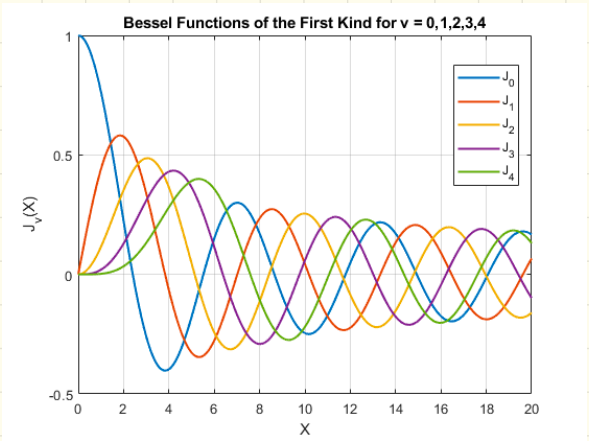
Chap 4 Roots of Nonlinear Equations

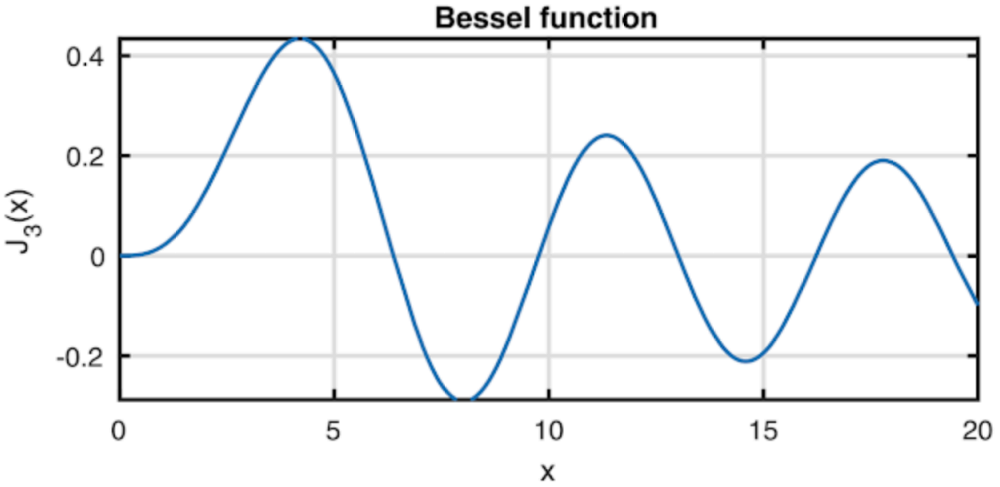
	Nonlinear problem	Linear problem
	$f(x) = 0$	$g(x) = 0$
Ex:	$J_m(\omega_{k,m} r)\cos(m\theta)\cos(c\omega_{k,m} t)$	$Ax = b$ $(3x+2=0)$
Idea:	Replace a problem you can't solve with an <u>approximate</u> one you can	Replace a problem you can't solve with one that you can solve (e.g. $Ax=b \Rightarrow LUx=b$)
Conv:	<u>Infinite</u> iteration even in exact arithmetics	<u>Finite</u> step

§ 4.1 The rootfinding problem

- Goal: Find r, such that $f(r) = 0$
  - r: root
  - f: a continuous scalar function ($\mathbb{R}^1$)
- MATLAB: "fzero"
- Example: 4.1.1
  - Displacement of the drumhead
  - Bassel function of the first kind (Hard to find the root!)



```
J3 = @(x) besselj(3,x);
fplot(J3,[0 20]), grid on
```



From the graph we see roots near 6, 10, 13, 16, and 19. We use `fzero` to find these roots accurately.

```
omega = [];
for guess = [6,10,13,16,19]
    omega = [omega;fzero(J3,guess)];
end
omega
```

Conditioning, Error, Residual

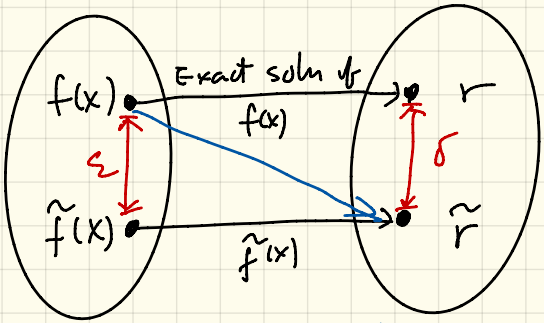
- The perturbated problem

  Eg. $f(x) \Rightarrow \tilde{f}(x) = f(x) + \varepsilon$ ← perturbation

  Root r s.t. $f(r)=0 \Rightarrow \tilde{r} = r + \delta$ s.t. $\tilde{f}(\tilde{r}) = 0$

- condition number

  $\kappa_r = \left|\frac{\delta}{\varepsilon}\right|$ as $\varepsilon \to 0$ (δ: change in output, ε: change in input)



By Taylor series

$$0 = f(r+\delta) + \varepsilon = f(r) + f'(r)\delta + \varepsilon + O(\delta^2)$$ (f(r) = 0; $O(\delta^2)$: small number can be approximated by 0)

$$\Rightarrow \kappa_r = \left|\frac{\delta}{\varepsilon}\right| \approx 1/|f'(r)|$$

(與 r 有關)

Recall that the absolute condition number for the evaluation of f at x is simply $|f'(x)|$. *The condition number of the rootfinding problem is equivalent to that of evaluating the inverse function.* The influence of $|f'(r)|$ on rootfinding is easily illustrated.

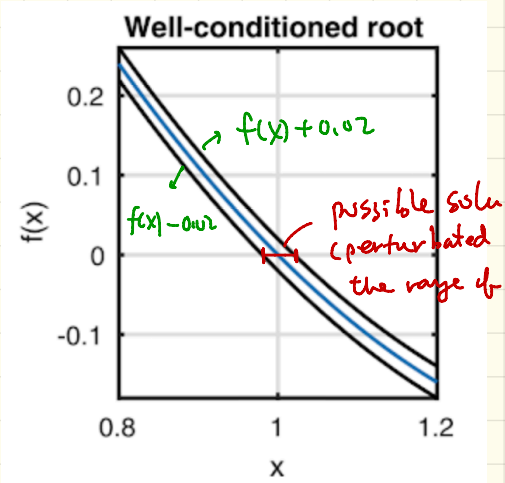
Example 4.1.2

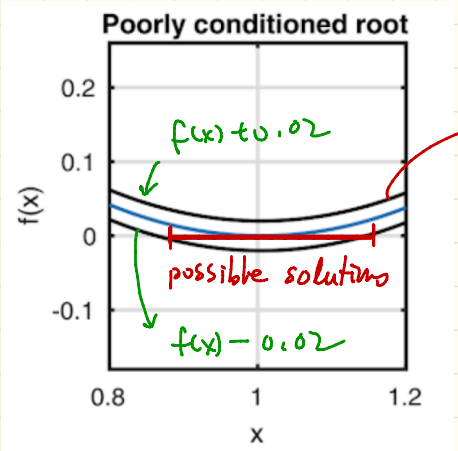
(1) $f(x) = (x-1)(x-2)$

$\Rightarrow r = 1,\ f'(r) = -1,\ \kappa_r = 1$

(2) $f(x) = (x-1)(x-1.01)$

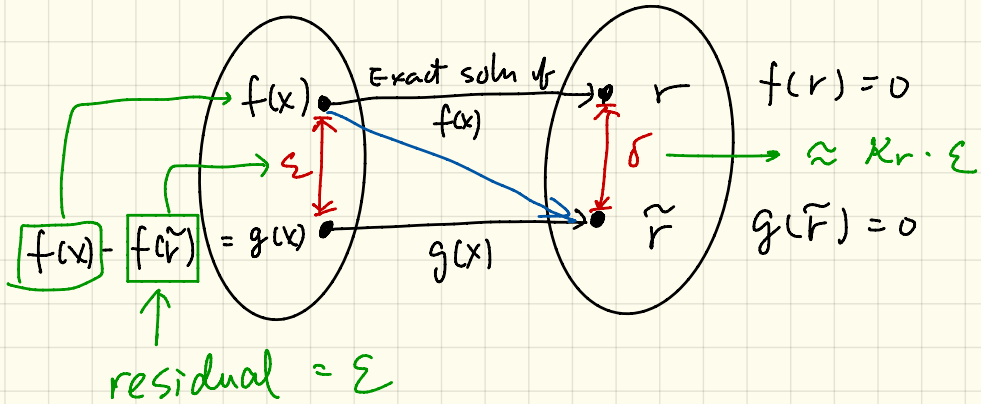
$\Rightarrow r = 1,\ f'(1) = -0.01,\ \kappa_r = 100$





- When $|f'(r)|$ is small at the root (Unknown!) (the cond. # is big)
  we may not get small computed error.

- Residual: $f(\tilde{r})$ → Computable!



small residual (i.e. small backward error)

Multiple roots

Let r be a root of $f(x)$ and $q(x) = \frac{f(x)}{(x-r)}$.

We have $f(x) = (x-r) \cdot q(x)$

$\Rightarrow f'(x) = (x-r) q'(x) + q(x)$ [HW]

$\Rightarrow f'(r) = q(r)$ (By L'Hopital's rule, $q(x)$ is well-defined at $x=r$ as long as $f'(x)$ is well-defined)

Claim: r is a simple root $\Leftrightarrow f'(r) \neq 0$

$\Leftrightarrow$ (i.e. $q(r) \neq 0$)

Claim: Suppose $f'(r) = q(r) = 0$ and $q(x)$ is differentiable.

We have r is not a simple root of $q(x) \Leftrightarrow q'(r) \neq 0$

Since $f''(x) = (x-r)\,g''(x) + 2g'(x)$

$\Rightarrow f''(r) = 2g'(r)$

$\Rightarrow f''(r) = 0 \iff r$ is a multiple root of g

Thm Let r be a root of multiplicity m if

$$f(r) = f'(r) = \cdots = f^{(m-1)}(r) = 0 \quad \text{and} \quad f^{(m)}(r) \neq 0$$

Another view

$$f(x) = \underbrace{f(r)}_{\substack{r \text{ is simple root} \\ \iff f'(r) \neq 0}} + \underbrace{f'(r)}_{\substack{r \text{ is a double root} \\ \Rightarrow \frac{f''(r)}{2} \neq 0}} (x-r) + \frac{f''(r)}{2}(x-r)^2 + \cdots + \frac{f^{(n)}(r)}{n!}(x-r)^n + \cdots$$

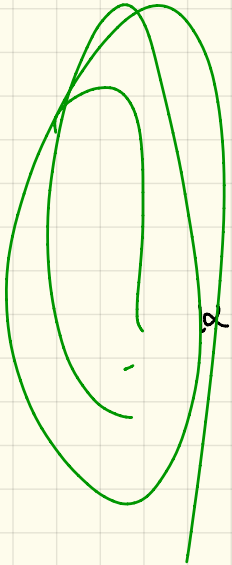
Recall $\kappa_r = \dfrac{1}{|f'(r)|}$

If r_1 is simple ($m=1$): Ill-conditioned if $f'(r_1)$ small ($= g(r_1)$)

$\Leftrightarrow g(r_1) \approx 0$

$\Leftrightarrow$ another root r_2 is close to r_1

$r_2 \approx r_1$



$\kappa_{r_2} = \dfrac{1}{f'(r_2)}$

$\because g(x) = \dfrac{f(x)}{x - r_1} \Rightarrow$

$g(r_2) = \dfrac{f(r_2)}{r_2 - r_1}$

$f'(r_2) = g(r_2) = \dfrac{f(r_2)}{r_2 - r_1}$

$r_2 - r_1 = g(r_2) \cdot f(r_2)$

§ 4.2 Fixed point iteration

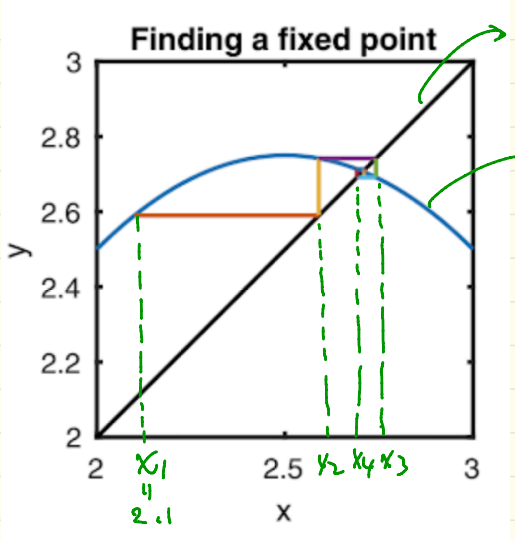
Idea: Replace $f(x)=0$ with $g(x)=x$ (fixed-point problem)

let $g(x) = x - f(x)$ and r be the fixed point of $g(x)$.

(i.e. $g(r)=r$). Then $g(r) = r - f(r) = r$ or $f(r)=0$.

Example 4.2.1

$f(x) = x^2 - 4x + 3.5$



MATLAB

```
>> f = @(x) x.^2 - 4*x + 3.5
>> r = roots([1 -4 3.5])
```

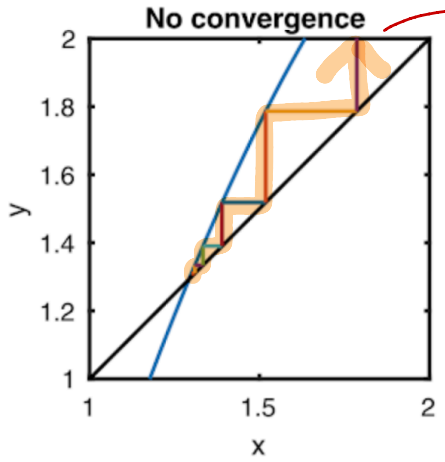
r = 2.7071, 1.2929

$x_1 = 1.3$

x_k divergence!

Now let's try to find the other fixed point in the same way. We'll use 1.3 as a starting approximation.

```
cla
fplot(g,[1 2])
hold on, plot([1 2],[1 2],'k'), ylim([1 2])
x = 1.3; y = g(x);
for k = 1:5
    plot([x y],[y y],'-'),   x = y;     % y --> new x
    y = g(x);  plot([x x],[x y],'-')  % g(x) --> new y
end
```



divergent!

We started near the true fixed point, which is at about 1.293. But this time, the iteration is pushing us *away* from the correct answer.

Series Analysis

Fixed point iteration

- Let $\varepsilon_k = x_k - r$ and $x_{k+1} = g(x_k)$

Since $\varepsilon_{k+1} = x_{k+1} - r$

$\Rightarrow x_{k+1} = \varepsilon_{k+1} + r = g(x_k)$

$= g(\varepsilon_k + r) = g(r) + g'(r)\varepsilon_k + \frac{1}{2}g''(r)\varepsilon_k^2 + \cdots$

$\Rightarrow \varepsilon_{k+1} = g'(r)\varepsilon_k + O(\varepsilon_k^2)$ (small if ε_k is small)

$\Rightarrow \varepsilon_{k+1} \approx g'(r)\cdot\varepsilon_k$ $\quad C$

$\Rightarrow \varepsilon_{k+1} \approx g'(r)^{k+1}\cdot(\varepsilon_k\cdot\varepsilon_{k-1}\cdots\varepsilon_0)$

$\Rightarrow$ (1) $|g'(r)| < 1$, $\varepsilon_{k+1} \searrow 0$

(2) $|g'(r)| > 1$, $\varepsilon_{k+1} \nearrow \infty$

In Example 4.2.1

$g(x) = -x^2 + 5x - 3.5$

$g'(x) = -2x + 5$

$|g'(\sim 2.71)| \approx |-0.42| < 1$

$|g'(\sim 1.29)| \approx |2.42| > 1$

Linear Convergence (Rate of convergence)

$$|\varepsilon_{k+1}| \approx |\varepsilon_k| \cdot \sigma, \quad 0 < \sigma < 1$$

$$|\varepsilon_k| = |x_k - r| \approx C\,|g'(r)|^k = C\,\sigma^k$$

$$\Rightarrow \log|\varepsilon_k| \approx \log|C\sigma^k| = \underbrace{\log|C|}_{\beta} + k\underbrace{\log\sigma}_{\alpha}$$

$$= \alpha k + \beta$$

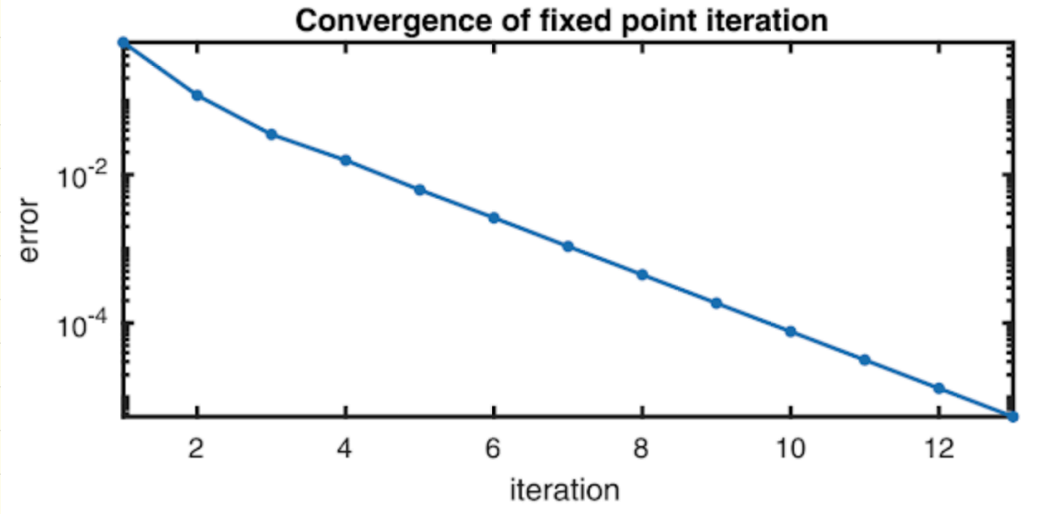
Linear convergence

$$\lim_{k\to\infty} \frac{|x_{k+1} - r|}{|x_k - r|} = \sigma \quad \text{where} \quad 0 < \sigma < 1$$

Practically speaking, linear convergence is identified by two different observations about the errors ϵ_k:

1. The errors lie on a straight line on a log–linear graph, as implied by $\log|\epsilon_k| \approx \alpha k + \beta$.
2. The error is reduced by a constant factor in each step, as implied by $|\epsilon_k| \approx C\sigma^k$.

Both statements are approximate and only apply for sufficiently large values of k, so a certain amount of judgment has to be applied.



The first value is the slope, which must be negative. We can exponentiate it to get the convergence constant σ.

```
sigma = exp(p(1))
```

```
sigma =
    0.4145
```

The numerical values of the error should decrease by a factor of σ at each iteration. We can check this easily with an elementwise division.

```
err(9:12) ./ err(8:11)
```

```
ans =
    0.4138    0.4144    0.4141    0.4142
```

The methods for finding σ agree well.

Contraction Maps

Definition 4.2.1. A function g is said to satisfy a **Lipschitz condition** with constant L on the interval $S \subset \mathbb{R}$ if

$$|g(s)-g(t)| \leq L|s-t| \quad \text{for all } s,t \in S. \tag{4.2.5}$$

It can be shown that a function satisfying (4.2.5) is continuous in S. If $L < 1$, we call g a **contraction map**, because distances between points decrease after an application of g.

From the Fundamental Theorem of Calculus, which asserts $g(s)-g(t)=\int_s^t g'(x)\,dx$, it's not difficult to conclude that an upper bound of $|g'(x)| \leq L$ for all x results in (4.2.5). But the weaker Lipschitz condition alone is enough to guarantee the success of fixed point iteration.

Theorem 4.2.1: Contraction Mapping Theorem

Suppose that g satisfies (4.2.5) with $L < 1$ on an interval S. Then S contains exactly one fixed point r of g. If $x_1, x_2, \ldots$ are generated by the fixed point iteration (4.2.1), and $x_1, x_2, \ldots$ all lie in S, then $|x_k - r| \leq L^{k-1}|x_1 - r|$ for all $k > 1$.

Partial proof. First we show there is at most one fixed point in S. Suppose $f(r) = r$ and $f(s) = s$ in S. Then by (4.2.5), $|r - s| = |g(r) - g(s)| \leq L|r - s|$, which for $L < 1$ is possible only if $|r - s| = 0$, so $r = s$.

Now suppose that for some $r \in S$, $g(r) = r$. By the definition of the fixed point iteration and the Lipschitz condition,

$$|x_{k+1} - r| = |g(x_k) - g(r)| \leq L|x_k - r|,$$

which shows that $x_k \to r$ as $k \to \infty$. To show that r must exist and complete the proof, one needs to apply the Cauchy theory of convergence of a sequence, which is beyond the scope of this book. □

§ 4.3 Newton's Method in One Variable

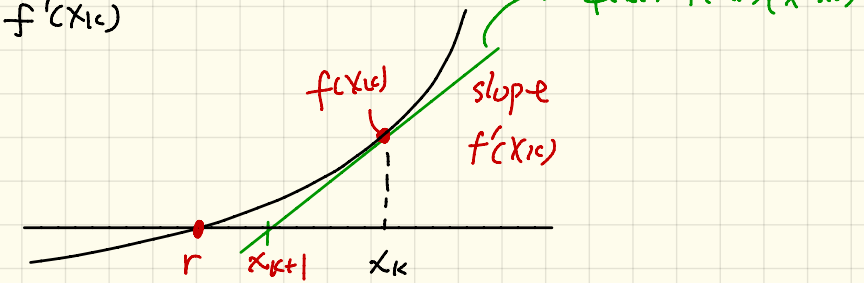
Idea: Replace $f(x)$ with a linear model

$$f(x) = f(x_k) + f'(x_k)(x - x_k) + O(\xi^2)$$

$$\Rightarrow f(x) \approx f(x_k) + f'(x_k)(x - x_k)$$

$$\Rightarrow \text{let } f(x_k) + f'(x_k)(x - x_k) = 0$$

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



Example 4.3.1

Quadratic Convergence

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

$$\Rightarrow \varepsilon_{k+1} = \varepsilon_k - \frac{f(r-\varepsilon_k)}{f'(r-\varepsilon_k)}$$

$$\epsilon_{k+1} = \epsilon_k + \frac{f(r) - \epsilon_k f'(r) + \frac{1}{2}\epsilon_k^2 f''(r) + O(\epsilon_k^3)}{f'(r) - \epsilon_k f''(r) + O(\epsilon_k^2)}.$$

We use the fact that $f(r) = 0$ and additionally assume now that $f'(r) \neq 0$. Then

$$\epsilon_{k+1} = \epsilon_k - \epsilon_k \left[1 - \frac{1}{2}\frac{f''(r)}{f'(r)}\epsilon_k + O(\epsilon_k^2)\right]\left[1 - \frac{f''(r)}{f'(r)}\epsilon_k + O(\epsilon_k^2)\right]^{-1}.$$

The series in the denominator is of the form $(1+z)^{-1}$. Provided $|z| < 1$, this is the limit of the geometric series $1 - z + z^2 - z^3 + \cdots$. Keeping only the lowest-order terms, we derive

$$\begin{aligned} \epsilon_{k+1} &= \epsilon_k - \epsilon_k \left[1 - \frac{1}{2}\frac{f''(r)}{f'(r)}\epsilon_k + O(\epsilon_k^2)\right]\left[1 + \frac{f''(r)}{f'(r)}\epsilon_k + O(\epsilon_k^2)\right] \\ &= -\frac{1}{2}\frac{f''(r)}{f'(r)}\epsilon_k^2 + O(\epsilon_k^3). \end{aligned} \tag{4.3.4}$$

Equation (4.3.4) suggests that, *eventually, each iteration of Newton's method roughly squares the error*. This behavior is called **quadratic convergence**. The formal definition of quadratic convergence is that there exists a number $\alpha > 0$ such that

$$\lim_{k\to\infty} \frac{|x_{k+1} - r|}{|x_k - r|^2} = \alpha. \tag{4.3.5}$$

Recall that linear convergence is identifiable by trending toward a straight line on a log–linear plot of the error. When the convergence is quadratic, no such straight line exists—the convergence keeps getting steeper. Alternatively, note that (neglecting high-order terms)

$$\log(|\epsilon_{k+1}|) \approx 2\log(|\epsilon_k|) + \text{constant},$$

which is equivalent to saying that the number of accurate digits approximately doubles at each iteration, once the errors become small enough.

Example 4.3.2

Example 4.3.3

§4.4 Interpolation - based

- Secant method
- Inverse interpolation
- Bracketing

Secant Method

- 접선 ⇒ 할선

Example 4.4.1 demonstrates the **secant method**. *In the secant method, one finds the root of the linear approximation through the two most recent root estimates.* That is, given previous approximations $x_1, \ldots, x_k$, define the linear model function as the line through $\big(x_{k-1}, f(x_{k-1})\big)$ and $\big(x_k, f(x_k)\big)$:

$$q(x) = f(x_k) + \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x - x_k). \tag{4.4.1}$$

Solving $q(x_{k+1}) = 0$ for x_{k+1} gives the formula

$$x_{k+1} = x_k - \frac{f(x_k)(x_k - x_{k-1})}{f(x_k) - f(x_{k-1})}, \quad n = 1, 2, \ldots. \tag{4.4.2}$$

Example 4.4.1

Function 4.4.1

Superlinear Convergence

$$\epsilon_{k+1} \approx -\frac{1}{2}\frac{f''(r)}{f'(r)}\epsilon_k\epsilon_{k-1}. \tag{4.4.3}$$

If we make an educated guess that

$$\epsilon_{k+1} = c(\epsilon_k)^\alpha, \qquad \epsilon_k = c(\epsilon_{k-1})^\alpha, \qquad \alpha > 0,$$

then (4.4.3) becomes

$$\left[\epsilon_{k-1}^{\alpha}\right]^\alpha \approx C\epsilon_{k-1}^{\alpha+1} \tag{4.4.4}$$

for an unknown constant C. Treating the implied approximation as an equality, this becomes solvable if and only if the exponents match, i.e., $\alpha^2 = \alpha+1$. The only positive root of this equation is the golden ratio,

$$\alpha = \frac{1+\sqrt{5}}{2} \approx 1.618.$$

Hence the errors in the secant method converge like $\epsilon_{k+1} = c(\epsilon_k)^\alpha$ for $1 < \alpha < 2$, a situation called **superlinear convergence**.

Example 4.4.2

P.149 Bottom

P.149 Top

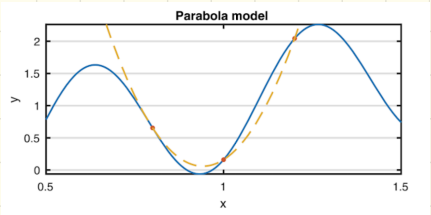
Inverse Interpolation

Idea: Newton's method: 1 point
Secant method: 2 points
higher-order: > 2 points ?

0 1 2 roots

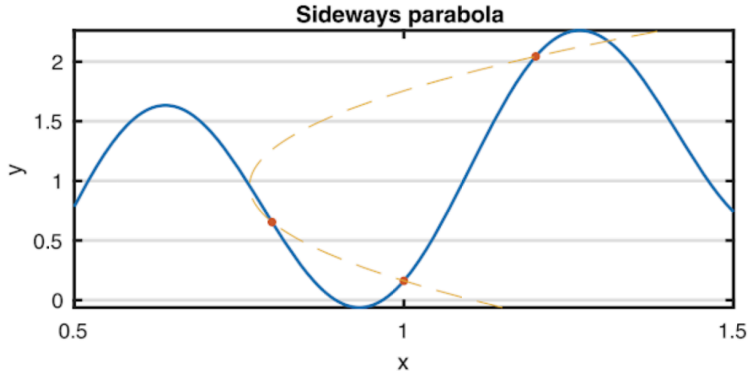
⇒ ∪ ∪ ∪

⇒ Inverse interpolation



To do inverse interpolation, we swap the roles of x and y in the interpolation.

```
c = polyfit(y,x,2);     % coefficients of interpolating
    polynomial
q = @(y) polyval(c,y);
fplot(q,@(y) y,ylim,'--')     % plot x=q(y), y=y
```



We repeat the process a few more times.

```
for k = 4:8
    c = polyfit(y(k-2:k),x(k-2:k),2);
    x(k+1) = polyval(c,0);
    y(k+1) = f(x(k+1));
end
```

Check the convergence rate on $p_1(3)$

Bracketing

- Idea: Intermediate Value Theorem

  If $f(x)$ is continuous on $[a, b]$ and $f(a)f(b) < 0$, $f(x)$ have <u>at least</u> one root in (a, b)

- Convergence: $\left|\frac{b_k - a_k}{2}\right|^k$. a guarantee method

MATLAB

- To blend fast-converging methods w/ the guarantee provided by a bracket
- For example, bracketing ⇒ sub-intervals
  ⇒ inverse quadratic estimates
  ⇒ secant methods

§ 4.5 Newton for Nonlinear Systems

Idea : Replace the nonlinear system on $\mathbb{R}^n$ by a linear model

$f: \mathbb{R}^n \to \mathbb{R}^n$

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

$$
\boldsymbol{f}(\boldsymbol{x}+\boldsymbol{h}) = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{J}(\boldsymbol{x})\boldsymbol{h} + O(\|\boldsymbol{h}\|^2), \tag{4.5.1}
$$

where $\boldsymbol{J}$ is called the **Jacobian matrix** of $\boldsymbol{f}$ and is defined by

$$
\boldsymbol{J}(\boldsymbol{x}) = \begin{bmatrix}
\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\
\frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\
\vdots & \vdots & & \vdots \\
\frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n}
\end{bmatrix} = \left[\frac{\partial f_i}{\partial x_j}\right]_{i,j=1,\ldots,n}. \tag{4.5.2}
$$

- Notes: If $f(x)$ is linear (e.g. $f(x) = Ax = b$), then the Jacobian $J(x)$ is a constant (e.g. A) and the higher-order term $O(\|h\|^2)$ disappear

The Multidimensional Newton Iteration

With a method in hand for constructing a linear model for the vector system $\boldsymbol{f}(\boldsymbol{x})$, we can generalize Newton's method. Specifically, at a root estimate $\boldsymbol{x}_k$, we set $\boldsymbol{h} = \boldsymbol{x} - \boldsymbol{x}_k$ in (4.5.1) and get

$$\boldsymbol{f}(\boldsymbol{x}) \approx \boldsymbol{q}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{J}(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k).$$

We define the next iteration value $\boldsymbol{x}_{k+1}$ by requiring $\boldsymbol{q}(\boldsymbol{x}_{k+1}) = \boldsymbol{0}$,

$$\boldsymbol{0} = \boldsymbol{f}(\boldsymbol{x}_k) + \boldsymbol{J}(\boldsymbol{x}_k)(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k),$$

which can be rearranged into

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \left[\boldsymbol{J}(\boldsymbol{x}_k)\right]^{-1} \boldsymbol{f}(\boldsymbol{x}_k). \tag{4.5.3}$$

Note that $\boldsymbol{J}^{-1}\boldsymbol{f}$ now plays the role that f/f' had in the scalar case; in fact the two are the same in one dimension. In computational practice, however, we don't compute matrix inverses. Instead, define the *Newton step* $\boldsymbol{s}_k$ by the linear $n \times n$ system

$$\boldsymbol{J}(\boldsymbol{x}_k)\boldsymbol{s}_k = -\boldsymbol{f}(\boldsymbol{x}_k), \tag{4.5.4}$$

so that $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{s}_k$. *Computing the Newton step is equivalent to solving a linear system using the Jacobian matrix and the function value.*

An extension of our series analysis of the scalar Newton method shows that the vector version is also quadratically convergent in any vector norm, under suitable circumstances.

[LAB] Example 4.5.2

Function 4.5.1

Example 4.5.3

§4.6 Quasi-Newton Methods

Issues of Newton's methods:

- Programming nuisance and computational expen of calculating the Jacobian matrix
- divergence of many starting points

Jacobian by finite differences

No derivative, ft. evaluation only

Idea

- In 1-D, $f'(x_k) \Rightarrow \left[\frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \right]$ → Conv. to $f'(x_k)$ as $x_k \to r$

- In n-D, jth column of the Jacobian:

$$\boldsymbol{J}(\boldsymbol{x})\boldsymbol{e}_j = \begin{bmatrix} \frac{\partial f_1}{\partial x_j} \\ \frac{\partial f_2}{\partial x_j} \\ \vdots \\ \frac{\partial f_n}{\partial x_j} \end{bmatrix} \approx \frac{\boldsymbol{f}(\boldsymbol{x}+\delta \boldsymbol{e}_j) - \boldsymbol{f}(\boldsymbol{x})}{\delta}, \qquad j = 1, \ldots, n.$$

- $\delta = \sqrt{\varepsilon_{mach}}$
- Function 4.6.1

Broyden's Update

Issues of the finite-difference

- Needs n extra function evaluations $\left(\vec{f}(\vec{x}+\delta e_j),\ j=1:n\right)$
- Jacobian changes little as iteration converges

Idea

- "Cheap-and-good-enough" way to update the Jacobian

Derivation

Recall $0 \approx \underbrace{f(x_{k+1})}_{f_{k+1}} \approx \underbrace{f(x_k)}_{f_k} + \underbrace{J(x_k)}_{A_k}\underbrace{(x_{k+1}-x_k)}_{s_k\ (\text{Newton step})}$

Notations $\Rightarrow$ f_{k+1} f_k A_k s_k (Newton step)

Newton iteration: $A_k s_k = -f_k$

After get x_{k+1}, we need A_{k+1}

Idea: the "secant" method

new Jacobian $A_{k+1} = \dfrac{f_{k+1} - f_k}{x_{k+1} - x_k}$ → 向量，不好算，But

$$\Rightarrow A_{k+1}(x_{k+1} - x_k) = (f_{k+1} - f_k) \quad \text{对等！}$$

$$\Rightarrow A_{k+1} s_k = f_{k+1} - f_k \qquad (+)$$

what's A_{k+1} ?

Idea: $A_{k+1} = A_k + [\text{rank-1 matrix}]$

scalar　　u_k

$$A_{k+1} = A_k + \frac{1}{s_k^T s_k}(f_{k+1} - f_k - A_k s_k)\, s_k^T \qquad (++)$$

① $(++) * s_k$

$$A_{k+1} \cdot s_k = A_k \cdot s_k + \frac{1}{s_k^T s_k}(f_{k+1} - f_k - A_k s_k)\, s_k^T \cdot s_k$$

$$\Rightarrow A_{k+1} s_k = f_{k+1} - f_k \equiv (+)$$

② is rank-1 matrix!

$u_k \cdot s_k^T$ (outer product)

③ No extra function evaluations

Needs only f_k, f_{k+1}

- Note: With some assumptions, $x_{ic} \to r$ superlinearly
  (Ak may not converge to the Jacobian)

- Note: Use finite-difference in the beginning.
  If the Broyden update fails, reinitialize A_{ic}
  by finite-difference.

Levenberg's Method

Issues

- To choose starting points
- The Jacobian matrices (① derivative, ② finite-difference, ③ Broyden's iteratively updated) become increasingly inaccurate

Ideas

- Detailed accuracy analysis (too expensive!)
- Checking whether take x_{k+1} or find an alternative

Derivations

- Balance between two ideas

$$(A_k^T A_k + \boxed{\lambda} I)\, s_k = -A_k^T f_k \qquad s_k = x_{k+1} - x_k \qquad (*)$$

(1) $\lambda = 0$

$(*) \Rightarrow A_k^T A_k s_k = -A_k^T f_k$

$\Leftrightarrow$

$A_k s_k = -f_k$

(Regular linear model)

(2) $\lambda = \infty$

$(*) \Rightarrow \lambda s_k = -A_k^T f_k$

$\Rightarrow$ s_k is the steepest descent dir. to $\min \|f(x)\|_2^2$

To solve $f(x) = 0$

$\Leftrightarrow \min \phi(x) = \|f(x)\|_2^2 = f(x)^T f(x)$

$\nabla \phi(x) = 2\boxed{J(x)^T} f(x) \approx 2\boxed{A_k^T} f(x)$

$= -2\lambda s_k$

$\Rightarrow s_k \left(\approx -\frac{1}{2\lambda} \nabla \phi(x)\right)$

$A_k = J(x)$ or $A_k \approx J(x)$

Implementation

Function 4.6.2

Example 4.6.1

§4.7 Nonlinear Least Squares

Recall: $f(x) = f(x_k) + A_k(x - x_k) + O(\|h\|^2)$

- Jacobian
- Finite-difference
- Broyden's update

$\approx q(x)$ (linear model)

$\Rightarrow q(x) \simeq 0 \Rightarrow A_k s_k = -f_k$ ($s_k = x - x_k$ or $x = x_k + s_k$; s_k: search direction)

	linear	nonlinear
square	$Ax = b$	$f(x) = 0$
overdetermined	$\min \|Ax - b\|_2$	$\min \|f(x)\|_2$

square: we can solve $A_k s_k = -f_k$

Overdetermined:

$m > n$

$f: \mathbb{R}^n \to \mathbb{R}^m$

$$\begin{cases} f_1(x_1, \cdots, x_n) = 0 \\ \vdots \\ f_m(x_1, \cdots, x_n) = 0 \end{cases}$$

$m > n$

$$\begin{cases} \min \|q(x)\|_2^2 \simeq \|A_k s_k + f(x_k)\|_2 \\ x_{k+1} = \operatorname{argmin} \|A_k s_k + f(x_k)\|_2 \end{cases}$$

Gauss-Newton Method

- Function 4.5.1 newtonsys
- Function 4.6.2 levenberg

Nonlinear Data Fitting

We can apply the Gauss-Newton method to perform nonlinear data fitting.

$$\min_{c \in \mathbb{R}^n} \sum_{i=1}^{m} | g(t_i; c) - y_i |^2 = \min_{c \in \mathbb{R}^n} \| f(c) \|^2$$

coefficients to be determined

Linear case: $g(t; c) = c_1 g_1(t) + c_2 g_2(t) + \cdots + c_n g_n(t)$

[LAB] Example 4.7.1