F2

Pivotering

Känslighetsanalys

Beräkningsvolym

Matlab

Olinjära ekvationer: Newton-Raphson

Newton-Raphson: Analys, I

Sats: (NAM p76)

Antag, att i ett intervall I omkring ett enkelt nollställe α till f det gäller att

- f har två kontinuerliga derivator
- $\max_{x,y \in I} \left| \frac{f''(x)}{2f'(y)} \right| \leq m$
- x_0 är tillräckligt nära α

så konvergerar x_j kvadratiskt mot α:

$$\lim_{j \to \infty} \frac{x_{j+1} - \alpha}{(x_j - \alpha)^2} = \frac{f''(\alpha)}{2f'(\alpha)}$$

Sats:

Om α är p-tippel rot gäller istället linjär konvergens

$$\lim_{j \to \infty} \frac{x_{j+1} - \alpha}{x_j - \alpha} = \frac{p-1}{p}$$

Newton-Raphson: Analys, II

När ska iterationen avbrytas?

När $|x_n - \alpha| < tol$ är felet mindre än *tol*. Man kan visa:

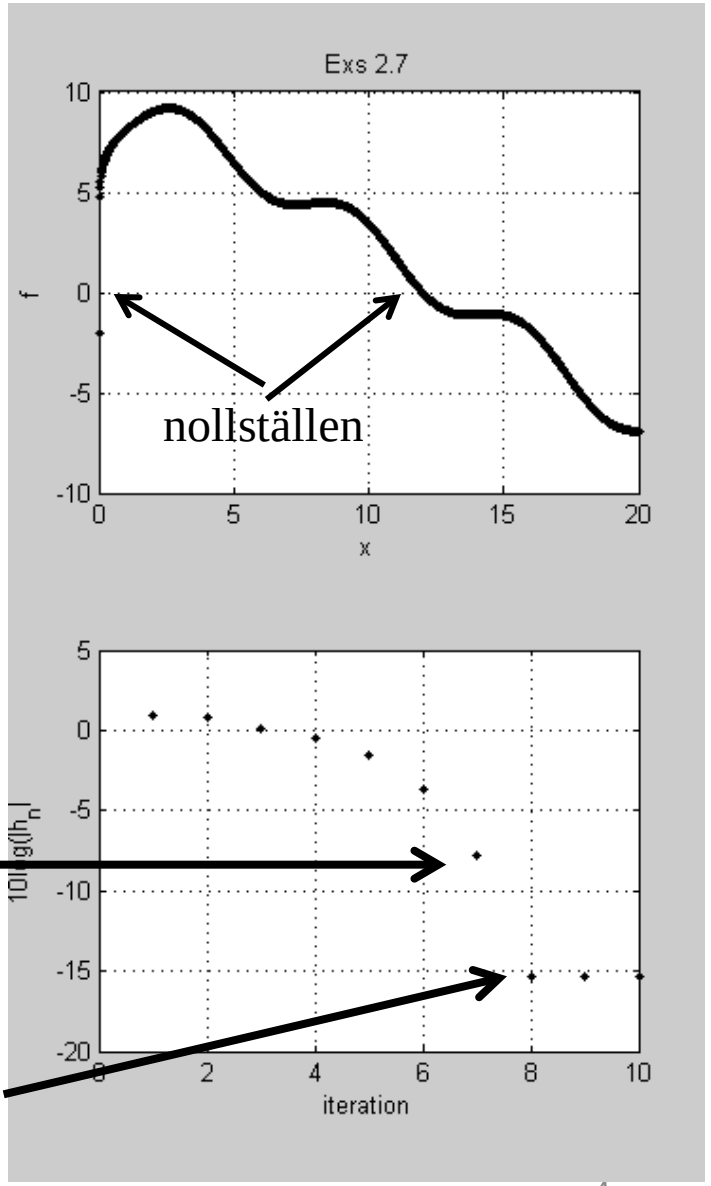
Om $m|x_n - x_{n-1}| << 1$ så är $|x_n - \alpha| < m|x_n - x_{n-1}|^2$

m kan uppskattas med $m_n = \dfrac{|x_n - x_{n-1}|}{|x_{n-1} - x_{n-2}|^2}$

om denna verkar ”konstant”

Exempel 2.7 i exempelsamlingen, II

x	h	h/hold^2
1.8722e+001	-9.7216e+000	-9.7216e+000
1.2747e+001	5.9745e+000	6.3216e-002
1.1590e+001	1.1574e+000	3.2426e-002
1.1954e+001	-3.6482e-001	-2.7233e-001
1.1984e+001	-2.9267e-002	-2.1991e-001
1.1984e+001	-2.4217e-004	-2.8272e-001
1.1984e+001	-1.6806e-008	-2.8656e-001
1.1984e+001	4.4238e-016	1.5662e+000
1.1984e+001	4.4238e-016	2.2605e+015
1.1984e+001	4.4238e-016	2.2605e+015



Kvadratisk konvergens

$h_n/(h_{n-1})^2$

… avrundningsfelen tar över

Julia-mängd, I

Newton-iterationen för att hitta rötter till polynom $p(z)$ av gradtal N, med komplexa rötter r_k, $k = 1,2, \ldots, N$ är

$$z_{n+1} = z_n - p(z_n)/\, p'(z_n), z_0 \text{ startgissning}$$

För startgissningar z_0 tillräckligt nära en rot konvergerar z_n mot roten. Men hur är det globalt? Definiera

$$R_k = \{z_0 : \lim_{n\to\infty} z_n = r_k\}$$

Hur ser R_k ut? Vi vet att R_k innehåller en liten cirkelskiva runt r_k, men annars inte just något. Man kan också undra hur *komplementet till unionen av alla* R_k ser ut.

Randen till R_k är en Julia-mängd, medan det inre kallas en Fatou-mängd.

Julia-mängd, II

Vi provar med $p(z) = z^3 + 1, r_k = e^{i(2k-1)\pi/3}, k = 1,2,3$

och startar i ett rutnät av punkter

$$z_{0,a,b}, a, b = 1,2....$$

För varje iteration kontrollerar vi vilka *z* som kommit nära rot *k* och målar dem i färg *k*.

Gaston Julia, 1893–1978

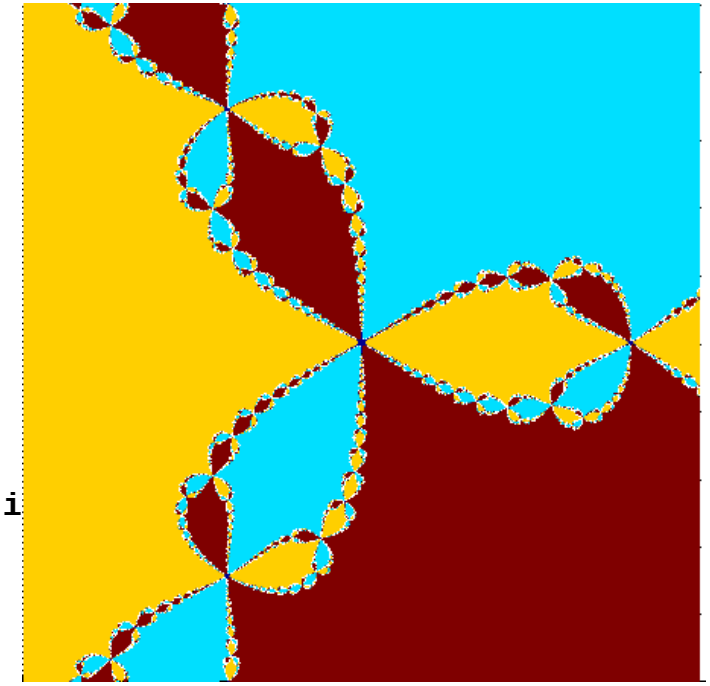
Algoritm

1. konstruera en n x n tabell z av komplexa tal
2. gör en Newtoniteration, z = z – p(z)/p’(z)

med hela tabellen

3. Bildtabellen n x n f initieras till 0
4. för k 1, 2, 3
   de f(a,b) där avståndet till rk är litet sätts till k
5. måla f (alla element är 0 1 2 eller 3)
6. tillbaka till 2

Julia-mängd, III



```
% Newtons metod på komplext polynom
%    z^3 + 1 = 0;
% rötter
%    rk = exp(i(2k-1)pi/3), k = 1,2,3
% startgissning za,b, a,b = 1,..., n
% Rk = mängd zab som ger konvergens mot rot nr k
% Gör bild av Rk!
%----------------------------------------
n = 500;
x = linspace(-1,1,n);                  % centrum i 0
y = x';                                % d:o
z = ones(n,1)*x + i*y*ones(1,n);       % kvadrat z(1:n,1:n) i
                                       % komplexa talplanet
nit = 0;                               % iterations-räknare
root = exp((2*(1:3)-1)*i*pi/3);        % rötterna
while 1                                % oändlig slinga
    clf                                % sudda
    z = z - (z.^3 + 1)./(3*z.^2);      % Newton-iteration
    nit = nit + 1;                     % räkna iterationer
    f = zeros(size(z));                % initiera bild-tabell f
    for k = 1:3                        % 3 rötter
        ind  = abs(z-root(k)) < 0.1;   % punkter nära en rot
        f(ind) = k;
    end
    surf(x,y,f); shading  interp       % z=f(x,y)
    view(2)
    axis equal
    title (['Efter ',num2str(nit),' iter.'])
    pause
end
```

Inverkan av indata-fel

$$y = f(x)$$

$$f(x+\delta) - f(x) = f'(x)\cdot\delta + O(\delta^2) \text{ då } \delta \to 0$$

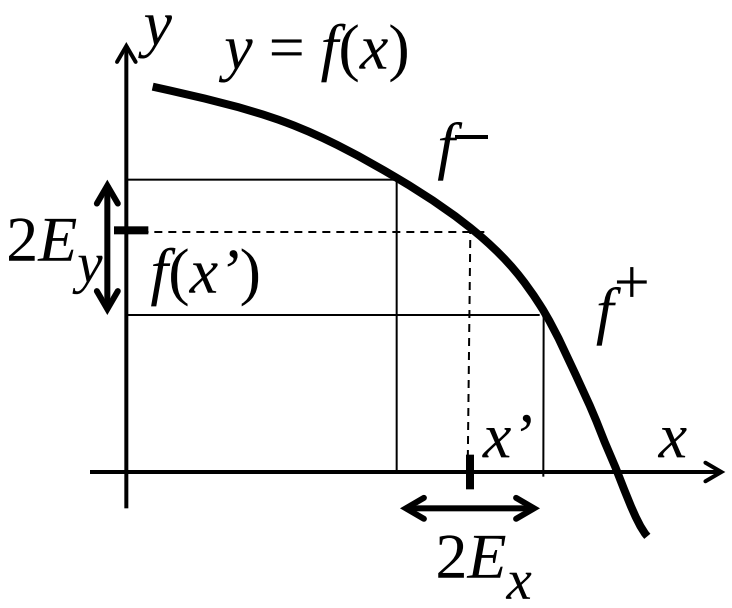
$$|\delta| \le E_x \Rightarrow |f(x+\delta) - f(x)| \le |f'(x)| \cdot E_x (+O(E_x^2))$$

$$E_y \le |f'(x)| \cdot E_x$$

Alternativ ” Instängning”,

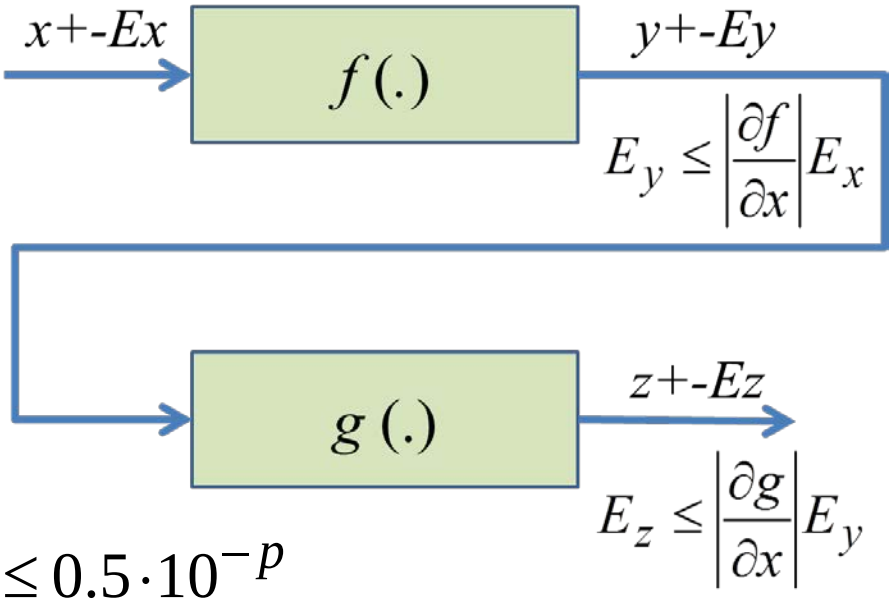
Om svårt att derivera *f:*

1. Beräkna
2. $f^+ = f(x'+E_x)$ och $f^- = f(x-E_y)$;
3. $y = 1/2(f^+ + f^-)$;
4. $E_y = |f^+ - f^-|/2$



Känslighetsanalys – felfortplantning

- Närmevärde x’ till x.
- Absolut fel x’ – x, relativt fel (x’– x)/x
- Fel***gränser***: absolut fel $x = x' \pm E_x$



- *x’* har *p* korrekta decimaler om $E_x \le 0.5 \cdot 10^{-p}$

- Vad vet vi om x’ har *p korrekta siffror*?

$$x' \cdot 10^n, 0.1 < |x'| < 1, \text{ fel } < 0.5\ 10^{-p}$$

$$\text{Relativfel} < \frac{0.5\ 10^{-p}}{0.1} = 0.5\ 10^{-p+1}$$

- Analysera EXS 8.5, 8.1

$$z = \sqrt{4318} - \sqrt{4317}$$

NAM 1.3

1. Härled trunkeringsfelet R för differensapproximationen

$$D_1(h) = \frac{f(x+h) - f(x)}{h} = f'(x) + R$$

Taylor-utveckling av f runt x ger

$$\frac{f(x+h) - f(x)}{h} = \frac{f(x) + h \cdot f'(x) + \frac{h^2}{2} \cdot f''(x) + O(h^3) - f(x)}{h} =$$

$$= f'(x) + \underbrace{\frac{h}{2} \cdot f''(x) + O(h^2)}_{R}$$

Om (dominerande delet av) felet är proportionellt mot h^p kallas formeln noggrann av ”p:te ordningen”. $p = 1$ kallas ”linjär”

NAM 1.3, forts

Tabellen i boken visar beräknade värden för $f(x) = e^x$ med $x = 1$ med precision 8 decimaler. Det visar, att felet minskar med h till h blir 10^{-4}, sedan ökar det igen för att för $h = 10^{-8}$ bli e – differensen blir precis 0. Vi ska ge en analys som visar detta.

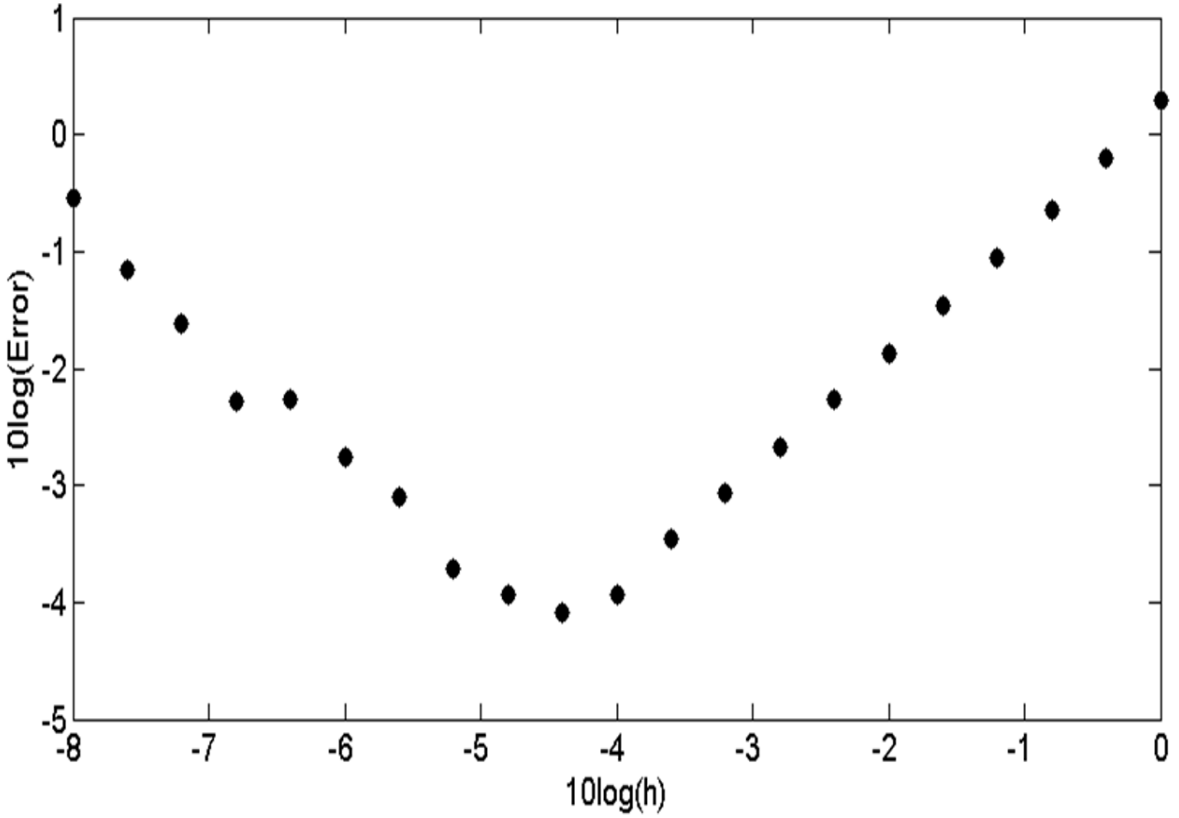
Antag, att funktionen kan beräknas med ett fel som begränsas av Ef (= 0.5 10^{-8}). Skillnaden kan få fel 2Ef. Då blir totala felet begränsat av summan av trunkeringsfel och avrundningsfel:

$$E_{tot}(h) = \frac{h}{2}\underbrace{f''(x)}_{e^1} + \frac{2E_f}{h}$$

som växer som $1/h$ då $h \to 0$ och som h då h växer. Minimum antas för

$$h = \sqrt{\frac{4E_f}{e}} \approx 1.2 \cdot 10^{-4}$$

Här är en log-log plot av resultaten (något fler punkter än i boken)
Lutningen för små *h* är -1 och för stora *h* +1. Formeln för ”optimalt” *h* ger $h = 1.2\ 10^{-4}$ – stämmer bra!



Olinjära ekvationssystem (NAM 6.8)

$\mathbf{f}(\mathbf{x}) = 0$, $\mathbf{x},\mathbf{f}$ i $\mathbf{R}^n$

Linjarisering av komponent k i $\mathbf{f}$:

$$f_k(\mathbf{x}+\mathbf{h}) = f_k(\mathbf{x}) + \frac{\partial f_k}{\partial x_1}\cdot h_1 + \frac{\partial f_k}{\partial x_2}\cdot h_2 + \ldots + \frac{\partial f_k}{\partial x_n}\cdot h_n = 0$$

$$\sum_{j=1}^{n} J_{kj} h_j = f_k(\mathbf{x}) \text{ där } J_{kj} = \left.\frac{\partial f_k}{\partial x_j}\right|_{\mathbf{x}}$$

som för $k = 1,2,\ldots,n$ sammanfattas

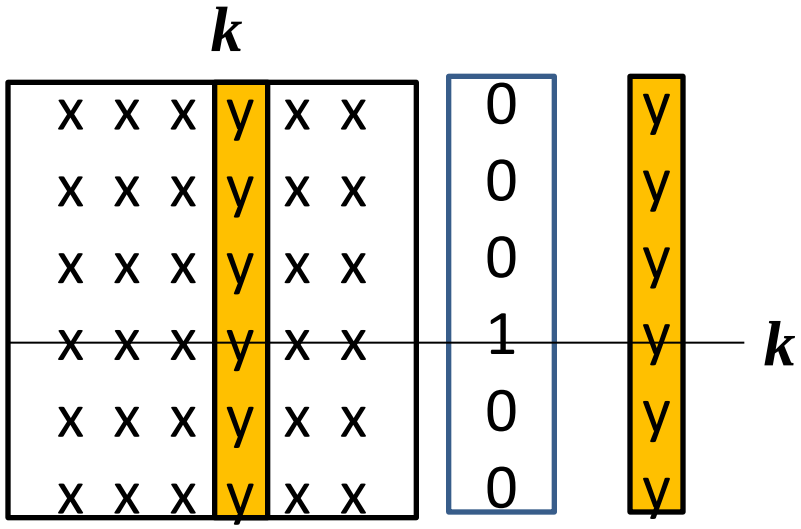
som linjära ekvationssystemet för korrektions-vektorn $\mathbf{h}$

$$\mathbf{Jh} = \mathbf{f}, \mathbf{J} = \{J_{kj}\}$$

Iterativa metoden:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{h}_n,$$
$$\mathbf{J}(\mathbf{x}_n)\mathbf{h}_n = \mathbf{f}(\mathbf{x}_n),$$
$$n = 0,1,2,\ldots \text{ tills } \|\mathbf{h}_n\| \le tol$$

Jacobian-matris NAM 6.9.3



$$\mathbf{e}_k = \mathbf{J}(:,k)$$

$$\mathbf{f}(\mathbf{x}+\delta\mathbf{e}_k)-\mathbf{f}(\mathbf{x}) \approx \mathbf{J}(:,k)\delta$$

```
function J = jacobian(F,x,steg)
n = length(x);
J = zeros(n,n); % pre-allokering
f = feval(F,x);
for k=1:length(x)
   xplus = x;
   if x(k)==0
      xplus(k) = steg;
   else
      xplus(k) = (1+steg)*x(k);
   end
   J(:,k)=(feval(F,xplus)-f)/(xplus(k)-x(k));
end
```

```
function J = Jan(xv)
x = xv(:);
n = length(xv);
J = x*ones(1,n);
```

```
function F = heath(xv)
x=xv(1);y=xv(2);z=xv(3);
f=[sin(x)+y^2+log(z)-3
   3*x+2^y-z^3
   x^2+y^2+z^3-6];
end
```

```
x0 = [1 2 3]';
delta = 1e-4;
J = jacobian('heath',x0,delta)
Jan = Jan(x0)
norm(J(:)-Jan(:))
```

Gles Jacobian: tri-diagonal

```
function J = tridiajac(F,x,steg)
n = length(x);
J = zeros(n,n); % pre-allokering
f = feval(F,x);
for k = 1:3
   j = k:3:n;
   xplus = x;
   xplus(j) = x(j)+steg;
   kol=(feval(F,xplus)-f)/steg;
   for col = j
       rows = max(col-1,1):min(col+1,n);
       J(rows,col) = kol(rows);
   end
end
```

Övning

Exs 2.7 – egen matlab Förlaga:

```
function x = newrap(F,dF,x,tol)
h  = 2*tol*abs(x);
hold = 1; iter = 0; itmax = 20;
while abs(h) > tol*abs(x)
   f     = feval(F,x);
   fprim = feval(dF,x);
   h = f/fprim;
   x = x-h;
   disp([x h h/hold h/hold^2])
   hold = h;
   iter = iter + 1;
   if iter > itmax
       break
   end
end
```

```
% testa newrap
x = newrap…
('exs2_7','exs2_7d',4,1e-12)
```

```
function f = exs2_7(x)
f = x^2-2;
```

```
function fprim = exs2_7d(x)
fprim = 2*x;
```