



KTH Computer Science  
and Communication

## Föreläsning 5

# Approximationsteori

Låt  $f$  vara en kontinuerlig funktion som vi vill approximera med en enklare funktion  $\tilde{f}(x)$ . Vi kommer använda två olika approximationsmetoder: interpolation och minstrakvadratanpassning. Vi börjar med interpolation och det enklaste fallet då  $\tilde{f}(x)$  är ett polynom.

## 1 Polynominterpolation

Vid polynominterpolation vill vi hitta ett polynom  $p(x)$  som överensstämmer med  $f$  i ett antal noder ( $x$ -koordinater),  $x_0 < x_1 < \dots < x_n$ , dvs

$$p(x_j) = f(x_j), \quad j = 0, \dots, n.$$

Man kan först fråga sig om detta verkligen alltid är möjligt. Svaret är ja och ges i precis form av satsen på sid 347 i Bradie. Vi behöver bara se till att polynomets gradtal är minst lika stort som antal interpolationsnoder minus ett. I vårt fall är antal noder  $n + 1$  och vi behöver alltså ett polynom av gradtal minst  $n$ . (Om vi väljer gradtalet precis lika med  $n$  är polynomet också unikt; tex, givet två punkter finns det exakt en rak linje som passerar igenom båda två.)

Nästa fråga är hur man kan få fram polynomet numeriskt. För att göra det antar vi att gradtalet är  $n$  och ansätter

$$p(x) = c_0 + c_1x + c_2x^2 + \dots + c_nx^n. \quad (1)$$

Vi vill hitta värdena för koefficienterna  $c_k$ . Villkoren för interpolation blir

$$p(x_j) = \sum_{k=0}^n c_k x_j^k = f(x_j), \quad j = 0, \dots, n. \quad (2)$$

Detta kan skrivas som ett linjärt ekvationssystem för koefficienterna  $c_k$ ,

$$A\mathbf{c} = \mathbf{f}, \quad (3)$$

där

$$\mathbf{c} = \begin{pmatrix} c_0 \\ \vdots \\ c_n \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}, \quad A = \begin{pmatrix} 1 & x_0 & \dots & x_0^n \\ 1 & x_1 & \dots & x_1^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \dots & x_n^n \end{pmatrix}.$$

Notera att  $\mathbf{c} \in \mathbb{R}^{n+1}$ ,  $\mathbf{f} \in \mathbb{R}^{n+1}$  och  $A \in \mathbb{R}^{(n+1) \times (n+1)}$ . Matrisen  $A$  kallas en *Vandermonde*-matris för noderna. Vi kan nu beräkna de okända koefficienterna  $c_k$  genom att lösa det linjära ekvationssystemet.

Ett problem med denna metod är att elementen i Vandermondematrisen kan bli väldigt stora om vi har många noder, dvs stort  $n$ . Det gör matrisen illa konditionerad och metoden blir opålitligt för stora problem. En bättre variant är att ändra ansatsen (1) till den så kallade Newton-ansatsen,

$$p(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + c_3(x - x_0)(x - x_1)(x - x_2) + \cdots + c_n \prod_{j=0}^{n-1} (x - x_j). \quad (4)$$

Denna ansats utnyttjar mer information om problemet när den inkluderar nodvärdena. Den ger fortfarande ett polynom av gradtal  $n$ . På samma sätt som ovan leder den till ett linjärt ekvationssystem för koefficienterna  $c_k$ , med matrisen

$$A_{\text{Newton}} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & (x_1 - x_0) & 0 & \cdots & 0 \\ 1 & (x_2 - x_0) & (x_2 - x_0)(x_2 - x_1) & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & (x_n - x_0) & (x_n - x_0)(x_n - x_1) & \cdots & \prod_{j=0}^{n-1} (x_n - x_j) \end{pmatrix}.$$

Denna matris är dels bättre konditionerad än Vandermondematrisen, dels triangulär och det linjära ekvationssystemet är därför billigare att lösa.

Vi ska slutligen diskutera felet vi gör vid polynominterpolation. Vi antar att alla noder ligger i intervallet  $[a, b]$ . Man kan då visa att (se satsen på sid 348 i Bradie),

$$f(x) - p(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} (x - x_0) \cdots (x - x_n) \quad (5)$$

för något  $\xi \in [a, b]$ . (Detta kan ses som en generalisering av medelvärdessatsen, vilken ges för  $n = 0$ .) Om noderna är ekvidistant placerade, dvs om

$$x_j = a + jh, \quad h = \frac{b - a}{n}.$$

kan man också visa att

$$|(x - x_0) \cdots (x - x_n)| \leq n! \frac{h^{n+1}}{4}. \quad (6)$$

Definera felet  $E_n$  vid  $n$  noder som det maximala felet i intervallet,

$$E_n = \max_{a \leq x \leq b} |f(x) - p(x)|.$$

För ekvidistanta noder ger då (5) och (6) tillsammans att

$$E_n \leq \max_{a \leq \xi \leq b} \frac{|f^{(n+1)}(\xi)|}{4(n+1)!} h^{n+1}. \quad (7)$$

Betyder då detta att felet går mot noll när vi tar många punkter, dvs går  $E_n \rightarrow 0$  då  $n \rightarrow \infty$ ? Nej, i allmänhet inte. Visserligen går  $h^{n+1} \rightarrow 0$ , men ofta går  $\max |f^{(n+1)}| \rightarrow \infty$  snabbare och  $E_n$  kan inte begränsas med (7). För ekvidistant interpolation får interpolationspolynomet vilda oscillationer vid intervallets ändpunkter när  $n$  blir stort. Detta kallas *Runges fenomen* och illustreras i Bradie på sidorna 386–387.

Slutsatsen är att ekvidistant interpolation med höga gradtal inte är bra. Om man själv kan välja noder är ett bättre alternativ ta noderna icke ekvidistant. Det optimala valet ges av de sk

Chebyshev-noderna, se sid 379 i Bradie. Ett annat mycket praktiskt alternativ är att använda styckvis polynominterpolation med lågt gradtal. I detta fall delar vi in intervallet  $[a, b]$  i många små intervall och interpolerar med lågt gradtal i vart och ett av dessa. Det enklaste fallet är styckvis linjär interpolation (se kap 5.5 i Bradie) där vi använder förstgradspolynom för att interpolera mellan varje nodpar, dvs vi approximerar genom att "dra ett rakt streck mellan punkterna". Vi kan använda samma feluppskattning som tidigare, (7), och applicera den på varje enskilt intervall. Den ger då

$$E_{\text{styckvis}} = \max_j \max_{x_j \leq x \leq x_{j+1}} |f(x) - p_{\text{styckvis}}(x)| \leq \max_j \max_{x_j \leq \xi \leq x_{j+1}} \frac{|f''(\xi)|}{8} h^2 \leq \max_{a \leq \xi \leq b} \frac{|f''(\xi)|}{8} h^2.$$

Här är  $n$  i (7) konstant lika med ett och  $h$  går mot noll oberoende av  $n$ . Därmed konvergerar  $E_{\text{styckvis}} \rightarrow 0$  som  $O(h^2)$ . Noggrannhetsordningen är alltså två för styckvis linjär interpolation. I allmänhet för styckvis interpolation med gradtal  $p$  är noggrannhetsordningen  $p + 1$ .

*Kommentarer:*

- Lagrange-polynomen för noderna är definierade som

$$L_j(x) := \prod_{k=0, k \neq j}^n \frac{x - x_k}{x_j - x_k}, \quad j = 0, \dots, n.$$

De är polynom av gradtal  $n$  med egenskapen att

$$L_j(x_k) = \begin{cases} 1, & j = k, \\ 0, & j \neq k. \end{cases}$$

Med dessa kan interpolationsproblemet enkelt lösas,

$$p(x) = \sum_{j=0}^n L_j(x) f(x_j).$$

Denna form för interpolationspolynomet är dock krånglig att använda i numeriska beräkningar. Lagrange-polynomen är snarare ett teoretiskt verktyg och har ett stort värde vid analys av approximationsproblem.

- Några praktiska MATLAB-funktioner för interpolation är `c=polyfit(x,f,n)`, som ger  $c_k$ -koefficienterna när  $x_k$  och  $f(x_k)$  ligger i  $(n + 1)$ -vektorerne `x` och `f`; `polyval(c,x)` som evaluerar ett polynom med koefficienterna `c` i punkten `x`; samt `interp1(xnod,fnod,x,typ)` som evaluerar styckvis polynominterpolation av `(xnod,fnod)` i punkten `x`, med varierande interpolationstyp som bestäms av sista argumentet. Tex kan `typ` vara `'linear'`, `'cubic'`, eller `'spline'`. Se MATLABs `help`-kommando för mer info.

## 2 Minstakvadratanpassning

Vid minstakvadratanpassning söker vi en enkel funktion som approximerar  $f(x)$  genom att minimera den totala avvikelsen från  $f$  i ett antal givna noder. Mer precist söker vi en funktion  $\tilde{f}$  som minimerar "felkvadratsumman",

$$r = \sum_{j=0}^n (\tilde{f}(x_j) - f(x_j))^2.$$

Vi börjar med att anta att  $\tilde{f}$  är ett polynom. Om  $f$  tex är en snäll funktion med brus eller andra störningar överlagrade kommer vanlig polynominterpolation, både styckvis och direkt, ge en mycket oscillerande approximation. Minstakvadratlösningen med ett polynom av lågt gradtal ger en snällare lösning.

För att beräkna minstakvadratlösningen för ett polynom gör vi som tidigare ansatsen

$$p(x) = c_0 + c_1x + c_2x^2 + \cdots + c_mx^m,$$

där  $m$  nu är mindre än  $n$ . Vi kan då inte längre hitta  $c_k$  så att  $p(x_j) = f(x_j)$  för alla  $j$ . Istället vill vi minimera skillnaden. Liksom ovan kan vi skriva denna på matrisform,

$$A\mathbf{c} - \mathbf{f} = \begin{pmatrix} p(x_0) - f(x_0) \\ \vdots \\ p(x_n) - f(x_n) \end{pmatrix}, \quad \left\{ \overbrace{\begin{array}{|c|} \hline \phantom{0} \\ \hline \end{array}}^{m+1} \begin{array}{|c|} \hline \phantom{0} \\ \hline \end{array} - \begin{array}{|c|} \hline \phantom{0} \\ \hline \end{array} \right\}_{n+1}. \quad (8)$$

Matrisen blir alltså nu rektangulär,  $A \in \mathbb{R}^{(n+1) \times (m+1)}$ ,  $\mathbf{c} \in \mathbb{R}^{m+1}$  och  $\mathbf{f} \in \mathbb{R}^{n+1}$ . Systemet  $A\mathbf{c} = \mathbf{f}$  är överbestämt och saknar då i allmänhet lösningar. Vi noterar dock att eftersom

$$\|A\mathbf{c} - \mathbf{f}\|_2^2 = \sum_{j=0}^n (p(x_j) - f(x_j))^2 = r,$$

är minimering av felkvadratsumman  $r$  ekvivalent med att lösa  $A\mathbf{c} = \mathbf{f}$  i minstakvadratmening, dvs hitta den vektor  $\mathbf{c}$  som minimerar den euklidiska normen av residualen,

$$\|A\mathbf{c} - \mathbf{f}\|_2 = \min_{\mathbf{b} \in \mathbb{R}^{m+1}} \|A\mathbf{b} - \mathbf{f}\|_2. \quad (9)$$

Vi ska nu visa att lösningen  $\mathbf{c}$  till minstakvadratproblemet ges av *normalekvationerna*

$$\boxed{A^T A \mathbf{c} = A^T \mathbf{f}}. \quad (10)$$

Notera att  $A^T A \in \mathbb{R}^{(m+1) \times (m+1)}$  är en kvadratisk matris och systemet har alltid en lösning när kolumnerna i  $A$  är linjärt oberoende (se nedan).

Normalekvationerna kan härledas på följande sätt. Antag att  $\mathbf{c}$  löser (10). Eftersom  $\|\mathbf{x}\|_2^2 = \mathbf{x}^T \mathbf{x}$  har vi då för alla  $\mathbf{b} \in \mathbb{R}^{m+1}$ ,

$$\begin{aligned} \|A\mathbf{b} - \mathbf{f}\|_2^2 - \|A\mathbf{c} - \mathbf{f}\|_2^2 &= \mathbf{b}^T A^T A \mathbf{b} - 2\mathbf{b}^T A^T \mathbf{f} + \mathbf{f}^T \mathbf{f} - \mathbf{c}^T A^T A \mathbf{c} + 2\mathbf{c}^T A^T \mathbf{f} - \mathbf{f}^T \mathbf{f} \\ &= \mathbf{b}^T A^T A \mathbf{b} - 2\mathbf{b}^T A^T \mathbf{f} - \mathbf{c}^T A^T A \mathbf{c} + 2\mathbf{c}^T A^T \mathbf{f} = \{\text{utnyttja (10)}\} \\ &= \mathbf{b}^T A^T A \mathbf{b} - 2\mathbf{b}^T A^T A \mathbf{c} + \mathbf{c}^T A^T A \mathbf{c} \\ &= \|A(\mathbf{b} - \mathbf{c})\|_2^2 \geq 0. \end{aligned}$$

Med andra ord är

$$\|A\mathbf{b} - \mathbf{f}\|_2^2 \geq \|A\mathbf{c} - \mathbf{f}\|_2^2 \quad \forall \mathbf{b} \in \mathbb{R}^{m+1},$$

vilket visar att  $\mathbf{c}$  faktiskt minimerar residualen och uppfyller (9).

Alternativt kan vi härleda (10) med vanlig flervariabelanalys. Definiera

$$\Phi(\mathbf{b}) := \|A\mathbf{b} - \mathbf{f}\|_2^2.$$

Låt  $\nabla$  betyda gradienten med avseende på  $\mathbf{b}$ . Då uppfyller minimumet  $\mathbf{c}$

$$\nabla\Phi(\mathbf{c}) = 0.$$

Vi utnyttjar att

$$\nabla(\mathbf{b}^T U \mathbf{b}) = 2U\mathbf{b}, \quad \nabla \mathbf{u}^T \mathbf{b} = \mathbf{u},$$

för alla kvadratiska matriser  $U \in \mathbb{R}^{(m+1) \times (m+1)}$  och vektorer  $\mathbf{u} \in \mathbb{R}^{m+1}$ . (Jämför skalära fallet där  $(ub^2)' = 2ub$  och  $(ub)' = u$ .) Detta ger

$$\nabla\Phi(\mathbf{b}) = \nabla(\mathbf{b}^T A^T A \mathbf{b} - 2\mathbf{b}^T A^T \mathbf{f} + \mathbf{f}^T \mathbf{f}) = 2A^T A \mathbf{b} - 2A^T \mathbf{f}.$$

Följdaktligen har vi

$$0 = \nabla\Phi(\mathbf{c}) = 2A^T A \mathbf{c} - 2A^T \mathbf{f},$$

vilket är samma sak som (10). Det återstår att visa att  $\mathbf{c}$  är ett minimum (och inte ett maximum eller en sadelpunkt, till exempel). Villkoret för detta är att Hessianen till  $\Phi$  är en positivt definit matris vid  $\mathbf{c}$ . I vårt fall är Hessianen precis  $2A^T A$  och vi behöver alltså visa att

$$\mathbf{x}^T A^T A \mathbf{x} > 0, \quad \text{för alla } \mathbf{x} \neq 0 \text{ i } \mathbb{R}^{m+1}. \quad (11)$$

Vi noterar nu att vårt tidigare antagande om att kolumnerna i  $A$  är linjärt oberoende betyder att  $A\mathbf{x} \neq 0$  när  $\mathbf{x} \neq 0$ . Därmed följer (11) från det faktum att  $\mathbf{x}^T A^T A \mathbf{x} = \|A\mathbf{x}\|_2^2$ . Att  $A^T A$  är positivt definit implicerar också att  $A^T A$  är ickesingulär och att normalekvationerna (10) har en lösning.

Minstakvadratproblem i MATLAB:

- `c=polyfit(x,f,m)` ger minstakvadratlösningen om  $m < n$  och vektorerna  $\mathbf{x}$  och  $\mathbf{f}$  har längd  $n+1$ .
- `A\b` ger minstakvadratlösningen om  $A$  är en  $n \times m$ -matris,  $\mathbf{b}$  en  $n$ -vektor och  $m < n$ .

Se MATLABS `help`-kommando för mer info.

### 3 Interpolation med andra basfunktioner

Vi betraktar ett allmännare fall när  $\tilde{f}(x)$  är en linjärkombination av godtyckliga basfunktioner, dvs när den kan skrivas som

$$\tilde{f}(x) = \sum_{k=0}^m c_k \psi_k(x),$$

med några funktioner  $\psi_k(x)$  och obekanta koefficienter  $c_k \in \mathbb{R}$ ,  $k = 0, \dots, m$ . I polynomfallet använder vi till exempel  $\psi_k(x) = x^k$ . I allmänhet kan man använda välja andra funktioner som liknar  $f$ , tex trigonometriska funktioner  $\psi_k(x) = \exp(ikx)$  om  $f$  är periodisk. Interpolationsvilkoren blir då

$$\tilde{f}(x_j) = \sum_{k=0}^m c_k \psi_k(x_j) = f(x_j), \quad j = 0, \dots, n. \quad (12)$$

Liksom tidigare kan det skrivas som ett linjärt ekvationsystem för  $c_k$ ,

$$A\mathbf{c} = \mathbf{f}, \quad (13)$$

där

$$\mathbf{c} = \begin{pmatrix} c_0 \\ \vdots \\ c_m \end{pmatrix}, \quad \mathbf{f} = \begin{pmatrix} f(x_0) \\ \vdots \\ f(x_n) \end{pmatrix}, \quad A = \begin{pmatrix} \psi_0(x_0) & \psi_1(x_0) & \cdots & \psi_m(x_0) \\ \psi_0(x_1) & \psi_1(x_1) & \cdots & \psi_m(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \psi_0(x_n) & \psi_1(x_n) & \cdots & \psi_m(x_n) \end{pmatrix}.$$

Vi antar att basfunktionerna  $\psi_k(x)$  och noderna  $\{x_i\}$  är valda så att kolumnerna i  $A$  är linjärt oberoende. När  $n = m$  är då  $A$  ickesingulär och vi kan hitta  $\mathbf{c}$  som uppfyller (13) och alltså  $c_k$  som uppfyller (12). När antalet punkter är större än antalet obekanta, dvs då  $n > m$ , kan vi lösa systemet i minstakvadratmening som ovan, vilket ger det  $\tilde{f}$  som minimerar

$$\sum_{j=0}^n (\tilde{f}(x_j) - f(x_j))^2.$$