Mer om beräkningsmetoder

1 Inledning

Förra läsperioden såg vi på metoder för beräkning av nollställen och integraler. Det är praktiskt att packetera en metod genom att skriva ett program eller funktion som utför metoden. Vi skall använda våra nya kunskaper i MATLAB för att göra detta.

2 Newtons metod

Vi påminner oss hur vi använde Newtons metod för beräkna nollställen till funktioner. Först ritar vi en graf av funktionen för att avgöra hur många nollställen som finns, ungefär var de ligger och vilka vi vill beräkna noggrant.

Newtons metod: Givet en approximation x_0 av ett nollställe till $f(x)$. 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

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

Vi får iterationsformeln

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

för att successivt beräkna nollstället allt noggrannare.

Som exempel tar vi $f(x) = \cos(x) - x$. En graf visar ett nollställe nära $x_0 = 0.75$ som vi beräknar

```
>> 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;
       if abs(h)<tol, break, end
   end
```

Iterationen går maximalt $k_{max} = 10$ steg och avbryts om vi får mer är åtta korrekta decimaler (felet mindre än $tol = \frac{1}{2} \times 10^{-8}$).

2.1 Eget program för Newtons metod i MATLAB

Det är praktiskt att packetera en metod genom att skriva ett program eller funktion som utför metoden. Vi skall göra det för Newtons metod.

Uppgift 1. 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. Utgå från programskalet `min_newton.m` som kan hämtas från kurshemsidan.

Funktionen skall vidare 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)
```

När ni väl har gjort funktionen `min_newton` kan vi beräkna nollstället i exemplet från förra avsnittet enligt

```
>> f=@(x)cos(x)-x; Df=@(x)-sin(x)-1;
>> x0=0.75;
>> x=min_newton(f,Df,x0,0.5e-8)
x=
   0.7391
```

Skall vi sedan beräkna nollställen till en ny funktion så är det bara att beskriva den nya funktionen och dess derivata, rita graf och läsa av approximation av nollställe och använda `min_newton` igen.

Uppgift 2. 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).** $f(x) = x^3 - \cos(4x) = 0$

3 Beräkning av integraler

I förra läsperioden såg vi hur vi kunde approximera en integralen $\int_a^b f(x)\,dx$ med Riemannsummor på lite olika sätt.

Vi gjorde en likformig indelning av intervallet $a \le x \le b$

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$$

så att vi fick n lika långa delintervall $x_{i-1} \le x \le x_i$ med bredden $h = \frac{b-a}{n}$.
Sedan delade vi upp integralen i en summa av delintegraler över varje delintervall

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{x_{i-1}}^{x_i} f(x)\,dx$$

Om vi approximerar $f(x)$ med $f(m_i)$ i intervallen $x_{i-1} \le x \le x_i$, där m_i är mittpukterna i intervallen, får vi mittpunktsmetoden:

$$\int_a^b f(x)\,dx \approx M_n = \sum_{i=1}^{n} h\, f\left(\frac{x_{i-1}+x_i}{2}\right)$$

Antag vi vill beräkna $\int_0^1 x\sin(x)\,dx$ med mittpunktsmetoden och att vi tar $n = 100$ delintervall.
Vi skulle kunna använda en `for`-sats för att beräkna summan men vi genererar hellre en vektor av alla funktionsvärdena $f(x_i)$ och sedan summerar dessa enligt

```
>> n=100;
>> f=@(x)x.*sin(x); a=0; b=1;
>> x=linspace(a,b,n+1); h=(b-a)/n;
>> m=(x(1:n)+x(2:n+1))/2;
>> q=sum(h*f(m))
```

Vi måste tänka på att använda elementvisa operationerna när vi beskriver $f(x)$, precis som när vi ritar en graf.

3.1 Eget program för integralberäkning i MATLAB

Åter igen är det praktiskt att packetera en metod genom att skriva ett program eller funktion som utför metoden. Vi skall göra det med metoderna ovan.

Uppgift 3. Skriv en funktion `min_integral` med anropet `q=min_integral(f,I,n,k)` som beräknar en integral approximativt. Använd det programskal som finns på kurshemsidan. In- och ut-variablerna förklaras i programskalet.

Uppgift 4. Testa ditt program `min_integral` på följande integraler. Variera metodval och antal delintervall n för att få en uppfattning om hur stort n måste väljas för att få en god noggranhet i approximationen.

(a). $\int_0^1 \exp(-x^2)\,dx$ **(b).** $\int_{-1}^1 \frac{1}{1+x^2}\,dx$ **(c).** $\int_0^1 \tan(\sqrt{x})\,dx$