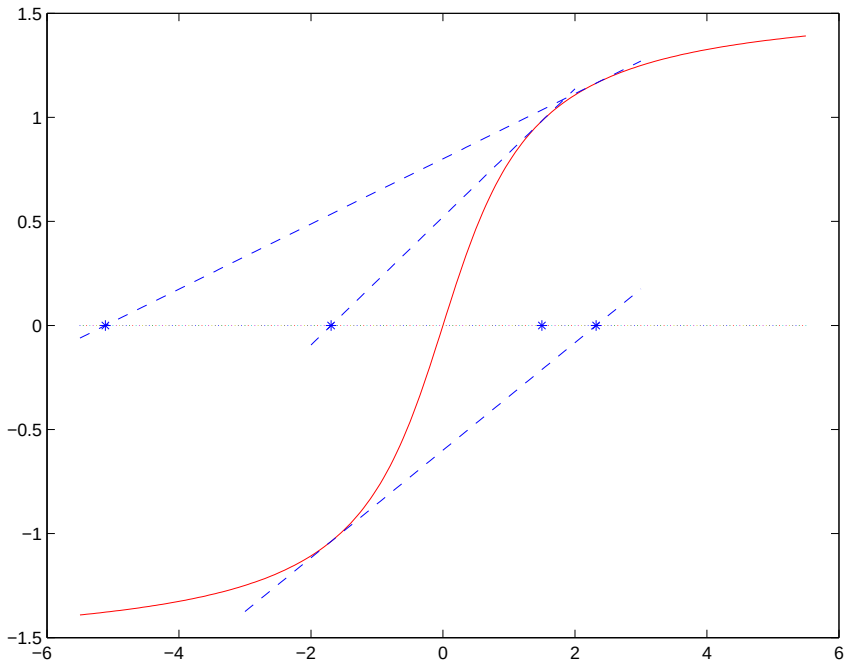
$x_2 = 2.3211$



Convergence

No guarantee of convergence (unlike bisection).
For example, $f(x) = \text{atan}(x)$, $x_+ = x_c - (1 + x_c^2)\text{atan}(x_c)$

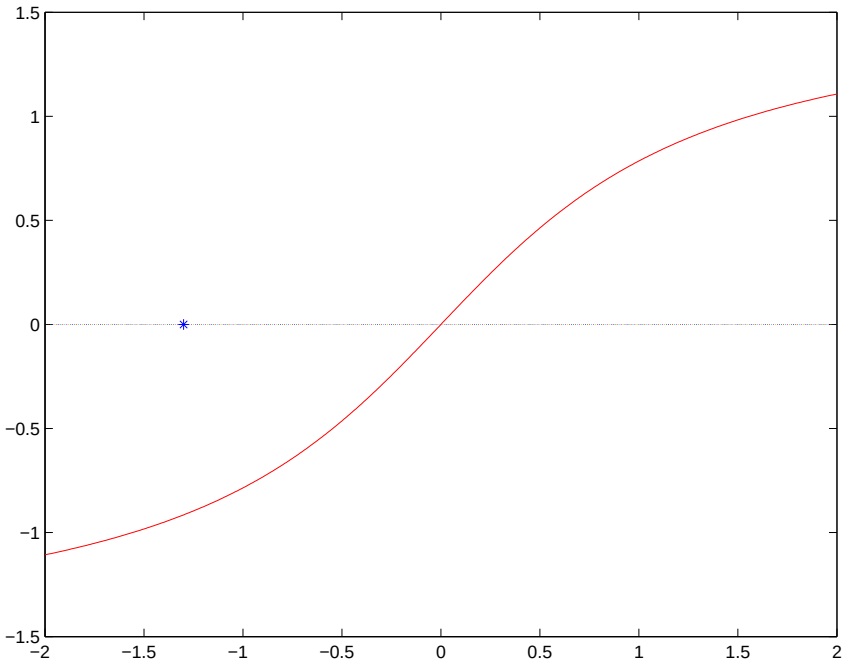
$x_3 = -5.1141$



Convergence

$f(x) = \text{atan}(x)$, $x_+ = x_c - (1 + x_c^2)\text{atan}(x_c)$

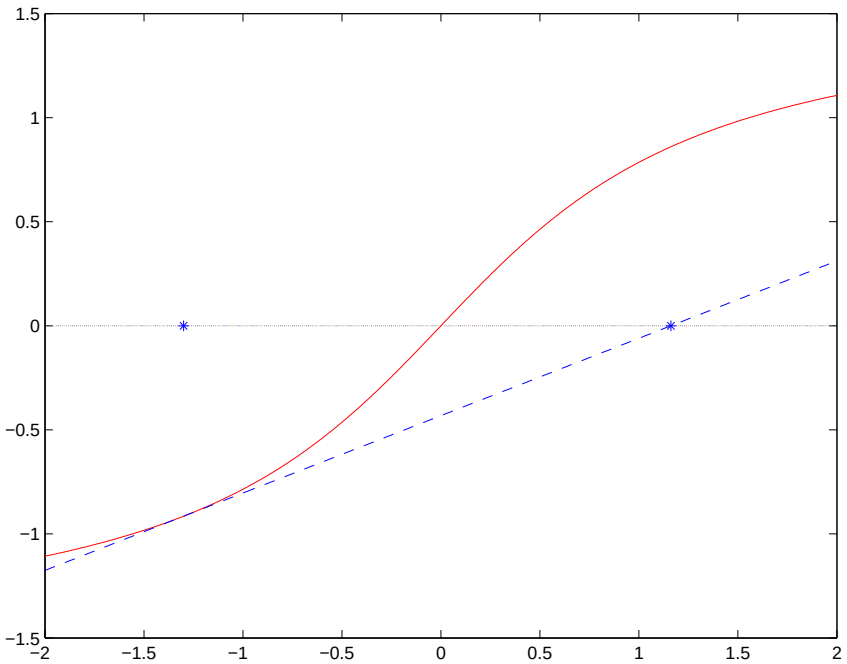
$x_0 = -1.3$ ($|-1.3| < 1.3917$)



Convergence

$f(x) = \text{atan}(x)$, $x_+ = x_c - (1 + x_c^2)\text{atan}(x_c)$

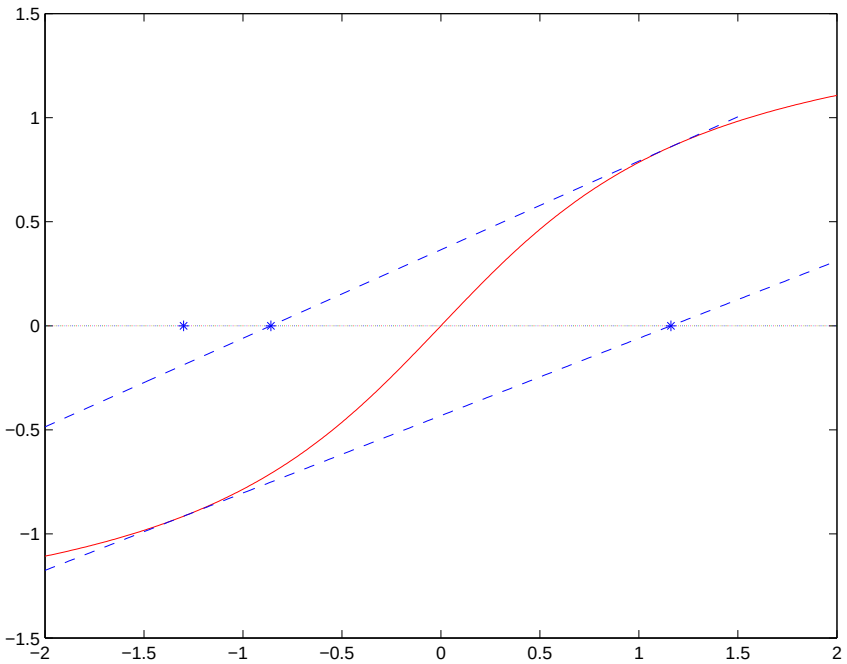
$x_1 = 1.1616$



Convergence

$f(x) = \text{atan}(x)$, $x_+ = x_c - (1 + x_c^2)\text{atan}(x_c)$

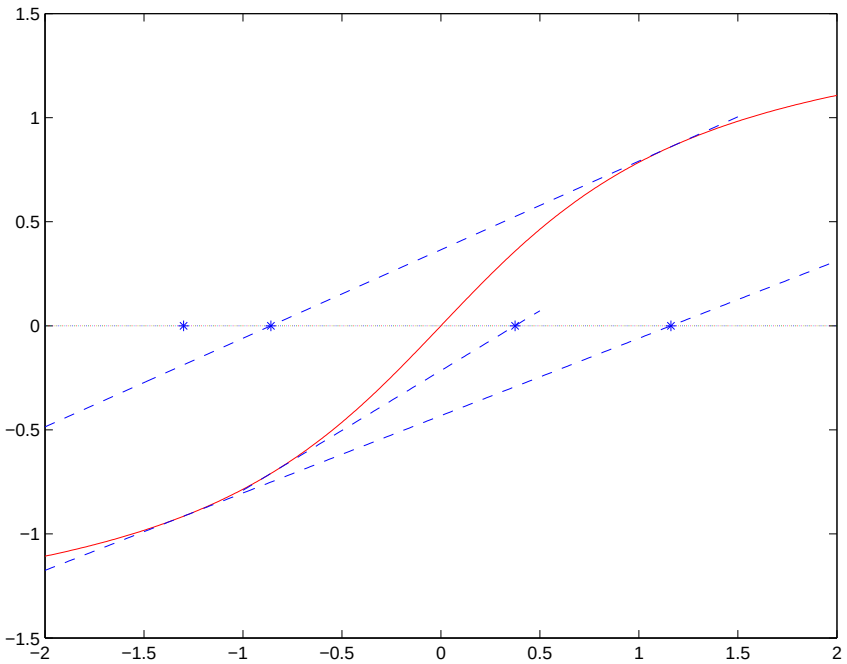
$x_1 = -0.8589$



Convergence

$f(x) = \text{atan}(x)$, $x_+ = x_c - (1 + x_c^2)\text{atan}(x_c)$

$x_1 = 0.3742$



Convergence

Conditions for convergence (qualitative):

- x_0 close enough to x_*
- $f'(x)$ does not change sign near x_*
- $f(x)$ is not too nonlinear near x_*

Newton's method is a *local* method.

Convergence

Conditions for convergence (qualitative):

- x_0 close enough to x_*
- $f'(x)$ does not change sign near x_*
- $f(x)$ is not too nonlinear near x_*

Newton's method is a *local* method.

Difficulty: Finding x_0.

Hybrid methods

Combining bisection and Newton's methods

Bracketing interval $[a, b]$, and $x_c = a$ or b
if $x_+ = x_c - f(x_c)/f'(x_c) \in [a, b]$
  bracketing interval $[a, x_+]$ or $[x_+, b]$
else
  $m = (a + b)/2$;
  bracketing interval $[a, m]$ or $[m, b]$

Termination criteria: Any one of

- $(b_k - a_k) < \delta$
- $|f(x_c)| < \delta$
- Too many function evaluations

Avoiding derivatives

Approximation

$$f'(x_c) \approx \frac{f(x_c + \delta_c) - f(x_c)}{\delta_c}$$

Avoiding derivatives

Approximation

$$f'(x_c) \approx \frac{f(x_c + \delta_c) - f(x_c)}{\delta_c}$$

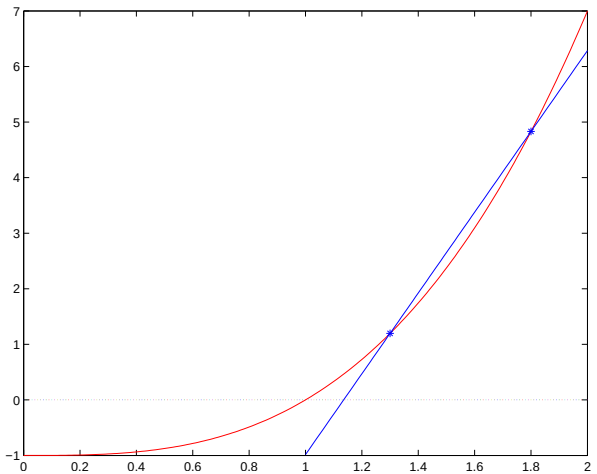
Choice of δ_c

Example. Secant method ($\delta_c = x_- - x_c$)

$$f'(x_c) \approx \frac{f(x_c) - f(x_-)}{x_c - x_-}$$

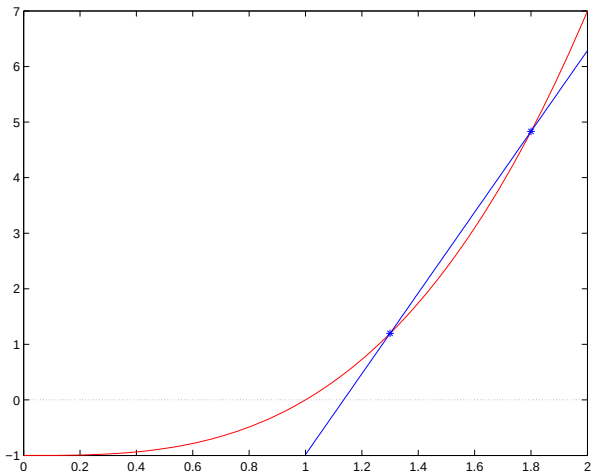
Secant method

$$x_+ = x_c - \frac{x_c - x_-}{f(x_c) - f(x_-)} f(x_c)$$



Secant method

$$x_+ = x_c - \frac{x_c - x_-}{f(x_c) - f(x_-)} f(x_c)$$



Usually, the convergence rate (if it converges) is $(1 + \sqrt{5})/2 \approx 1.6$

$e_{k+1} \leq c e_k^{1.6}$, superlinear, between quadratic and linear.

Zeros of a polynomial

Finding the zeros of a polynomial

$$p = x^n + c_{n-1}x^{n-1} + \ldots + c_1 x^1 + c_0$$

Many methods were proposed.

Zeros of a polynomial

Finding the zeros of a polynomial

$$p = x^n + c_{n-1}x^{n-1} + \ldots + c_1 x^1 + c_0$$

Many methods were proposed.

The eigenvalues of its companion matrix

$$C(p) = \begin{bmatrix} 0 & 0 & \cdots & 0 & -c_0 \\ 1 & 0 & \cdots & 0 & -c_1 \\ 0 & 1 & \cdots & 0 & -c_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & -c_{n-1} \end{bmatrix}, \quad \det(xI - C(p)) = p$$

Example

The zeros of the polynomial

$$x^3 - 1$$

are the eigenvalues of

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Example

The zeros of the polynomial

$$x^3 - 1$$

are the eigenvalues of

$$\begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

One real and two complex conjugate eigenvalues.

Note

How to compute the eigenvalues of a matrix?

Finding the zeros of a polynomial used to be the way of finding the eigenvalues of a matrix A.

Note

How to compute the eigenvalues of a matrix?

Finding the zeros of a polynomial used to be the way of finding the eigenvalues of a matrix A.

Text book method:

The eigenvalues of a matrix A are the zeros of its characteristic polynomial $\det(\lambda I - A)$.

Note

Now, we have efficient and reliable methods for computing eigenvalues of a matrix.

QR method, John G.F. Francis and Vera N. Kublanovskaya, late 1950s.

We find the zeros of a polynomial by computing the eigenvalues of its companion matrix.

Outline

Problem setting

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \\ f_2(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, \ldots, x_n) &= 0 \end{aligned}$$

Denote

$$\mathbf{f}(\mathbf{x}) = 0$$

f: vector-valued function
x: vector

Newton's method

$$\mathbf{x}_+ = \mathbf{x}_c + \mathbf{s}_c$$

where $\mathbf{s}_c$ is the solution of

$$\mathbf{f}(\mathbf{x}_c) + J(\mathbf{x}_c)\mathbf{s}_c = 0,$$

i.e., $\mathbf{s}_c = -J^{-1}(\mathbf{x}_c)\mathbf{f}(\mathbf{x}_c)$, where $J(\mathbf{x}_c)$ is the Jacobian of $\mathbf{f}$ at $\mathbf{x}_c$:

$$J(\mathbf{x}) = \left[\frac{\partial f_i}{\partial x_j}\right] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Example

A system of nonlinear equations

$$\begin{cases} x_1^2 - x_2^2 = 0 \\ 2x_1x_2 = 1, \end{cases}$$

with starting point

$$x_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Solution: $x_1 = x_2 = 1/\sqrt{2}$

Example

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} x_1^2 - x_2^2 \\ 2x_1x_2 - 1 \end{bmatrix}$$

The Jacobian is

$$J(x) = \begin{bmatrix} 2x_1 & -2x_2 \\ 2x_2 & 2x_1 \end{bmatrix}$$

and

$$J(x_0) = \begin{bmatrix} 0 & -2 \\ 2 & 0 \end{bmatrix}$$

Example

Step 1:

$$x_1 = x_0 - J^{-1}(x_0) f(x_0)$$

Example

Step 1:

$$x_1 = x_0 - J^{-1}(x_0) f(x_0)$$

Solving for d_0 in $J(x_0)d_0 = f(x_0)$, we have

$$d_0 = \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix}$$

Thus

$$x_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} -0.5 \\ 0.5 \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

Example

Step 2:

$$x_2 = x_1 - J^{-1}(x_1) f(x_1)$$

$$J(x_1) = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}, \qquad f(x_1) = \begin{bmatrix} 0 \\ -0.5 \end{bmatrix}$$

Example

Solving for d_1 in $J(x_1)d_1 = f(x_1)$, we have

$$d_1 = \begin{bmatrix} -0.25 \\ -0.25 \end{bmatrix}$$

Thus

$$x_2 = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} - \begin{bmatrix} -0.25 \\ -0.25 \end{bmatrix} = \begin{bmatrix} 0.75 \\ 0.75 \end{bmatrix}$$

Avoiding derivatives

The jth column of $J(\mathbf{x})$

$$\begin{bmatrix} \frac{\partial f_1}{\partial x_j} \\ \vdots \\ \frac{\partial f_n}{\partial x_j} \end{bmatrix} = \frac{\partial \mathbf{f}}{\partial x_j}$$

can be approximated by the difference

$$\frac{\mathbf{f}(x_1, ..., x_j + \delta, x_{j+1}, ..., x_n) - \mathbf{f}(x_1, ..., x_n)}{\delta}$$

Outline

Problem setting

$$\min_{\mathbf{x} \in S} f(\mathbf{x}) \qquad \text{or} \qquad \max_{\mathbf{x} \in S} f(\mathbf{x})$$

$\mathbf{x}$: vector $f(\mathbf{x})$: objective function and real-valued
S: support

Problem setting

$$\min_{\mathbf{x}\in S} f(\mathbf{x}) \qquad \text{or} \qquad \max_{\mathbf{x}\in S} f(\mathbf{x})$$

$\mathbf{x}$: vector $f(\mathbf{x})$: objective function and real-valued
S: support

Find a zero of the gradient

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} \end{bmatrix}$$

Newton's method

View the gradient $\nabla f(\mathbf{x})$ as a vector-valued function and apply the Newton's method for solving nonlinear systems.

At current $\mathbf{x}_c$, find the correction $\mathbf{s}_c$:
$\mathbf{x}_+ = \mathbf{x}_c + \mathbf{s}_c$
where $\mathbf{s}_c$ is the solution of

$$\nabla f(\mathbf{x}_c) + H(\mathbf{x}_c)\mathbf{S}_c = 0.$$

The matrix $H(\mathbf{x}_c)$ (the Jacobian of the gradient at $\mathbf{x}_c$) is called the Hessian of f at $\mathbf{x}_c$ ($\nabla^2 f(\mathbf{x}_c)$):

$$H_{i,j} = \frac{\partial^2 f}{\partial x_i \, \partial x_j}.$$

Example

Minimizing $f : R^2 \rightarrow R$:

$$f(\mathbf{x}) = \frac{x_1^3}{3} - x_1 x_2^2 + x_2.$$

Perform one iteration of Newton's method for minimizing f using the starting point

$$\mathbf{x}_0 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Example

Apply the Newton's method for finding a zero of the gradient

$$\nabla f(\mathbf{x}) = \begin{bmatrix} x_1^2 - x_2^2 \\ -2x_1x_2 + 1 \end{bmatrix}$$

The Hessian

$$H(\mathbf{x}) = \nabla^2 f(\mathbf{x}) = \begin{bmatrix} 2x_1 & -2x_2 \\ -2x_2 & -2x_1 \end{bmatrix}$$

Example

Step 1:

$$
\begin{aligned}
\mathbf{x}_1 &= \mathbf{x}_0 - \nabla^2 f(\mathbf{x}_0)^{-1} \nabla f(\mathbf{x}_0) \\
&= \begin{bmatrix} 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 & -1/2 \\ -1/2 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} 1/2 \\ 1/2 \end{bmatrix}
\end{aligned}
$$

Outline

Software Packages

IMSL	zporc, zplrc, zpocc
MATLAB	roots, fzero
NAG	c02agf, c02aff
NAPACK	czero
Octave	fsolve

Summary

- Issues in an iterative method: Initialization, convergence and rate of convergence, termination. The example of computing square root
- Bisection method: Numerical termination problem
- Newton's method: Initial value, convergence problems
- Newton's method for systems of nonlinear equations, Jacobian matrix
- Newton's method for minimization, gradient and Hessian.

References

[1] George E. Forsyth and Michael A. Malcolm and Cleve B. Moler. Computer Methods for Mathematical Computations. Prentice-Hall, Inc., 1977.
Ch 7.