Beatrice Frock, NADA

2D1212 NumProg, Kompletterande material

Kap.3. Approximation: interpolation och linjära minstakvadratmetoden.

Problem: att representera en funktion givet en tabell över kända punkter.
T.ex. önskar vi approximera funktionen i mellanliggande punkter, eller derivera eller integrera ett uttryck för funktionen.

I. Interpolation: Ett interpolationspolynom är ett polynom av grad n som går exakt genom $n+1$ givna punkter.
Det existerar ett enda sådant polynom, men det finns olika sätt att representera och beräkna det interpolerande polynomet.
Tre viktiga former är naiva, Newton resp. Lagrange.

Naiva formen

I ett polynom av den naiva formen

$$P(x) = a_1 + a_2x + a_3x^2 + \ldots + a_{n+1}x^n$$

kan vi bestämma koefficienterna genom att helt enkelt ställa upp $(n+1)$ ekvationer i $(n+1)$ obekanta. Systemet löses med Gausselimination eller \ i Matlab. Funktionen `Polyfit` är också ett alternativ. Detta är inte en effektiv metod, och på grund av illa-konditionering kan noggrannheten bli dålig. Systemmatrisen blir fylld, en s.k. vandermonde matris.

Exempel. Temperaturen under jordytan.

Följande mätvärden finns för temperaturen $T(^oC)$ som funktion av djupet under jordytan $d(km)$ i ett geologiskt sett gammalt kontinentalområde, enligt geofysiska mätningar och beräkningar:

d	0	50	100	150	200	250
T	0	491	853	1162	1358	1442

Skriv ett Matlab-program som bestämmer koefficienterna i det interpolationspolynom av lämpligt gradtal som går genom dessa givna punkter. Ditt program ska även generera en plotbild över interpolationskurvan, med de givna 6 punkterna markerade, samt skriva ut det interpolerade värdet av $T(71.6)$, dvs då $d = 71.6$ km.

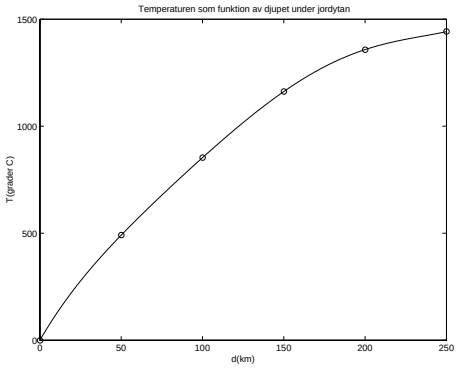
Matlab-kod för att bestämma koefficienterna i det femtegrads interpolationspolynomet, samt plotta resultatet:

```
clear, clf
x=[0:50:250]';
T=[0 491 853 1162 1358 1442]';
A=[ones(size(x)) x x.^2 x.^3 x.^4 x.^5];
% kolumnvis generering av systemmatrisen
c=A\T;

Tresult=c(1)+c(2)*71.6+c(3)*71.6^2+c(4)*71.6^3+c(5)*71.6^4+c(6)*71.6^5;
disp('Temperaturen T(71.6)='), Tresult

xplot=0:250;
Tplot=c(1)+c(2)*xplot+c(3)*xplot.^2+c(4)*xplot.^3+...
                                        c(5)*xplot.^4+c(6)*xplot.^5;
%  Alternativ:  Aplot=[ones(size(xplot)) xplot xplot.^2 xplot.^3 ...
                                        xplot.^4 xplot.^5];
%  följt av:  Tplot=Aplot*c;

plot(x,T,'o',xplot,Tplot)
title('Temperaturen som funktion av djupet under jordytan')
xlabel('d(km)')
ylabel('T(grader C)')
```



Högre-grads interpolationspolynom har en tendens att uppvisa kraftiga oscillationer nära intervallets ändpunkter. Detta kallas för Runges fenomen, och kan undvikas genom att i stället välja styckvis lägre grads interpolationspolynom (eller med en lämpligare placering av interpolationsnoder nära kanterna).

Newton interpolation

Det $n-$te grads polynom genom $(n+1)$ givna punkter kan representeras enligt Newtons form.

$$P_n(x) = c_1 + c_2(x - x_1) + c_3(x - x_1)(x - x_2) + \ldots + c_{n+1}(x - x_1) \cdot \ldots \cdot (x - x_n)$$

Beräkningarna leder till ett triangulärt linjärt ekvationssystem för de okända koefficienterna, och Newton är även lämplig för enkla handräkningar.

Det viktigaste fördelen är dock att det är enkelt att lägga till interpolationspunkter (och höja graden). Endast de koefficienter som tillkommer behöver räknas (de tidigare förblir oförändrade). Jämför Taylorutveckling för en funktion.
Trunkeringsfelet kan lätt skattas med "första försummade termen" i Newtons ansats (dvs nästa term, om en punkt tillkommer och graden skulle höjas med 1).

$$e_{trunk} = |P(x) - f(x)| = |\frac{f^{(n+1)}(\xi)}{(n+1)!}(x - x_1) \cdot \ldots \cdot (x - x_n)(x - x_{n+1})|$$

med $x_1 \leq \xi \leq x_{n+1}$.

Exempel. Man vill lägga ett polynom genom punkterna i tabellen nedan:

x	10001	10002	10003
g(x)	10	12	13

Föreslå en lämplig ansats för det sökta polynomet och bestäm dess koefficienter (handräkning).

Newtons ansats för interpolationspolynomet är lämplig här

$$P(x) = c_1 + c_2(x - 10001) + c_3(x - 10001)(x - 10002)$$

Tabellvärdena sätts in i uttrycket och ger succesivt
$c_1 = 10$,
$10 + c_2(10002 - 10001) = 12$ vilket ger $c_2 = 2$
$10 + 2(10003 - 10001) + c_3(10003 - 10001)(10003 - 10002) = 13$ vilket ger $c_3 = -1/2$. Polynomet är således

$$P(x) = 10 + 2(x - 10001) - \frac{1}{2}(x - 10001)(x - 10002)$$

II. Linjära minstakvadratmetoden

Läs 3.4 i Q&S. En rät linje anpassas till ett stort antal datapunkter, så väl som möjligt i minsta kvadratmetodens mening. Vi har allmänt flera ekvationer än obekanta, ett s.k. överbestämt problem.
Linjen behöver inte nödvändigtvis gå igenom någon given punkt (till skillnad från interpolation). Det anpassningskriteriet som används är att felkvadratsumman ska minimeras, dvs minimera summan av avvikelsernas kvadrater (avvikelserna mellan den anpassade räta linjen och de givna värdena). Problemet kan inte lösas exakt, utom då alla punkterna ligger på den räta linjen.
För en rät linje, enligt den matematiska framställningen i kursboken, deriverar man m.a.p. de obekanta koefficienterna a och b i uttrycket

$$g(a, b) = \sum_{i=0}^{n} (f(x_i) - (a + bx_i))^2$$

och sätter derivatorna $\frac{\partial q}{\partial a}$ och $\frac{\partial q}{\partial b}$ till noll för bestämning av minimum.

Detta kan generaliseras till högre gradspolynom (inte enbart en rät linje, som har gradtal = 1).
Matlab-kommandot `polyfit(x,y.m)` beräknar automatiskt det anpassade polynomet av grad m, och man får minsta kvadratanpassning då antalet punkter är större än antalet obekanta (och antalet ekvationer).

Linjära minstakvadratmetoden. Matrisformulering
Framställningen i boken gäller endast polynom. Vi kan i stället anpassa ett uttryck som är en linjär kombination av generella basfunktioner $\phi_i(x)$,

$$f^*(x) = c_1\phi_1(x) + c_2\phi_2(x) + \ldots + c_n\phi_n(x)$$

Exempel på basfunktioner:
(1) $\phi_1 = 1$, $\phi_2 = x$, $\phi_3 = x^2$, dvs f^* är en parabel.
(2) $\phi_1 = \sin x$, $\phi_2 = \cos 2x$, dvs vi anpassar modellen

$$f^* = c_1 \sin x + c_2 \cos 2x$$

(3) $\phi_1(t) = e^{-2t}$, $\phi_2(t) = e^{-5t}$, dvs anpassa modellen

$$f^*(t) = c_1 e^{-2t} + c_2 e^{-5t}$$

När antalet punkter > antalet obekanta får vi ett överbestämt linjärt ekvationssystem, $Ac \approx f$, av dimension $m \times n$, där $m > n$. Kolumnerna i A kallas **basvektorer**. Systemet löses med \ i Matlab. I handräkning ställer man upp det s.k. **normalekvationssystemet** $A^TAc = A^Tf$. Därmed reduceras problemet till ett kvadratiskt $(n \times n)$ linjärt ekvationssystem, som kan lösas m.h.a. Gausselimination. Ordet "normal" här syftar på en viss ortogonalitet. Både med \ i Matlab, och med lösning av normala ekvationerna i handräkning, fås den bästa möjliga anpassningen i minsta kvadratmetoden, dvs felkvadratsumman minimeras av metoden. Man kan även visa att **residualvektorn** $f - Ac$ är ortogonal mot basvektorerna i A, eftersom $A^T(f - Ac) = 0$.

Exempel 1. Givet de tre y-värdena, $y(-1) = 1,\ y(0) = 0,\ y(1) = 2$. Anpassa en rät linje $y = kx + m$ med minstakvadratmetoden till y-värdena. Bestäm k och m.
Ställ upp det kvadratiska uttryck i k och m, som minimeras av minstakvadratmetoden.

Sätt in datavärdena i den räta linjens ekvation så får vi

$$\begin{aligned} k \times (-1) + m &= 1 \\ k \times (0) + m &= 0 \\ k \times (1) + m &= 2 \end{aligned}$$

vilket vi med matris- och vektorbeteckningar skriver

$$Ac \approx b, \quad \text{där} \quad A = \begin{pmatrix} -1 & 1 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}, \; c = \begin{pmatrix} k \\ m \end{pmatrix}, \; b = \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix}$$

Minstakvadratösningen till detta överbestämda ekvationssystem ges av normalekvationerna $A^TAc = A^Tb$ dvs

$$\begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} k \\ m \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$$

Lösningen blir $k = 1/2$ och $m = 1$.
I minstakvadratmetoden minimeras felkvadratsumman, i detta fall

$$g(k, m) = (1 + k - m)^2 + m^2 + (2 - k - m)^2$$

Exempel 2. Antag att en funktion $f(x) = a \cdot \sin(0.2\pi x) + b \cdot \sin(0.02\pi x)$ ska anpassas till en uppsättning mätdata $(x_i, f_i), i = 1, 2, ..., 100$.
Formulera det linjära ekvationssystem som ger minstakvadratlösningen till problemet. Vilken dimension har detta ekvationssystem? Ange med formler hur elementen i systemmatrisen och komponenterna av högerledsvektorn ser ut.
Ställ upp det uttryck i variablerna a och b som i a. minimeras av minstakvadratmetoden.

Normalekvationssystemet har dimension 2x2 (två obekanta i det linjära överbestämda systemet).
Överbestämda systemet:

$$Ac \approx f, \quad \text{där} \quad A = \begin{pmatrix} \sin(0.2\pi x_1) & \sin(0.02\pi x_1) \\ \sin(0.2\pi x_2) & \sin(0.02\pi x_2) \\ \vdots & \vdots \\ \sin(0.2\pi x_{100}) & \sin(0.02\pi x_{100}) \end{pmatrix}, \; c = \begin{pmatrix} a \\ b \end{pmatrix}, \; f = \begin{pmatrix} f_1 \\ \vdots \\ f_{100} \end{pmatrix}$$

Minstakvadratösningen till detta överbestämda ekvationssystem ges av normalekvationerna $A^TAc = A^Tf$. Elementen i normalekvationssystemets systemmatris och högerled beräknas på följande sätt:

$$A^TA = \begin{pmatrix} \sin^2 0.2\pi x_1 + \ldots + \sin^2 0.2\pi x_{100} & \sin 0.02\pi x_1 \cdot \sin 0.2\pi x_1 + \ldots + \sin 0.02\pi x_{100} \cdot \sin 0.2\pi x_{100} \\ \sin 0.02\pi x_1 \cdot \sin 0.2\pi x_1 + \ldots + \sin 0.02\pi x_{100} \cdot \sin 0.2\pi x_{100} & \sin^2 0.02\pi x_1 + \ldots + \sin^2 0.02\pi x_{100} \end{pmatrix}$$

och

$$A^Tf = \begin{pmatrix} f_1 \sin 0.2\pi x_1 + \ldots + f_{100} \sin 0.2\pi x_{100} \\ f_1 \sin 0.02\pi x_1 + \ldots + f_{100} 0.02\pi x_{100} \end{pmatrix}$$

I minstakvadratmetoden minimeras felkvadratsumman

$$g(a, b) = \sum_{i=1}^{100} (a \cdot \sin(0.2\pi x_i) + b \cdot \sin(0.02\pi x_i) - f_i)^2$$

Centrering
För stora värden på den oberoende variabeln (t.ex. x) kan normalekvationssystemet bli illa-konditionerat, dvs känsligt för små fel såsom avrundningsfel. Det är fördelaktigare att **centrera** kring medelvärdet av $x-$värdena.
Exempelvis om x = 100,101,102,103,104, kan ett anpassande andragradspolynom skrivas i formen $f^* = c_1 + c_2(x - 102) + c_3(x - 102)^2$. I handräkning blir beräkningarna dessutom betydligt enklare.
Enkla icke-linjära problem
En generell metod för icke-linjära minsta kvadratanpassning (där de obekanta ingår icke-linjärt i uttrycket) studeras inte i denna kurs.
Dock kan vissa enkla icke-linjära problem transformeras till linjär form (t.ex. genom logaritmering eller invertering). Minsta kvadratlösningen blir då en hyfsad approximation till den egentliga, icke-linjära minsta kvadratlösningen.
Ett exempel kommer att visas på en föreläsning.