Numerisk lösning av ekvationer

1 Inledning

Vi skall lösa ekvationer, dvs. beräkna nollställen till funktioner. För att se hur många nollställen en funktion har och ungefär var de ligger behöver vi rita en graf av funktionen. Därefter beräknar vi noggrant de nollställen som vi är intresserade av, dels med egna program dels med hjälp av i MATLAB inbyggda program.

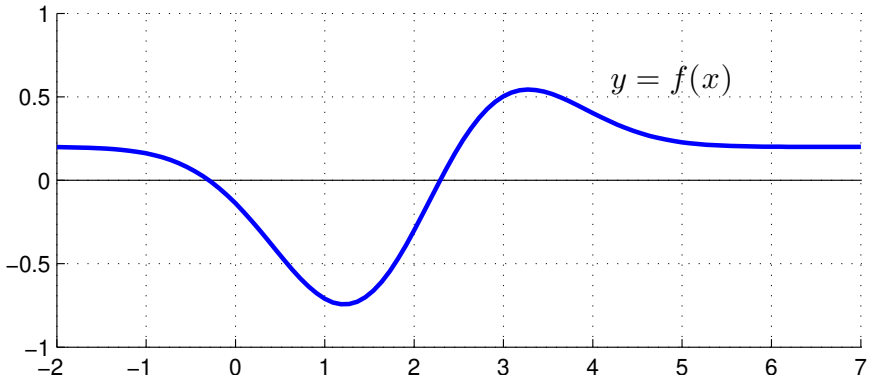
Som exempel kan vi ta ekvationen,

$$f(x) = (x - 2.5)\,\mathrm{e}^{-0.5\,(x-2)^2} + 0.2 = 0$$

Vi börjar med att rita grafen till f för att få en uppfattning om hur många nollställen vi har och ungefär var de ligger.

```
>> f=@(x)(x-2.5).*exp(-0.5*(x-2).^2)+0.2;
>> x=linspace(-2,7);
>> plot(x,f(x))
>> axis([-2 7 -1 1]), grid on
```

Vi ser lösningar till $f(x) = 0$ som de punkter där grafen skär x-axeln.



Vi ser att vi har två nollställen, ett i intervallet $[a, b] = [-1, 0]$ och ett i intervallet $[a, b] = [2, 3]$.

Givetvis kan vi grafiskt läsa av en approximation av nollställena, men det är svårt att få hög noggrannhet i denna avläsning.

Vi behöver en metod som stegvis kan förbättra en approximation tills den blir tillräckligt noggrann. Intervallhalveringsmetoden (eller bisektionsmetoden) är en sådan metod.

Metoden bygger på följande resonemang: Om en funktion f växlar tecken på ett intervall $[a, b]$ och om f är kontinuerlig så måste f bli noll någonstans i intervallet, dvs. vi har minst ett nollställe i intervallet. Delar vi intervallet i mitten, så att vi får två delintervall, så kommer vi fortfarande ha teckenväxling i något av delintervallen.

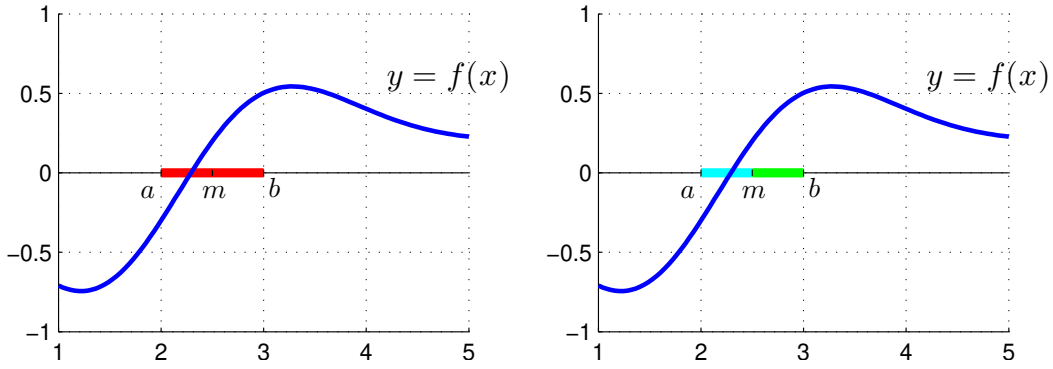
Detta delintervall kommer innehålla minst ett nollställe och vi behåller detta intervall och upprepar resonemanget för detta nya intervall.

Intervallhalveringsmetoden är pålitlig men långsam, den kräver vanligtvis väldigt många iterationer (intervallhalveringar) för att få tillräcklig noggrannhet i approximationen av lösningen. I ett senare avsnitt skall vi se på Newtons metod som bygger på att successivt approximera funktionen $f(x)$ med linjäriseringar (tangenter). Newtons metod är mycket snabbare, men kräver en tillräckligt bra första approximation av en lösning.

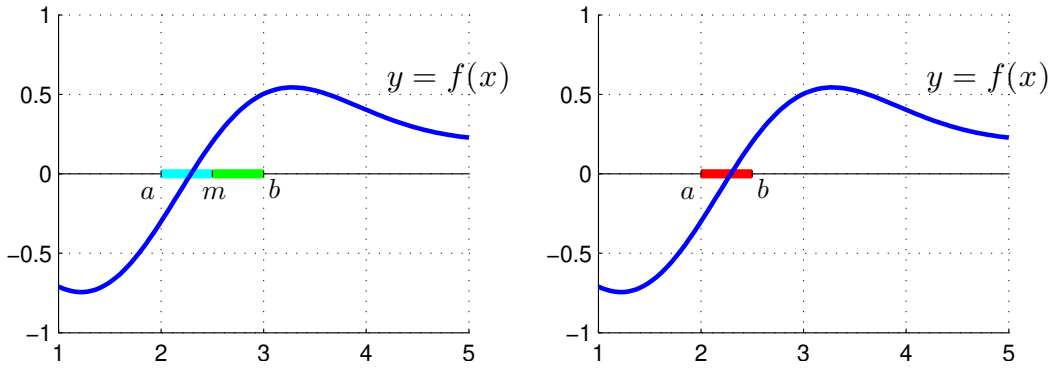
2 Intervallhalveringsmetoden

Antag att funktionen $f(x)$ är kontinuerlig. Starta med ett intervall $[a, b]$ där $f(x)$ växlar tecken, dvs. där $f(a) \cdot f(b) < 0$. Eftersom $f(x)$ är kontinuerlig så finns det minst ett nollställe i $[a, b]$. Bilda mittpunkten $m = \frac{a+b}{2}$ och dela intervallet i två lika långa delintervall $[a, m]$ och $[m, b]$.

Vi delar intervallet i två lika långa delintervall, dvs. $[a, m]$ och $[m, b]$.



Delintervall där $f(x)$ växlar tecken behåller vi, dvs. vi behåller $[a, m]$ om $f(a) \cdot f(m) < 0$, annars behåller vi $[m, b]$.



Upprepa tills vi har ett intervall som är tillräckligt smalt, dvs. att vi har stängt in ett nollställe med tillräcklig noggrannhet.

Läs gärna i Persson-Böiers kapitel 2.5.2 om intervallhalveringsmetoden.

Uppgift 1. Rita grafen till $f(x) = (x - 2.5)\,\mathrm{e}^{-0.5\,(x-2)^2} + 0.2$, dvs. återskapa bilden i inledningen. Använd en anonym funktion med ett funktionshandtag för att beskriva funktionen (precis som i det inledande exemplet).

Uppgift 2. Vi ser att vi har ett nollställe till $f(x) = 0$ i intervallet $[a, b] = [-1, 0]$.

Skriv följande i MATLAB kod:

(a). Låt $a = -1$ och $b = 0$.

(b). Undersök om f växlar tecken på $[a, b]$ genom att bilda $f(a) \cdot f(b)$.

(c). Bilda mittpunkten m.

(d). Om f växlar tecken på $[a, m]$, låt b få värdet av m, annars låt a få värdet av m. Med andra ord behåll vänster delintervall om vi har teckenväxling där, annars behåll höger delintervall.

Uppgift 3. Skriv en function som löser ekvationen $f(x) = 0$ genom intervallhalvering. Funktionen skall heta `min_bisect` och skall som indata ges en funktion som beräknar $f(x)$, ett intervall $[a, b]$ där $f(x)$ växlar tecken, dvs. där $f(a) \cdot f(b) < 0$, samt den noggrannhet lösningen skall bestämmas med. Funktionen skall som utdata ge mittpunkten m i det senaste intervallet som omsluter nollstället.

Funktionen skall anropas enligt `x = min_bisect(f,I,tol)`, där `f` är ett funktionshandtag till funktionen som beräknar $f(x)$, `I` är en radvektor som ger intervallgränserna a samt b och `tol` ett tal som anger önskad noggrannhet.

Uppgift 4. Använd nu din funktion `min_bisect` för att beräkna båda nollställena till funktionen i uppgift 1 med fem korrekta decimaler. Kontrollera att svaren är rimliga.

Uppgift 5. Kastbana utan luftmotstånd beskrivs av

$$y(x) = y_0 - \frac{g}{2v_0^2 \cos^2\theta}\left(x - \frac{v_0^2 \sin(2\theta)}{2g}\right)^2 + \frac{v_0^2 \sin^2\theta}{2g}$$

där v_0 är utkastfarten och θ är utkastvinkeln.

Hur långt når kastet om vi tar $y_0 = 1.85$, $v_0 = 10$ m/s och $\theta = 45°$? Du känner nog igen funktionen $y(x)$ för kastbanan från tidigare. Gör en funktionsfil för att beskriva den enligt

```
function y=kastbanan(x)
    y0=1.85; v=10; g=9.81; theta=45; t=theta*pi/180;
    a=g/(2*v^2*cos(t)^2); b=v^2*sin(2*t)/(2*g); c=v^2*sin(t)^2/(2*g);
    y=y0-a*(x-b).^2+c;
```

Rita en graf av kastbanan så du ser ungefär var marken träffas och använd sedan din funktion `min_bisect` för att bestämma träffpunkten noggrant. Kontrollera att svaret är rimligt.

3 Newtons metod

Antag att x_k är en approximation av ett nollställe till ekvationen $f(x) = 0$. Följ tangenten i punkten $(x_k, f(x_k))$, dvs.

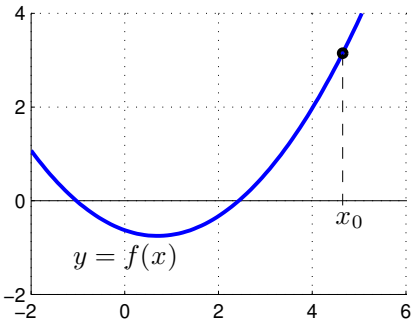
$$y = f(x_k) + f'(x_k)(x - x_k)$$

ned till x-axeln ($y = 0$) och tag skärningspunktens x-koordinat

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

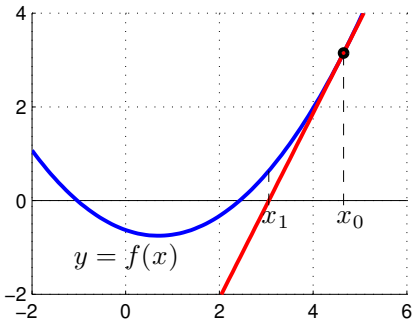
som en ny approximation av nollstället.

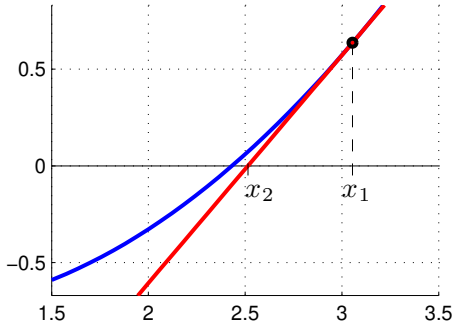
Vi ser på några steg med metoden: Starta med en approximation x_0 av ett nollställe till $f(x) = 0$.



Bilda tangenten $y = f(x_0) + f'(x_0)(x - x_0)$ till f i $x = x_0$ och tag dess skärningspunkt med x-axeln som en ny approximation

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$





Bilda tangenten $y = f(x_1) + f'(x_1)(x - x_1)$ i $x = x_1$ och tag dess skärningspunkt med x-axeln som en ännu nyare approximation

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

Som exempel tar vi: Lös ekvationen $f(x) = 0$ där $f(x) = \cos(x) - x$. En graf av funktionen (rita den gärna) visar att vi har ett nollställen nära $x_0 = 0.75$ som vi tar som startapproximation.

```
>> f=@(x)cos(x)-x; Df=@(x)-sin(x)-1;
>> x=0.75;
>> kmax=10; tol=0.5e-8;
>> for k=1:kmax
       h=-f(x)/Df(x);
       x=x+h;
       disp([x h])
       if abs(h)<tol, break, end
   end
```

```
0.739111138752579  -0.010888861247421
0.739085133364485  -0.000026005388094
0.739085133215161  -0.000000000149324
```

I kolumnen till vänster ser vi x_k-värdena och i den till höger ser vi motsvarande $f(x_k)$-värden. Vi ser att vi får snabb konvergens, dvs. vi får snabbt ett noggrant resultat. Iterationen avbryts eftersom vi har mer är åtta korrekta decimaler (felet mindre än $\frac{1}{2} \times 10^{-8}$).

Läs gärna i Persson-Böiers kapitel 4.5 om Newtons metod (eller Newton-Raphsons metod).

Uppgift 6. Låt $f(x) = x^3 - \cos(4x)$. Lös ekvationen $f(x) = 0$. Rita upp grafen till f för att se var ungefär lösningarna (skärningspunkterna) ligger. Hur många lösningar finns det? Läs av i grafiken en första approximation av en lösning för att sedan förbättra denna med Newtons metod. Rita ut lösningen med en liten ring. Upprepa tills du beräknat alla lösningar till ekvationen.

Det är praktiskt att packetera en metod genom att skriva ett program eller funktion som utför metoden. Vi gjorde det i förra avsnittet för intervallhalveringsmetoden och nu gör vi det även för Newtons metod.

Uppgift 7. Skriv en `function` som löser ekvationen $f(x) = 0$ med Newtons metod. Funktionen skall heta `min_newton` och skall som indata ges två funktioner, dels en som beräknar $f(x)$ dels en som beräknar $f'(x)$, en startapproximation av lösningen, samt den noggrannhet lösningen skall bestämmas med. Funktionen skall som utdata ge en approximation av nollstället som uppfyller noggrannhetskravet.

Funktionen skall anropas enligt `x = min_newton(f,Df,x0,tol)`, där `f` och `Df` är två funktionshandtag till de funktioner som beräknar $f(x)$ respektive $f'(x)$, `x0` är en startapproximation av lösningen och `tol` ett tal som anger önskad noggrannhet.

Uppgift 8. Pröva nu din funktion `min_newton` på följande ekvationer. Rita grafer och beräkna samtliga nollställen.

(a). $f(x) = 0.5(x-2)^2 - 2\cos(2x) - 1.5 = 0$ **(b).** $f(x) = x^3 - \cos(4x) = 0$

4 Färdigt program i MATLAB

Det finns en färdig funktion `fzero` för att lösa icke-linjära ekvationer. Genom att använda en kombination av intervallhalveringsmetoden (pålitlig men mycket långsam) och Newtons metod (kräver bra startapproximationer men mycket snabb) kan man förena de goda egenskaperna hos respektive metod och undvika de dåliga. (Derivatan i Newtons metod ersätts i `fzero` med en approximation.)

Funktion `fzero` används enligt något av alternativen

```
x=fzero(fun,x0)      x=fzero(fun,x0,opts)
```

där `fun` beskriver funktionen vi skall finna nollstället till, `x0` är en första approximation av nollstället eller ett intervall med teckenväxling som omsluter nollstället vi söker. I senare fallet kommer `fzero` garanterat finna en approximation av ett nollställ e.

Alternativet med `opts` använder vi då vi t.ex. vill ange hur noggrant lösningen skall beräknas. Man skapar vektorn `opts` med funktionen `optimset`, se hjälptexten.

För en sista gång ser vi på exemplet från inledningen. Vi har alltså

$$f(x) = 0.5\,(x-2)^2 - 2\,\cos(2x) - 1.5 = 0,$$

och i MATLAB löser vi med

```
>> f=@(x)0.5*(x-2).^2-2*cos(2*x)-1.5;
>> x0=4;
>> x=fzero(f,x0)
x =
    3.8664
```

Utskriften av beräkningsresultatet ovan gjordes med fem siffror. Vill vi få fler siffror utskrivna kan vi ge kommandot `format long` innan utskriften. Med `format short` får vi tillbaka den korta varianten. Så kallad scientific notation får vi med `format short e` respektive `format long e`.

Uppgift 9. Betrakta ekvationen

$$f(x) = \frac{3 + \sin(2x)}{1 + \mathrm{e}^{0.03x^2}} - 1.2 = 0$$

Rita graf och beräkna samtliga nollställen noggrant med `fzero`. Tänk på att använda elementvisa operationer.