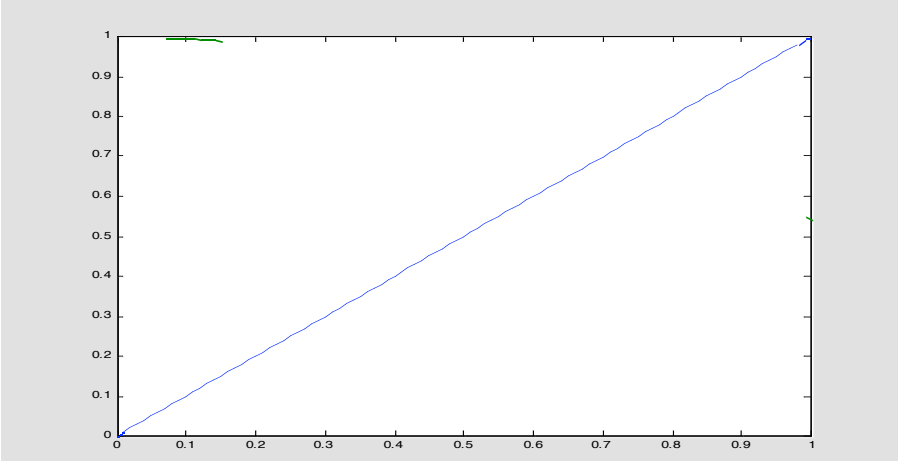
Icke-lineära ekvationer

Exempel: Roten till ekvationen $x = \cos(x)$ är lika med x -koordinaten för skärningspunkten mellan kurvorna $y = x$ och $y = \cos(x)$. Vi kan plotta kurvorna på intervallet $[0,1]$ med följande Matlabkommandon

```
x=0:0.01:1;plot(x,x,x,cos(x))
```



Vi ser ur figuren att ekvationen har en rot i intervallet $[0.7, 0.8]$. Vi skall i detta avsnitt studera olika sätt att bestämma denna rot mera noggrant.

□

Mer allmänt skall vi studera metoder för att lösa ekvationer av typen $f(x) = 0$, där f antas vara en kontinuerlig reellvärd funktion av en reell variabel. När så är lämpligt, kräver vi ytterligare regularitetsegenskaper (t.ex. att f är deriverbar) eller att den sökta roten x^* är en enkelrot (vilket innebär att $f'(x^*) \neq 0$).

I regel kan en icke-lineär ekvation $f(x) = 0$ inte lösas analytiskt dvs. lösningen kan inte anges explicit i en formel, som förutom de elementära räkneoperationerna endast innehåller rotutdragningar. Algebraiska ekvationer (dvs. ekvationer där $f(x)$ är polynom) kan lösas analytiskt då gradtalet är mindre än lika med fyra, men man kan visa, att detta i allmänhet inte är möjligt för ekvationer av högre gradtal. Inte heller lösningar till transcendenta ekvationer (som innehåller transcendenta funktioner som $e^x, \sin(x)$ etc.) kan i allmänhet anges explicit. Funktionen $f(x)$ behöver för övrigt inte ens vara känd genom ett analytiskt uttryck, utan kan t.ex. vara definierad som lösningen till en differentialekvation. I alla dessa fall måste man använda numeriska metoder för att lösa ekvationen.

Den grundidé, som de flesta numeriska metoder för ekvationslösning bygger på, är att först bestämma en grov approximation x_0 till roten x^*. Med utgångspunkt från x_0 konstrueras sedan en talföljd $\{x_n\}_{n=0}^{\infty}$, där varje tal x_n i talföljden beräknas utifrån de tidigare beräknade talen x_k, $k < n$. Under lämpliga förutsättningar konvergerar talföljden mot x^* (dvs. $\lim_{n\to\infty} x_n = x^*$) och man erhåller därför sucessivt bättre approximationer till roten genom att beräkna x_n, $n = 1, 2, 3, \ldots$, tills önskad noggrannhet uppnåtts. En metod som ger en sådan talföljd kallas en **iterationsmetod.**

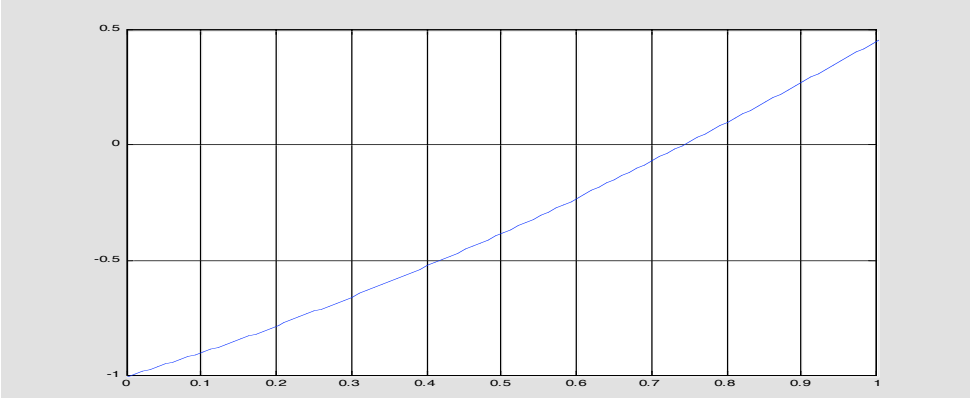
Vi skall nedan beskriva några iterationsmetoder och studera villkor för att talföljderna skall konvergera mot roten. Det är i många sammanhang också önskvärt att talföljden konvergerar snabbt. Vi definierar därför begreppet konvergenshastighet och undersöker några metoder med avseende på detta. Vi diskuterar också feluppskattning och den noggrannhet som kan uppnås då man räknar med begränsad precision.

Grovlokalisering

När man skall lösa en ekvation måste man först grovlokalisera roten för att bestämma en begynnelseapproximation x_0. Detta kan göras i två steg. Genom att plotta grafen $y = f(x)$ kan vi bestämma ett intervall i vilket roten ligger. Vi kan sedan systematiskt förbättra denna grova approximation med den s.k**. intervallhalveringsmetoden**. Denna metod bygger på att man succesivt innesluter roten i mindre och mindre intervall.

Exempel: Låt oss betrakta ekvationen i föregående exempel. Ekvationen kan skrivas på formen $f(x) = 0$, där $f(x) = x - \cos(x)$. Vi plottar grafen till $f(x)$ i intervallet [0,1] med Matlabkommandot

```
x=0:0.01:1;plot(x,x-cos(x)),grid
```



Vi ser att funktionen $f(x)$ har ett nollställe (dvs. att ekvationen $f(x)=0$ har en rot) i intervallet $[0.7,0.8]$. Detta bekräftas om vi beräknar funktionsvärdena i ändpunkterna på detta intervall. Vi får att $f(0.7)\approx -0.06<0$ och $f(0.8)\approx 0.10>0$. Eftersom $f(x)$ är kontinuerlig så följer det av satsen om mellanliggande värden att det måste finnas minst ett nollställe x^* i intervallet $[0.7,0.8]$. Genom att beräkna funktionsvärdet i mittpunkten på detta intervall dvs.i $x=0.75$ kan vi nu bestämma roten närmare. Eftersom $f(0.75)\approx 0.02>0$ så måste x^* ligga i intervallet $[0.7,0.75]$. Vi kan sedan upprepa denna procedur till önskad noggrannhet erhållits. Mittpunkten på intervallet $[0.7,0.75]$ är 0.725 och eftersom $f(0.725)\approx -0.02<0$ så måste x^* ligga i intervallet $[0.725,0.75]$ etc.

Följande Matlabkod implementerar intervallhalvering

```
function [a1,b1]=inthalv(f,a,b,tol)
a1=a;b1=b;fb=feval(f,b);fa=feval(f,a);
if fa*fb>0
   error('f(a) och f(b) måste ha olika tecken')
end
while b1-a1>tol
    disp([a1 b1])
    x=(a1+b1)/2;
    if feval(f,x)*fb>0
        b1=x;
    else
        a1=x;
    end
end
```

Om vi sparar ovanstående programrader i en m-fil med namnet **inthalv.m** så kan intervallhalveringen utföras med följande kommandorader:

```
f=inline('x-cos(x)');inthalv(f,0.7,0.8,0.001)

    0.7000    0.8000
    0.7000    0.7500
    0.7250    0.7500
    0.7375    0.7500
    0.7375    0.7438
    0.7375    0.7406
    0.7391    0.7406
ans =
    0.7391
```

□

Man inser omedelbart att denna metod alltid konvergerar, men konvergenshastigheten är låg. Efter n steg med metoden så har roten inneslutits i ett intervall av längden 2^{-n}

gånger längden av ursprungsintervallet. Då vi i vårt exempel startar med intervallet $[0.7, 0.8]$ måste vi t.ex. halvera intervallet 17 gånger för att få ett intervall som är kortare än 10^{-6} (mittpunkten i det intervallet är roten med sex korrekta decimaler).

Iterationsmetoder

På grund av den långsamma konvergensen bör man endast använda intervallhalveringsmetoden för att grovlokalisera roten. De metoder vi nu skall beskriva har snabbare konvergens, vi kan kalla dem iterationsmetoder för finjustering. De bygger på att man med hjälp av dittills beräknad information om funktionen f och om rotens approximativa läge beräknar en ny och noggrannare approximation. Allmänt kan man säga att ju mer information om funktionen man använder desto snabbare konvergens kan uppnås.

Ett naturligt sätt att approximera funktionen $f(x)$ är att dra en tangent till kurvan $y = f(x)$. Antag att vi har ett närmevärde x_0 till roten. Vi drar då tangenten till kurvan i punkten $(x_0, f(x_0)$ och låter tangentens skärningspunkt med x-axeln vara nästa approximation x_1 till roten

Tangentens ekvation är $y - f(x_0) = f'(x_0)(x - x_0)$. Dess skärningspunkt med x-axeln får vi genom att sätta $y = 0$ och lösa ut x ur denna ekvation. Vi får då att $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$. Om vi nu på samma sätt fortsätter att beräkna en ny approximation till roten med utgångspunkt från x_1 så får vi $x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$. Vi kan sedan upprepa denna process till önskad noggrannhet har uppnåtts. Detta sätt att erhålla approximativa lösningar på ekvationer kallas för Newton-Raphsons metod.

Newton-Raphsons metod

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \; n = 0, 1, 2, \ldots$$

I Newton-Raphsons metod använder vi alltså både funktionsvärdet och derivatavärdet när vi beräknar en ny approximation till roten. Vi kan därför förvänta oss att metoden ger en snabbt konvergerande talföljd.

Exempel: Kvadratroten $\sqrt{a}$ (där $a \geq 0$) kan bestämmas genom att lösa ekvationen $x^2 - a = 0$. Newton-Rahsons metod tillämpad på denna ekvation ger talföljden $x_{n+1} = \frac{1}{2}(x_n + \frac{a}{x_n})$. Med $a = 3$ och $x_0 = 2$ så får vi

$n=0:\quad x_1=\frac{1}{2}(2+\frac{3}{2})=\frac{7}{4}$

$n=1:\quad x_2=\frac{1}{2}(\frac{7}{4}+\frac{3}{7/4})=\frac{97}{56}$

$n=2:\quad x_3=\frac{1}{2}(\frac{97}{56}+\frac{3}{97/56})=\frac{18817}{10864}\approx 1.7320508$

vilket är $\sqrt{3}$ med 7 korrekta decimaler.

□

Exempel: Låt oss lösa vår illustrationsekvation $x-\cos(x)=0$ med Newton-Raphsons metod. Med $f(x)=x-\cos(x)$ så har vi $f'(x)=1+\sin(x)$.

Följande Matlabkod implementerar Newton-Raphsons metod

```
function xut=NR(f,x,tol)
h=0.0001;
fx=feval(f,x);
fprimx=(feval(f,x+h)-feval(f,x))/h;
format long
disp('      x_n              f(x_n)          f/fprim')
disp([x fx fx/fprimx])
while abs(fx/fprimx)>tol
   x=x-fx/fprimx;
   fx=feval(f,x);
   fprimx=(feval(f,x+h)-feval(f,x))/h;
   disp([x fx fx/fprimx])
end
format short
```

Om vi sparar ovanstående programrader i en m-fil med namnet **NR.m** så kan Newton-Raphsons metod utföras med följande kommandorader:

```
f=inline('x-cos(x)');NR(f,0.74,10^(-9))
```

```
      x_n                f(x_n)              f/fprim
   0.74000000000000   0.00153144127041   0.00091466198158
   0.73908533801842   0.00000034276122   0.00000020479873
   0.73908513321969   0.00000000000758   0.00000000000453
```

Vi ser att metoden i detta fall ger en snabbt konvergerande talföljd. Det är naturligt att fråga sig om metoden alltid fungerar lika bra som i detta fall. Man kan också fundera över hur stor noggrannheten är i approximationen `0.73908513321969`. Vi skall återkomma till dessa frågor lite längre fram.

□

I nästa avsnitt skall vi studera konvergensvillkor och konvergenshastighet hos iterationsmetoder. För den diskusionen är det praktiskt att skriva Newton-Raphsons metod på formen $x_{n+1} = F(x_n)$, där **iterationsfunktionen** är $F(x) = x - \frac{f(x)}{f'(x)}$. Observera att ekvationen $x = F(x)$ är en ekvivalent omskrivning av ekvationen $f(x) = 0$ (dvs. ekvationen $f(x) = 0$ kan bringas på formen $x = F(x)$ genom elementära matematiska operationer, och tvärtom). Eftersom x^* är en rot till ekvationen $f(x) = 0$ så gäller att $x^* = F(x^*)$. Man säger därför att x^* är en **fixpunkt** till funktionen $F(x)$ och iterationsmetoden $x_{n+1} = F(x_n)$ kallas för en **fixpunktiteration**.

Fixpunktiterationer kan fås genom att göra andra ekvivalenta omskrivningar än den som leder till Newton-Raphsons metod. Ekvationen i exemplen ovan kan t.ex. skrivas $x = \cos(x)$ och motsvarande iterationsmetod är $x_{n+1} = \cos(x_n)$

Följande Matlabkod implementerar fixpunktiteration

```
function xut=fixit(F,x,tol)
n=0;
format long
disp('          x_n                    x_n-F(x_n)')
while abs(x-feval(F,x))>tol
   n=n+1;
   x=feval(F,x);
   disp(n) , disp([x x-feval(F,x)])
end
format short
```

Om vi sparar ovanstående programrader i en m-fil med namnet **fixit.m** så kan fixpunkiterationen utföras med följande kommandorader:

```
F=inline('cos(x)');fixit(F,0.74,10^(-8))
          x_n              x_n-F(x_n)
     1
   0.73846855872959  -0.00103176596281
     2
   0.73950032469240   0.00069493314590
     3
   0.73880539154650  -0.00046815010057
     4
   0.73927354164707   0.00031533573523
     5
   0.73895820591185  -0.00021242110789

     ⋮

    28
```

```
   0.73908514756884   0.00000002402249
    29
   0.73908512354635  -0.00000001618184
    30
   0.73908513972819   0.00000001090028
    31
   0.73908512882791  -0.00000000734256
```

Det ser ut som den genererade talföljden konvergerar och att skillnaden från ett steg till ett annat dvs. $x_n - F(x_n)$ reduceras med en faktor som är ungefär 0.7 i varje iteration.

En annan ekvivalent omskrivning av ekvationen $x - \cos(x) = 0$ är $x = \arccos(x)$. Iterationen $x_{n+1} = \arccos(x_n)$ divergerar emellertid, vilket framgår tydligt av nedanstående beräkningar

```
F=inline('acos(x)');fixit(F,0.74,0.001)
         x_n              x_n-F(x_n)
     1
   0.73772596845325  -0.00337466397108
     2
   0.74110063242432   0.00501250846162
     3
   0.73608812396270  -0.00743537924048
     4
   0.74352350320319   0.01105132033738
     5
   0.73247218286581  -0.01637794430875
     6
   0.74885012717456   0.02437914423404
     7
   0.72447098294051  -0.03605872350073
     8
   0.76052970644124   0.05386191295854
     9
   0.70666779348270  -0.07935099972840
    10
   0.78601879321111   0.11956486367678
```

Uppenbarligen behöver vi analysera villkoren på funktionen $F(x)$ i fixpunktiterationen $x_{n+1} = F(x_n)$, under vilka den genererade talföljden konvergerar. Detta görs i nästa avsnitt.

Konvergensvillkor och konvergenshastighet

Ett grundläggande konvergensvillkor för fixpunktiterationer ges i följande sats.

Sats: Antag att $x_{n+1} = F(x_n)$, $n = 0,1,2,...$, och att $F(x)$ har en reell fixpunkt x^*. Antag vidare att det finns tal m och δ sådana att $|F'(x)| \le m < 1$, för alla $x \in J = \{x : |x - x^*| \le \delta\}$ Om $x_0 \in J$ så gäller då att

a) $x_n \in J$, $n = 1,2,3,...$
b) $\lim_{n\to\infty} x_n = x^*$
c) x^* är den enda roten till ekvationen $x = F(x)$ i intervallet J

□

Bevis: Vi börjar med att bevisa påståendet i a) med induktion. Antag att $x_{k-1} \in J$. Medlevärdessatsen ger då att $x_k - x^* = F(x_{k-1}) - F(x^*) = F'(\xi_k)(x_{k-1} - x^*)$, där ξ_k är ett tal mellan x_{k-1} och x^*. Eftersom x_{k-1} och x^* ligger i intervallet J så ligger även ξ_k i J. Av förutsättningarna följer därför att $|x_k - x^*| = |F'(\xi_k)(x_{k-1} - x^*)| \le m|x_{k-1} - x^*| \le m\delta < \delta$, så vi har även $x_k \in J$. Vi har därmed visat att $x_{k-1} \in J \Rightarrow x_k \in J$, för alla $k \ge 1$. Eftersom $x_0 \in J$ så följer det med induktion att $x_n \in J$, för alla $n \ge 0$.

Låt oss nu bevisa påståendet i b). Ur beviset av a) ovan så fick vi att $|x_k - x^*| \le m|x_{k-1} - x^*|$, för alla $k \ge 1$. Upprepad användning av denna olikhet, med först $k = n$ och sedan successivt med $k = n-1$, $k = n-2$, $k = n-3$ etc., får vi att

$$|x_n - x^*| \le m\underbrace{|x_{n-1} - x^*|}_{\le m|x_{n-2} - x^*|} \le m^2\underbrace{|x_{n-2} - x^*|}_{m|x_{n-3} - x^*|} \le m^3|x_{n-3} - x^*| \le \ldots \le m^n|x_0 - x^*|$$

Eftersom $m < 1$ så följer det att $|x_n - x^*| \to 0$ då $n \to \infty$, och vi har därmed visat b).

Entydigheten i c) visas med ett motsägelsebevis. Antag att x^{**} är en annan fixpunkt till $F(x)$ i intervallet J dvs. att $x^{**} \in J$, $x^{**} \ne x^*$ och $x^{**} = F(x^{**})$. Medelvärdessatsen och antagandena ger då att

$$|x^* - x^{**}| = |F(x^*) - F(x^{**})| = |F(\xi)(x^* - x^{**})| \le m|x^* - x^{**}| < |x^* - x^{**}|$$

vilket är en motsägelse.

□

Exempel: Låt oss undersöka de två fixpunktiterationerna $x_{n+1} = \cos(x_n)$ och $x_{n+1} = \arccos(x_n)$, för ekvationen $x = \cos(x)$ (se tidigare exempel). I det första fallet har vi $F(x) = \cos(x)$. Nära roten $x^* \approx 0.739085$ så är $|F'(x)| = |-\sin(x)| \approx \sin(0.739085) \approx$ $\approx 0.6736 < 1$. Det följer därför av satsen ovan att iterationen konvergerar, vilket vi även konstaterade experimentellt i föregående avsnitt. I det andra fallet har vi $F(x) = \arccos(x)$. Nära roten $x^* \approx 0.739085$ så är $|F'(x)| = \left|\frac{-1}{\sqrt{1-x^2}}\right| \approx \frac{1}{\sqrt{1-0.739085^2}} \approx$

$\approx 1.48 > 1$, vilket innebär att det finns ett intervall $I = \{x : |x - x^*| < \varepsilon\}$ kring roten x^* för vilket $|F'(x)| > 1$. Talföljden kommer därmed inte att konvergera mot roten x^*. Detta kan inses med ett motsägelseargument: Om det vore så att $\lim_{n\to\infty} x_n = x^*$ så skulle det finnas något tal n sådant att $|x_k - x^*| < |x_n - x^*| < \varepsilon$ för alla $k \geq n$. Detta stämmer emellertid inte ty $|x_{n+1} - x^*| = |F(x_n) - F(x^*)| = |F'(\xi_n)||x_n - x^*| > |x_n - x^*|$.

□

Som exemplet ovan visar så är villkoret $|F'(x)| \leq m < 1$, i en omgivning av roten x^*, inte bara ett tillräckligt villkor, utan även ett nödvändigt villkor, för att en fixpunktiteration skall konvergera mot roten. Ur beviset av satsen ser vi också att ju mindre talet m är, desto snabbare konvergerar iterationen. För den första metoden i exemplet ovan var $m \approx 0.7$. Denna metod bör därför ge något långsammare konvergens än intervallhalveringsmetoden, vilket också kan utläsas ur resultaten i föregående avsnitt.

Låt oss nu analysera konvergenshastigheten hos Newton-Raphsons metod. I den metoden är $F(x) = x - \frac{f(x)}{f'(x)}$ och därmed $F'(x) = \frac{f(x)f''(x)}{(f'(x))^2}$. Om man känner till ett litet intervall där roten ligger är det ofta lätt att verifiera att konvergensvillkoret för Newton-Raphsons metod är uppfyllt:

$$|F'(x)| = \left|\frac{f(x)f''(x)}{(f'(x))^2}\right| \leq m < 1$$

Eftersom $f(x^*) = 0$ så är även $F'(x^*) = 0$, så Newton-Raphsons metod bör konvergera mycket snabbt när man väl kommit nära roten. Låt oss undersöka konvergensen hos metoden i mer detalj. Som tidigare har vi $x_{n+1} - x^* = F(x_n) - F(x^*)$. Istället för att använda medelvärdessatsen så kan vi Taylorutveckla $F(x_n)$ kring x^*:

$$F(x_n) = F(x^*) + F'(x^*)(x_n - x^*) + \frac{F''(\eta)}{2}(x_n - x^*)^2$$

där η ligger mellan x_n och x^*. Eftersom $F'(x^*) = 0$ så följer att

$$|x_{n+1} - x^*| = |F(x_n) - F(x^*)| = \left|\frac{F''(\eta)}{2}(x_n - x^*)^2\right| \leq C|x_n - x^*|^2$$, för något C.

Detta visar att felet i Newton-Raphsons metod i princip kvadreras i varje steg. Speciellt konvergerar Newton-Raphsons metod snabbare än fixpunkiterationer i allmänhet. För att jämföra konvergenshastigheten hos olika iterationsmetoder inför vi följande definition.

Definition: Låt $\{x_n\}_{n=0}^{\infty}$ vara en talföljd som konvergerar mot ett tal x^*, och sätt $\varepsilon_n = x_n - x^*$. Talföljdens **konvergensordning** sägs vara p, om $p \geq 1$ är det största tal som är sådant att $\lim_{n\to\infty} \frac{|\varepsilon_{n+1}|}{|\varepsilon_n|^p} = C < \infty$. Talet C kallas för den **asymptotiska**

felkonstanten. Om $p=1$ sägs konvergensen vara lineär och om $p=2$ sägs konvergensen vara kvadratisk.

□

Ofta säger vi att en iterationsmetod har konvergensordningen p om den genererar konvergenta talföljder med denna konvergensordning. Fixpunktiterationer i allmänhet har lineär konvergens med asymptotisk felkonstant $F'(x^*)$. Newton-Raphsons metod har kvadratisk konvergens och dess asymptotiska felkonstant är $\frac{F''(x^*)}{2} = \frac{f''(x^*)}{2f'(x^*)}$.

Exempel: Antag att en iterationsmetod genererat en talföljd där $x_{n+1} = \frac{1}{3}x_n$, $n = 1,2,3,...$ med $x_1 = 10$. Vi ser direkt att $x_n \to 0$ så $x^* = 0$. Låt oss nu undersöka konvergenshastigheten hos denna talföljd. Vi har

$$\frac{|\varepsilon_{n+1}|}{|\varepsilon_n|^p} = \frac{|x_{n+1}-0|}{|x_n-0|^p} = \frac{|x_n/3|}{|x_n|^p} = \frac{1}{3}|x_n|^{1-p} \to \begin{cases} \infty & om\ p>1 \\ 1/3 & om\ p=1 \end{cases}, \text{ då } n \to \infty$$

så konvergensen är lineär med asymptotisk felkonstant $\frac{1}{3}$.

□

Det skall avslutningsvis påpekas att i praktiken bryr man sig inte om att kontrollera konvergensvillkoret $|F'(x)| \le m < 1$ i förväg, eftersom man upptäcker eventuell divergens vid själva iterationen.

Feluppskattning

När man beräknar approximationer till en rot x^* med hjälp av någon iterationsmetod, görs avrundningsfel. Det framgår dock av fixpunktsatsen ovan att så länge som man inte gör stora fel (så att man hamnar utanför intervallet J) så har dessa fel ingen inverkan alls på den slutliga noggrannheten. Man brukar säga att iterationsmetoder är självkorrigerande.

Konvergensordningen hos en iterationsmetod ger en uppfattning om felets uppförande asymptotiskt, dvs. efter ett stort antal steg. I praktiken måste man dock avbryta iterationen efter ett ändligt antal steg, och varken konvergensordningen eller den asymptotiska felkonstanten ger oss upplysning om hur stort felet är.

Ett sätt att uppskatta felet i en approximation $\tilde{x}$ till en rot x^* erhåller vi ur medelvärdessatsen. Medelvärdessatsen ger nämligen att

$$f(\tilde{x}) = f(\tilde{x}) - f(x^*) = f'(\xi)(\tilde{x} - x^*)$$

där ξ ligger mellan $\tilde{x}$ och x^*. Om x^* är en enkelrot så är $f'(x^*) \neq 0$, och om $\tilde{x}$ ligger nära x^* så är även $f'(\xi) \neq 0$ (f' förutsättes kontinuerlig). Vi får således att

$$|\tilde{x}-x^*|=\frac{|f(\tilde{x})|}{|f'(\xi)|}$$

Om nu $|f'(x)|\geq M$ för alla x nära x^* så får vi följande feluppskattning

$$\boxed{|\tilde{x}-x^*|\leq\frac{|f(\tilde{x})|}{M}}$$

Denna feluppskattning gäller oberoende av vilken metod som använts för att beräkna närmevärdet och kallas därför för **metodoberoende feluppskattning**.

Exempel: Vi har löst ekvationen $f(x)=0$, där $f(x)=x-\cos(x)$, med olika metoder i tidigare avsnitt och kommit fram till att roten bör vara ungefär 0.73908516. Vi skall nu undersöka hur noggrann denna approximation är. Med $\tilde{x}=0.73908516$ så har vi $f(\tilde{x})\approx 4.5\cdot 10^{-8}$ och $f'(\tilde{x})=1+\sin(\tilde{x})\approx 1.67$. Nära roten är därför $|f'(x)|\geq 1.6$ så den metodoberoende feluppskattningen ger att

$$|\tilde{x}-x^*|\leq\frac{4.5\cdot 10^{-8}}{1.6}\leq 2.9\cdot 10^{-8}\leq 0.5\cdot 10^{-7}$$

dvs. $\tilde{x}=0.73908516$ är en approximation till roten x^* med 7 korrekta decimaler.

□

Om man använt fixpunktiteration för att erhålla ett närmevärde till en rot så kan man uppskatta felet på ytterligare ett sätt. Ur beviset av fixpunktsatsen framgår nämligen att

$$\boxed{|x_n-x^*|\leq m^n\,|x_0-x^*|}$$

Exempel: I ett tidigare avsnitt löste vi t.ex. ekvationen $x-\cos(x)=0$ genom att utföra fixpunkiterationen $x_{n+1}=F(x_n)$, där $F(x)=\cos(x)$ och $x_0=0.74$. Vi vet från intervallhalveringsmetoden att $x^*\in[0.725,0.75]$. Eftersom $|F'(x)|=|\sin(x)|\leq$ $\leq\sin(0.76)\leq 0.7$, för alla $x\in[0.715,0.76]$, så följer det att $|F'(x)|\leq 0.7$, för alla $x\in J=\{x:|x-x^*|<0.01\}$. Startapproximationen $x_0=0.74$ ligger i J så vi får uppskattningen $|x_n-x^*|\leq 0.7^n\,|0.74-x^*|\leq 0.7^n\cdot 0.1$. Efter 31 iterationer erhöll vi tidigare närmevärdet $\tilde{x}=0.73908512$. Vi ser nu att felet i denna approximation är $\leq 0.7^{31}\cdot 0.1\leq 1.58\cdot 10^{-5}\leq 0.5\cdot 10^{-4}$ så $\tilde{x}=0.73908512$ är ett närmevärde till roten x^* med 4 korrekta decimaler.

□

Laborationen med Matlab(Java)
Använd
1)intervallhalveringsmetoden
2) fixpunktiterationsmetod
3)newtonsraphsons metod
För att lösa följande ekvationer

$a)\ \ x^4 = 4 - 2x$

$b)\ \ 2\cos x = 2 - x$

$c)\ \ \sqrt{x+3} = x^2$

$d)\ \ e^{2x} = \ln|x| + 10$

Glöm inte att rita via Matlab
Kommentera dina resultat t.ex vilken metoden som är lämplig (snabbbast , hur stor är felet)