Linjärisering och Newtons metod

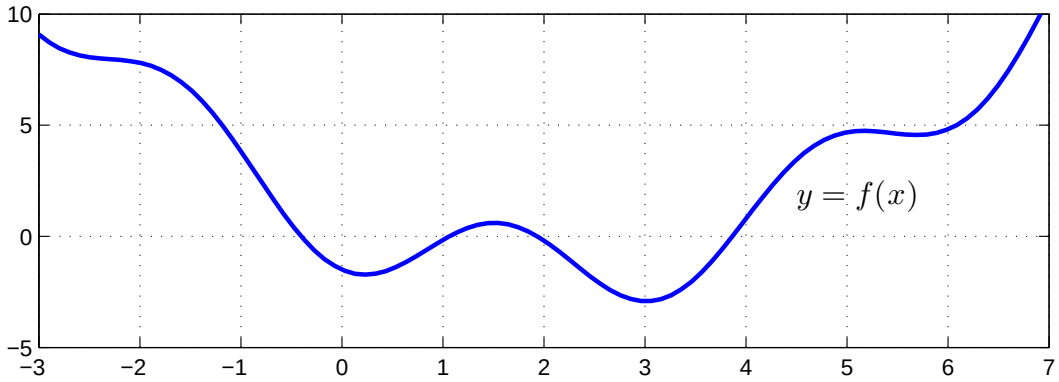
Analys och Linjär Algebra, del A, K1/Kf1/Bt1

1 Inledning

Vi skall fortsätta med att lösa ekvationer. I förra studio-övningen såg vi på intervallhalveringsmetoden. Den är pålitlig men ganska långsam. I denna studio-övning skall vi använda Newtons metod som är mycket snabbare, bara vi har en bra första approximation av en lösning. Som exempel kan vi ta,

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

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



Vi ser en lösning till $f(x) = 0$ som en punkt där grafen skär x-axeln. Vi kan grafiskt läsa av en första approximation av en lösning för att sedan förbättra denna med Newtons metod. Innan vi kan komma till Newtons metod måste vi dock först se på linjäriseringar av funktioner i en variabel.

2 Linjärisering

Vi skall se på linjärisering av en differentierbar (deriverbar) funktion i en variabel $f : \mathbb{R} \to \mathbb{R}$. Linjäriseringen av f runt punkten a ges av (Definition 8, Adams 4.9)

$$L(x) = f(a) + f'(a)(x-a).$$

Nära punkten a har vi $f(x) \approx L(x)$.

Vi påminner oss om att den räta linjen $y = L(x)$ är tangenten till kurvan $y = f(x)$ vid a.

Uppgift 1. Linjärisera $f(x) = 0.5(x-2)^2 - 2\cos(2x) - 1.5$ runt $a = -1$, $a = 2.5$ och $a = 2.8$. Rita upp funktionskurvan tillsammans med tangenterna i a för de olika a-värdena.

3 Newtons metod

Låt $f : \mathbb{R} \to \mathbb{R}$ vara en deriverbar funktion. Vi skall lösa ekvationen $f(x) = 0$ med Newtons metod (Adams 4.2).

Antag att vi har en approximativ lösning x_k och vi vill hitta en bättre approximation x_{k+1}. Vi bildar linjäriseringen av f i x_k:

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

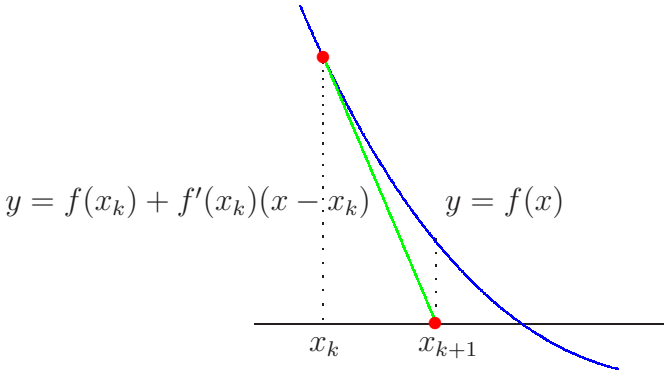
och löser $L(x) = 0$ istället för $f(x) = 0$.

$$f(x_k) + f'(x_k)(x - x_k) = 0 \tag{1}$$

Lösningen får bli nästa approximation:

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

Detta är Newtons metod. Geometriskt betyder (1) att vi följer tangenten och hittar x_{k+1} där denna skär x-axeln.



Som stoppvillkor för iterationen tar vi

$$|x_{k+1} - x_k| \leq \text{tol}$$

Det betyder att vi accepterar x_{k+1} om ändringen i sista iteration är mindre än toleransen. Vi tillåter maximalt k_{max} iterationer ($k_{max} = 10$ är rimligt).

Som ett litet exempel tar vi $f(x) = \cos(x) - x$ och vi skall lösa $f(x) = 0$. En graf (rita den gärna) visar att vi har ett nollställe och vi tar $x_0 = 1$ som startapproximation.

```
>> f=@(x)cos(x)-x;
>> Df=@(x)-sin(x)-1;
>> x=1;
>> 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
```

```
0.750363867840244  -0.249636132159756
0.739112890911362  -0.011250976928882
0.739085133385284  -0.000027757526078
0.739085133215161  -0.000000000170123
```

Uppgift 2. 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.

Uppgift 3. För att beräkna kvadratroten ur ett tal c, med upprepad addition och division, använde vi iterationsformeln

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{c}{x_k}\right),\ k = 0, 1, 2, \cdots, x_0 = c$$

i "Programmering i MATLAB", exempel 5. Visa att man får fram formeln genom att använda Newtons metod för att lösa ekvationen $f(x) = x^2 - c = 0$.

Uppgift 4. 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 innehålla en hjälptext som beskriver hur den skall användas. Skriver vi `help min_newton` i Command Window så skall det se ut något liknande:

```
>> help min_newton
min_newton - beräknar nollställe till f(x) givet startapproximation x0.
  Syntax:
          x = min_newton(f,Df,x0,tol)
  Argument:
          f   - funktionshandtag: pekar på namnet till en funktionsfil eller
                till en anonym funktion. T.ex. f=@funk eller f=@(x)cos(x)-x
          Df  - funktionshandtag: pekar på namnet till en funktionsfil eller
                till en anonym funktion som ger derivatan av f.
                T.ex. f=@Dfunk eller Df=@(x)-sin(x)-1
          x0  - ett tal som ger en startapproximation av nollstället.
          tol - positivt tal som anger önskad noggrannhet för nollstället.
  Returnerar:
          x   - ett tal som ger approximativt nollställe.
  Beskrivning:
          Programmet beräknar ett approximativt nollställe till f(x) med
  Newtons metod.
  Exempel:
          x = min_newton(@(x)cos(x)-x,@(x)-sin(x)-1,1.0,1e-5)
```

För att underlätta lite finns på studiohemsidan ett programskal `min_newton.m` att utgå ifrån.

Uppgift 5. Pröva nu din funktion `min_newton` på ekvationerna

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

4 Modifiering av Newtons metod

En dålig startapproximation kan leda till att Newtons metod divergerar (inte konvergerar), då är det lämpligt med dämpad Newton

$$x_{k+1} = x_k - \alpha_k \frac{f(x_k)}{f'(x_k)},\ k = 0, 1, \cdots .$$

Dämpningsfaktorn α_k väljs så att $|f(x_{k+1})| < |f(x_k)|$. Man kan t.ex. börja med $\alpha_k = 1$, på försök ta ett steg i iterationen, om vi har fått en minskning av $|f|$ accepterar vi steget. I annat fall halverar vi successivt α_k och gör nya försök, tills vi har en minskning av $|f|$.

Om vi inte vill beräkna derivator, så kan vi approximera dem med differenskvoter,

$$f'(x) \approx \frac{f(x+\delta) - f(x-\delta)}{2\delta},$$

för lämpligt valt litet positivt tal δ.

I MATLAB OPTIMIZATION TOOLBOX finner vi `fzero` som använder dessa modifieringar. 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
```

Uppgift 6. Lös ekvationen $f(x) = (x-5)\,\mathrm{e}^x + 5 = 0$ med `fzero`. Rita graf och se efter var nollställena ligger och beräkna det intressanta. Läs gärna helt kort bakgrunden till ekvationen under länken ”Svartkroppsstrålning” på studiohemsidan.

5 Redovisning

Denna vecka skall uppgifterna 1-6 redovisas för handledaren.