

Nonlinear Equations

(in 1D)

Numerical Analysis Spring 2021
A. Donev, Courant

Consider solving a nonlinear equation like

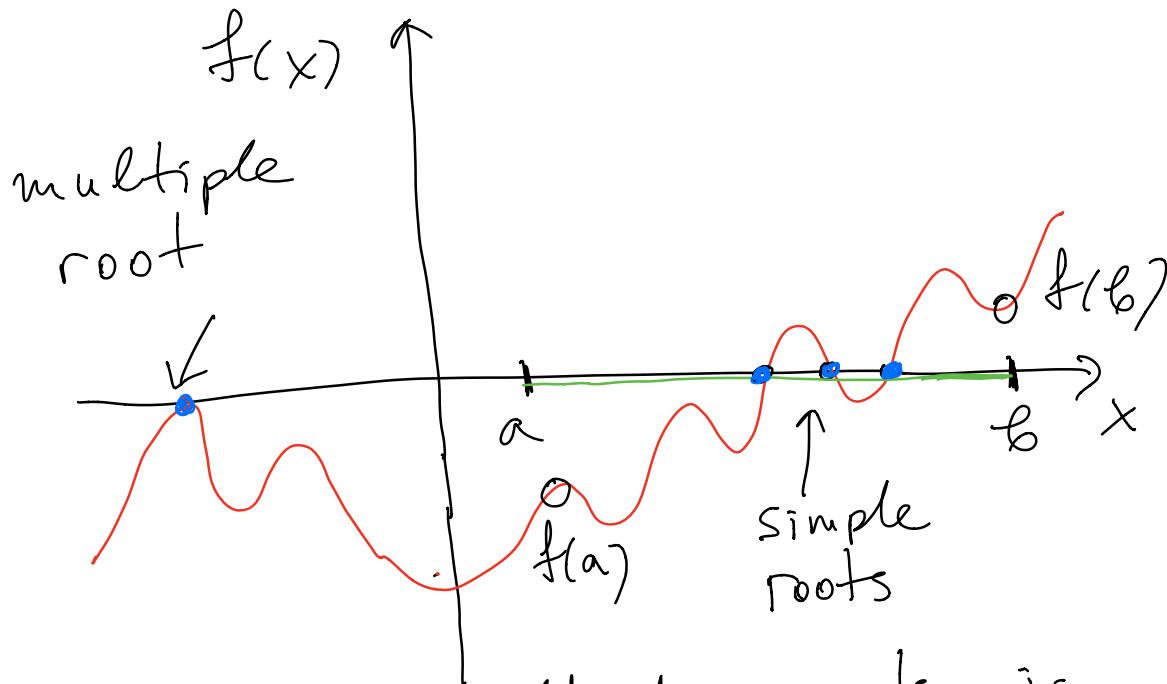
$$\cos x + x^2 - 7 = 0$$

No closed form solution exists,
so the only thing we can do
is to solve numerically (on a
computer)

Solve $f(x) = 0$ on $x \in [a, b]$

By this we mean find as
accurately as possible at least
one solution in $[a, b]$, assuming
one exists.

①



Finding multiple roots is harder, so we focus on simple roots & assume there is at least one root in $[a, b]$ & f is continuous

{ Theorem If f is continuous
 on $[a, b]$ and $f(a) \cdot f(b) < 0$
 (i.e., $f(a)$ & $f(b)$ have opposite sign)
 then $\exists x \in (a, b)$ s.t. $f(x) = 0$

(2)

"Proof": It's "obvious" but if you insist apply the Intermediate Value Theorem \square

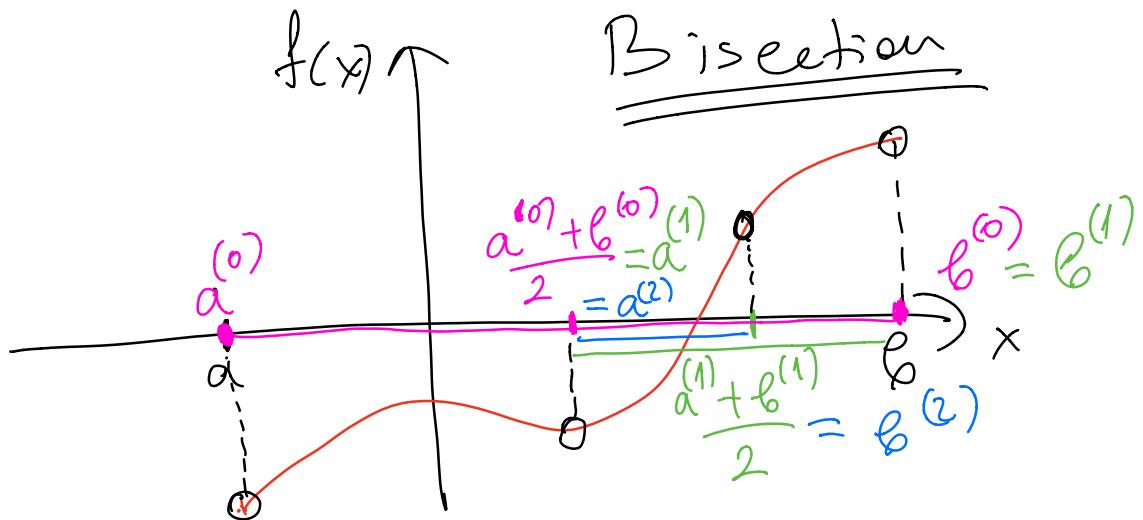
Now, assume that we are given an $f(x)$, meaning, given a real x we can evaluate $f(x)$ using some computer code, and an interval $[a, b]$ such that $f(a) \cdot f(b) < 0$

(Q1) Why don't I say $f(a)f(b) \leq 0$?

Can we devise an algorithm that will give us a root of f on $[a, b]$?

Answer: Yes, use bisection

③



Algorithm Bisection

Input $[a, b]$, $n \in \mathbb{Z}^+$
 Output x such that $f(x) \approx 0$
 approximately
 equal

Set $a^{(0)} = a$, $b^{(0)} = b$

For $k = 0, 1, 2, \dots$, $n-1$ do

$$x^{(k)} = \frac{a^{(k)} + b^{(k)}}{2}$$

$$f^{(k)} = f(x^{(k)})$$

(4)

If $f^{(k)} f(a^{(k)}) < 0$
 $b^{(k+1)} = x^{(k)} ; a^{(k+1)} = a^{(k)}$

else $b^{(k+1)} = b^{(k)} ; a^{(k+1)} = x^{(k)}$

end if

end for

Return

or $x \in [a^{(n)}, b^{(n)}]$

$$x \approx x = \frac{a^{(n)} + b^{(n)}}{2}$$

[Matlab Demo Bisection.m]

How accurate is x ?

What is absolute error

$$\epsilon^{(n)} = |x^{(n)} - x|$$

where $f(x) = 0$

(5)

We know
 $x \in [a^{(n)}, b^{(n)}]$

$$\Rightarrow |x^{(n)} - x| \leq \frac{b^{(n)} - a^{(n)}}{2}$$

Observe: $b^{(n)} - a^{(n)} = \frac{b^{(0)} - a^{(0)}}{2^n}$

$$\Rightarrow E^{(n)} = |x^{(n)} - x| \leq \frac{b - a}{2^{n+1}}$$

↑

error estimate

This means the error approximately halves each iteration. This is guaranteed, which means the method is very robust, but it's not very fast.

⑥

$$\text{given } \epsilon \quad \leftarrow \frac{b-a}{2^{n+1}}$$

$$2^{n+1} \leftarrow \frac{b-a}{\epsilon}$$

$$n+1 = \lfloor \log_2 \left(\frac{b-a}{\epsilon} \right) \rfloor$$

"floor" in Matlab

$$\Rightarrow n = \lceil \log_2 \left(\frac{b-a}{\epsilon} \right) \rceil$$

ceil in Matlab

So given a target error tolerance ϵ we can directly estimate how many iterations to take.

[numerical analyst job complete]

Newton & Secant

Can we do better than bisection? Bisection only used the sign of f , not the value of $f(x)$.

"Definition": A function $f(x)$ is a **smooth function** on $[a, b]$ if it can be approximated well by a low order polynomial (linear, quadratic, cubic) over the whole interval $[a, b]$

③

In most of this class,
we will assume to work
with functions that are
sufficiently smooth. This
is more than just being
sufficiently differentiable,
which we may require as
an assumption. Smoothness
also requires high-order
derivatives not to be very
large. In some sense

[See Functional Analysis]

Our error / convergence estimates
will only be accurate
for smooth $f(x)$. ⑨

Note: If $f(x)$ is continuously differentiable, it will be almost linear over a sufficiently small interval $[a, b]$ (think calculus as $b-a \rightarrow dx$)

Idea for "all" of NA:
Approximate smooth functions by low-degree polynomials

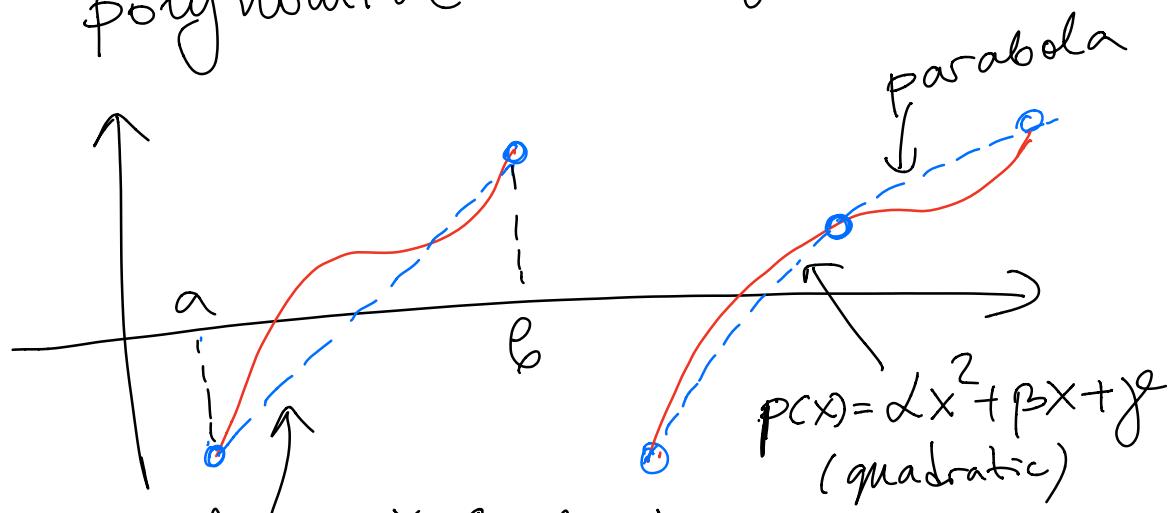
If $f(x) \approx p(x)$ on $[a, b]$
then we can solve

$p(x) = 0$
instead. If p is
a linear or quadratic
function then we have
explicit solution! (10)

How do we approximate
 $f(x)$ by $p(x)$?

Two ideas common in all
of numerical analysis:

- ① Evaluate $f(x)$ at a few points & fit a polynomial through them:



$$p(x) = f(a) + \frac{x-a}{b-a} f(b)$$

↑ linear

②

② Approximate $f(x)$ by
 its Taylor series around
 the current guess for the
 answer

Reminder

If $f(x)$ is infinitely
 differentiable at x_0 ,

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} (x-x_0)^n$$

$$= f(x_0) + f'(x_0)(x-x_0)$$

$$+ \frac{1}{2} f''(x_0)(x-x_0)^2$$

+ ...

12

Taylor series with
remainder

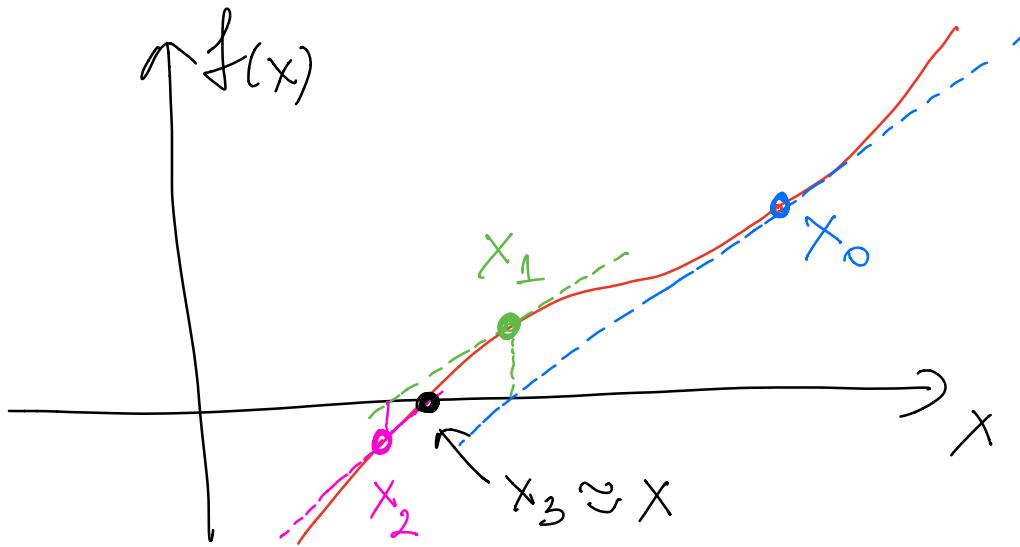
$$f(x) = \sum_{n=0}^k \frac{f^{(n)}(x_0)}{n!} (x-x_0)^n + \frac{f^{(k+1)}(\xi)}{(k+1)!} (x-x_0)^{k+1}$$

for some ξ between x_0 & x

Newton's method

Approximate $f(x)$ by it's
tangent at x_k , i.e., first-
order Taylor series

(13)



$$f(x) \approx f_k(x) = f(x_k) + f'(x_k)(x - x_k)$$

Solve $f_k(x) = 0 \Rightarrow$

$$x \approx x_k - \frac{f(x_k)}{f'(x_k)}$$

Newton's method

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

(14)

E.g. $f(x) = x^2 - c$
 (square root calculation)

$$f'(x) = 2x \Rightarrow$$

$$x_{k+1} = x_k - \frac{x_k^2 - c}{2x_k}$$

$$= \frac{x_k}{2} + \frac{c}{2x_k}$$

which is the Babylonian method!

Remember that the method converged fast. Why?

General idea: When analyzing the error/convergence of method based on Taylor series, use Remainder Theorem

(15)

$$\text{Error} \quad e_k = x_k - x$$

$$f(x) = 0 = f(x_k) + (x-x_k)f'(x_k)$$

remainder $\rightarrow +\frac{1}{2}(x-x_k)^2 f''(\xi)$

ξ between x_k and x

Divide both sides by $f'(x_k) \neq 0$
 & rearrange

$$\underbrace{\left(x_k - \frac{f(x_k)}{f'(x_k)} \right) - x}_{e_{k+1}} = -\frac{1}{2} \frac{(x-x_k)^2 f''(\xi)}{f'(x_k)}$$

$$x_{k+1} - x = e_{k+1} = -\frac{1}{2} e_k^2 \cdot \frac{f''(\xi)}{f'(x_k)}$$

$$\frac{e_{k+1}}{e_k^2} = -\frac{f''(\xi)}{2f'(x_k)}$$

(16)

If the method converges,
=

$$x_k \rightarrow x \\ \text{and } \underline{\xi} \rightarrow x \Rightarrow$$

$$\frac{|e_{k+1}|}{e_k^2} \rightarrow \frac{f''(x)}{2f'(x)} = c$$

$$\Rightarrow |e_{k+1}| \approx c \cdot e_k^2$$

Error gets squared every iteration, not halved like for bisection. This leads to fast convergence after the error is small

$$\text{If } e_k \ll 1 \Rightarrow e_k^2 \ll e_k \quad (17)$$

much less than

This explains why the Babylonian method converged fast.

But when does Newton method converge?

General answer hard to give but we can give a sufficient condition

$$\text{We want } |e_{k+1}| < |e_k|$$

(error decreases)

$$\left| \frac{f''(\xi)}{2f'(x_k)} \right| e_k^2 < |e_k|$$

$$\text{If } |e_k| < \left| \frac{2f'(x_k)}{f''(\xi)} \right|$$

then error decreases

(18)

I imagine that

$$\left| \frac{f''(y_1)}{f'(y_2)} \right| \leq A \quad \forall y_1, y_2 \in I$$
$$I = [x - \delta, x + \delta]$$

(see Theorem 1.8 in Suli)

and $x_0 \in I \Rightarrow$

$$|e_0| = |x_0 - x| < \delta$$

$$\text{If } |e_0| < \frac{1}{A} < \left| \frac{f'(x_0)}{f''(\xi)} \right|$$

$$\Rightarrow |e_1| < \frac{|e_0|}{2} \quad \text{and error}$$

decreases strictly by at least
half and eventually much faster

(15)

Conclusion :

If we give Newton's method an initial guess closer to a simple root x than $\approx \left| \frac{f'(x)}{f''(x)} \right|$, it will converge rapidly.

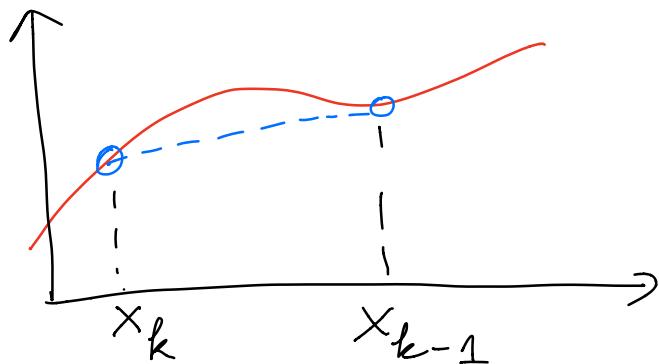
Otherwise, it may converge very slowly, erratically, or diverge.

Idea : Use bisection first to find an initial guess that is good & then switch to Newton

This is called safe guarded Newton method but we will not go into the details in class.

Secant method

Newton's method was based on approximating $f(x)$ with its first-order Taylor series (tangent). How about approximating $f(x)$ based on two points?



(21)

$$f(x) \approx p(x) = f(x_k) + \frac{x - x_k}{x_{k-1} - x_k} (f(x_{k-1}) - f(x_k))$$

Check:

$$\begin{cases} p(x_k) = f(x_k) \\ p(x_{k-1}) = f(x_{k-1}) \end{cases}$$

$$f(x_{k+1}) \approx p(x_{k+1}) = 0$$

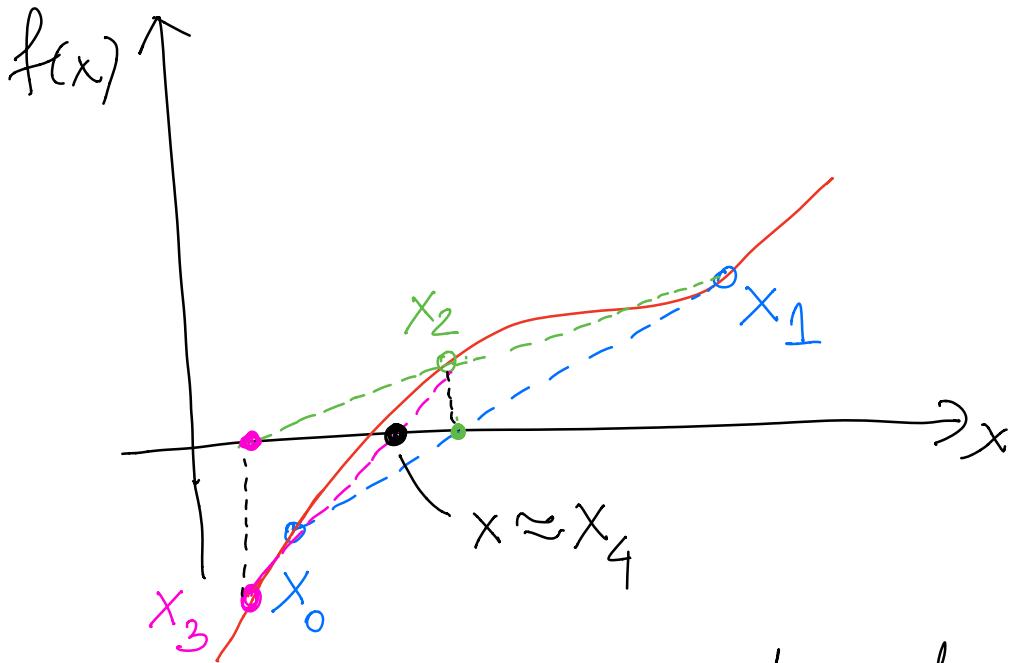
Solve for x_{k+1} (linear eq.)

$$x_{k+1} = x_k - \frac{f_k}{f_k - f_{k-1}} \left(\frac{x_k - x_{k-1}}{f_k - f_{k-1}} \right)$$

Secant method

Here $f_k = f(x_k)$ (notation)

(22)



One way to think of
secant method is as
Newton's method with
the approximation

$$f'(x_k) \approx \frac{f(x_{k-1}) - f(x_k)}{x_{k-1} - x_k}$$

which is good if f is
smooth and $x_{k-1} \approx x_k$. 23
Called finite-difference approximation

Note we need two points to start the iteration.

It is therefore very natural to combine secant with bisection: instead of using midpoint use secant point as new guess. This method of Regula Falsi is guaranteed to converge to a root but maybe it is faster than bisection?

[Demo Secant.m]

Conclusion: True secant converges faster than bisection or Regula Falsi.

But how much faster?

Bisection: $|e_{k+1}| \leq \frac{|e_k|}{2}^1$
(linear)

Newton: $|e_{k+1}| \rightarrow \left| \frac{f''(x)}{2f'(x)} \right| |e_k|^2$
(quadratic)

Secant (we will not prove):

$$|e_{k+1}| \rightarrow M \cdot |e_k|^q$$

where $q = \frac{1}{2}(1 + \sqrt{5}) \approx 1.6$
is Golden Ratio

We say bisection converges
at a linear rate, Newton at
quadratic rate, and secant at
②5

order of convergence ≈ 1.6

Aside: For secant method,
one can prove
(see (4.17) in "Practice" book)

$$|e_{k+1}| \rightarrow \left| \frac{f''(x)}{2f'(x)} \right| |e_k e_{k-1}|$$

which makes the relationship
with Newton's method more
obvious.

Homework

Try & (maybe) analyze
Steffensen's method

$$x_{k+1} = x_k - \frac{f_k^2}{f(x_k + f_k) - f_k} \quad (26)$$

Fixed-point iteration

Every iterative method
of the form

$$x_{k+1} = g(x_k)$$

that converges to $x_k \rightarrow x$
satisfies

$$x = g(x)$$

which for consistency must
be equivalent (imply) $f(x)=0$

For example, Babylonian
method for $x^2 - c = 0$ has

$$g(x) = \frac{1}{2} \left(x + \frac{c}{x} \right)$$

(27)

$$x = g(x) \Rightarrow \cancel{x} = \frac{1}{\cancel{x}} c$$

$$\Rightarrow x^2 = c \quad \text{as desired}$$

But also

$$x = x^2 + x - c = g(x) \quad \text{is equivalent to } x = \sqrt{c}.$$

So we have lot's of options to choose $g(x)$.

Every $g(x)$ gives us an iterative fixed-point method

$$x_{k+1} = g(x_k)$$

When does this method converge, and how fast?

(28)

For example, relaxation method

$$x_{k+1} = x_k - \lambda f(x_k)$$

where $\lambda \neq 0$ has

$$g(x) = x - \lambda f(x) = x \\ \text{iff } f(x) = 0$$

What is a good choice
of λ ?

Full analysis of fixed-point
in theory textbook. We will
follow a quicker analysis
(less general & precise).

Let's use Taylor series

$$\begin{aligned}x_{k+1} &= g(x_k) = g(x) + (x_k - x)g'(\xi_k) \\&= x + (x_k - x)g'(\xi_k)\end{aligned}$$

\Rightarrow

$$e_{k+1} = x_{k+1} - x = g'(\xi_k) e_k$$

If we want error to
decrease (converge to zero)

we want $|g'(\xi_k)| < 1$

$$|e_{k+1}| = |g'(\xi_k)| |e_k|$$

(30)

Theorem continuously differentiable

If $g \in C^1$ and

$$|g'(y)| < 1 \quad \forall y \in [x-\delta, x+\delta]$$

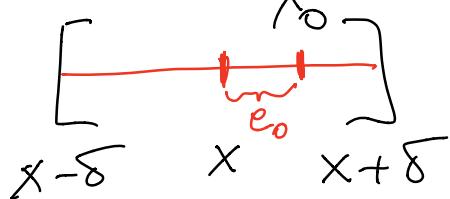
where $g(x) = x$ is a fixed point,

then fixed point iteration

starting with $x_0 \in [x-\delta, x+\delta]$

converges to x .

"Proof": $|e_0| < \delta$ and



$$|e_1| = |g(e_0)| |e_0|$$

$$\Rightarrow |e_1| < \delta \text{ and}$$

$$x_1 \in [x-\delta, x+\delta]$$

(31)

By induction, iterates
remain inside interval

$$[x-\delta, x+\delta]$$

and $|e_{k+1}| < L |e_k|$

where (Lipschitz constant)

$$L = \max_{y \in [x-\delta, x+\delta]} |g'(y)| < 1$$

$$\Rightarrow |e_k| < L^k |e_0|$$

with $0 \leq L < 1$

and thus $|e_k| \rightarrow 0$.

As we get closer to root

$$\frac{|e_{k+1}|}{|e_k|} \rightarrow |g'(x)|$$

(32)

Convergence is fastest if

$$|g'(x)| << 1$$

{ Ideally $g'(x) = 0 \Rightarrow$
Super-linear convergence

Go back to relaxation
method

$$g(x) = x - \lambda f(x)$$

$$g'(x) = 1 - \lambda f'(x) = 0$$

$$\Rightarrow \lambda = -\frac{1}{f'(x)} \Rightarrow$$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x)}$$

But of course we don't know x but if we converge

$$x_k \rightarrow x \text{ so } \leftarrow$$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

which is Newton's method!

Newton's: $g = x - \frac{f(x)}{f'(x)}$

$$g'(x) = f - \frac{f'(x)}{f'(x)} + \frac{f(x)f''(x)}{(f'(x))^2}$$

$$g'(x) = 0 \quad \text{if} \quad f(x) = 0$$

$$\text{and} \quad f'(x) \neq 0$$