Linjär algebra och numerisk analys - Heathsammanfattning av Philip Krantz

Anteckningar från de aktuella avsnitten i Heath

1.2.2 Abs, Rel fel

Absolut fel = approx. värde − exakt värde

Relativt fel = $\frac{\text{Abs. fel}}{\text{exakt värde}}$ (Rel. fel odef. då exakt värde = 0)

Ett användbart sätt att beskriva förhållandet mellan abs- och rel fel:

approx. värde = (exakt värde) · (1 + rel. fel)

1.2.4

Trunkeringsfel: Differensen mellan exakt lösning av ett problem och den lösning som fås ur en given algoritm. Detta uppstår bl.a när en iterativ metod avslutas före konvergens.

Avrundningsfel: Differensen mellan resultatet efter avslutad algoritm och den lösning som hade fåtts om iterationerna nått konvergens.

Det sammanlagda beräkningsfelet är summan av dessa.

1.2.5 Framåt, Bakåt fel

Framåtfel: Vi vill beräkna värdet av en funktion $y = f(x)$

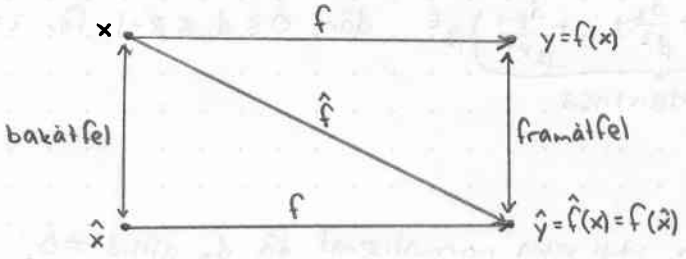
$$\Delta y = \hat{y} - y$$

där $\hat{y}$ är ett approximativt värde.

Bakåt fel:

$$\Delta x = \hat{x} - x \quad \text{då} \quad f(\hat{x}) = \hat{y}$$

Betrakta schemat nedan



1.2.6 **Konditionstal och känslighet hos en matris**

$$\text{Konditionstal} = \frac{|\Delta y/y|}{|\Delta x/x|} \quad \text{dvs} \quad \frac{|(f(\hat{x}) - f(x))/f(x)|}{|(\hat{x}-x)/x|}$$

Vi har alltså att

$$|\text{Rel. framåtfel}| = \text{konditionstal} \cdot |\text{Rel. bakåtfel}|$$

Konditionstalet defineras även som

$$\text{Konditionstal} \approx \left| \frac{x f'(x)}{f(x)} \right|$$

Känslighet

$$\frac{|\Delta y|}{|\Delta x|}$$

1.2.7 **Stabilitet och exakta lösningar**

> Def. En stabil algoritm genererar en exakt lösning för ett problem närliggande det ursprungliga.

1.3.1 **Flyttalsnummer**

Ett flyttals-system $\mathbb{F}$ karaktäriseras av följande fyra nummer:

β bas

p precision

$[L, U]$ exponentintervall

Varje flyttalsnummer $x \in \mathbb{F}$ har formen

$$x = \pm \underbrace{\left(d_0 + \frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \ldots + \frac{d_{p-1}}{\beta^{p-1}}\right)}_{\text{Mantissa}} \beta^E \quad \text{där } 0 \le d_i \le \beta - 1 \text{ för } i = 0, \ldots, p-1$$

$L \le E \le U$

1.3.2 **Normalisering**

Ett flyttalssystem sägs vara normaliserat då d_0 alltid $\neq 0$, såvida inte $d_0 = 0$ från början.

Mantissan m uppfyller då villkoret:

$$1 \le m < \beta$$

1.3.3 **Egenskaper hos flyttalssystem**

Antalet normaliserade flyttal

$$2(\beta-1)\beta^{p-1}(U-L+1)+1$$

↑ "leading digit" i mantissa

Underflow level = β^L

Overflow level = $\beta^{U+1}(1-\beta^{-p})$

1.3.9 **Kansellation**

Def. Kansellation uppstår då två tal, båda med p st siffror, samma tecken och liknande storlek, subraheras. Detta leder till ett svar med färre gällande siffror än p, vilket tar bort resultatet.

2. Linjära ekvationssystem

2.2 En $n \times n$ matris A sägs vara icke-singulär om den uppfyller någon av nedanstående ekvivalenta punkter

- A har en invers A^{-1} $(AA^{-1} = A^{-1}A = I)$
- $\det(A) \neq 0$
- $\text{rank}(A) = n$
- $Az \neq 0$ för någon godt. vektor $z \neq 0$

Om någon (således alla) dessa punkter gäller, har systemet $Ax = b$ den unika lösningen $x = A^{-1}b$ oavsett b.

2.3.1 Vektornormer

$$\|x\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \quad \text{där } p > 0$$

Vi har tre viktigaste normerna

1-norm

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|$$

2-norm

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2}$$

∞-norm

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$$

2.3.2 Matrisnormer

$$\|A\|_1 = \max_j \sum_{i=1}^{m} |a_{ij}| \qquad \left\| \quad \right|$$

$$\|A\|_\infty = \max_i \sum_{j=1}^{n} |a_{ij}| \qquad \left| \overline{\quad} \right|$$

Matrisnormerna uppfyller följande egenskaper där $\mathbb{A}$ och $\mathbb{B}$ är godtyckliga matriser.

1. $\|\mathbb{A}\| > 0$ om $\mathbb{A} \neq \mathbb{0}$
2. $\|\gamma \mathbb{A}\| = |\gamma| \cdot \|\mathbb{A}\|$ för någon skalär γ
3. $\|\mathbb{A} + \mathbb{B}\| \leq \|\mathbb{A}\| + \|\mathbb{B}\|$
4. $\|\mathbb{A}\mathbb{B}\| \leq \|\mathbb{A}\| \cdot \|\mathbb{B}\|$
5. $\|\mathbb{A}x\| \leq \|\mathbb{A}\| \cdot \|x\|$ för någon vektor x

2.3.3 Konditionstal för en matris $\mathbb{A}$

$$\text{cond}(\mathbb{A}) = \|\mathbb{A}\| \cdot \|\mathbb{A}^{-1}\| = \left(\max_{x \neq 0} \frac{\|\mathbb{A}x\|}{\|x\|}\right) \cdot \left(\max_{x \neq 0} \frac{\|\mathbb{A}x\|}{\|x\|}\right)^{-1}$$

Konditionstalet för en matris mäter hur nära matrisen är att vara singulär. Ju högre cond $\Rightarrow$ desto närmre singulär.

2.3.4 Felmarginal

Låt x vara lösningen till $Ax = b$, icke singulärt system. Låt $\hat{x}$ vara lösningen till $A\hat{x} = b + \Delta b$ med ett störd högerled.

Vi def. $\Delta x = \hat{x} - x$

Efter tillämpning av räkneregler för matrisnormer följer att

$$\frac{\|\Delta x\|}{\|x\|} \leq \text{cond}(\mathbb{A}) \frac{\|\Delta b\|}{\|b\|}$$

Liknande samband fås då man stör matrisen $\mathbb{A}$ med en matris $\mathbb{E}$.

$$\frac{\|\Delta x\|}{\|\hat{x}\|} \leq \text{cond}(\mathbb{A}) \frac{\|\mathbb{E}\|}{\|\mathbb{A}\|}$$

2.3.5 Residual

Residualen för en approximerad lösning $\hat{x}$ till ett linj. system $Ax=b$ är följande differens

$$r = b - A\hat{x}$$

Vi får följande samband mellan residualen och konditionen för ett system:

$$\frac{\|\Delta x\|}{\|\hat{x}\|} \leq \text{cond}(A) \frac{\|r\|}{\|A\| \cdot \|\hat{x}\|}$$

2.4.1 Lösningsmetoder för linj. ekv. syst.

Transformation

För att lösa ett system $Ax=b$ kan vi istället lösa ett system med identisk lösning, men som är enklare att beräkna.

Detta görs genom att multiplicera båda leden med M:

$$MAz = Mb$$

$$z = (MA)^{-1}Mb = A^{-1}\overbrace{M^{-1}M}^{I}b = A^{-1}b$$

Permutation

Detta görs genom att multiplicera båda leden i $Ax=b$ med en permutationsmatris P, innehållande t.ex $\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$. Detta byter plats på raderna i A.

Lösningen blir då $x = Pz$

2.4.4 LU-faktorisering

För att enklare lösa ett system där man senare vill använda olika vektorer b i högerledet LU-faktoriserar vi matrisen A.

$$A = LU = \begin{bmatrix} 1 & & 0 \\ * & \ddots & \\ * & * & 1 \end{bmatrix} \begin{bmatrix} * & * & * \\ & \ddots & * \\ 0 & & * \end{bmatrix}$$

Lös sedan följande system:

$$\begin{cases} Ly = b \\ Ux = y \end{cases}$$

2.5. Speciella typer av linjära system

Symmetriskt: $A = A^T$

Pos. definit: $x^T A x > 0 \quad \forall x \neq 0$

"Banded": $a_{ij} = 0 \ \forall \ |i-j| > \beta$ där β är "bandwidth" för A

Ett specialfall är den tridiagonala matrisen för vilken $\beta = 1$

"Sparse": De flesta element i A är 0.

2.5.1 Symmetriska pos. definita system

Om A symmetrisk, pos def matris så kan LU-faktoriseringen ordnas så att $U = L^T$, $A = LL^T$ för den vanliga def. av L.

Detta kallas Cholesky faktorisering.

3.1 Linjära minsta-kvadrat problem

Vårt mål är att lösa systemet $Ax=b$ approximativt genom att minimera residualvektorn $r=b-Ax$, tillika skillnaden mellan höger och vänsterledet. Det handlar givetvis om ett överbestämt ekv. system.

3.2 Unika lösningar

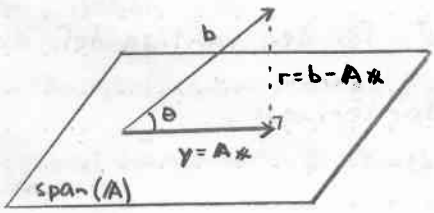
$A\hat{x} \cong b$ har en unik lösning om och endast om A har full kolonnrang. dvs $\text{rang}(A)=n$

3.2.1 Normal ekvationer

För att finna en lösning till vårt minsta-kvadrat problem, sätter vi upp följande normal ekvationer

$$A^T A\hat{x} = A^T b$$

3.2.2 Ortogonalitet och ortogonal projektion



3.3 Känslighet och konditionstal hos en matris $\mathbb{A}$

Vi har en icke kvadratisk matris $\mathbb{A}_{(m \times n)}$. Eftersom $\mathbb{A}$ ej är inverterbar kan vi inte tillämpa den formel för konditionstal som vi tidigare definierat.

Vi definierar därför en pseudoinvers, som gör det möjligt att invertera $A \in (m \times n)$ $m \neq n$.

Denna pseudoinvers definieras som

$$\mathbb{A}^{+} = (\mathbb{A}^T \mathbb{A})^{-1} \mathbb{A}^T$$

Vi ser då att

$$\mathbb{A}^{+} \mathbb{A} = I$$

Lösningen till ett minsta kvadrat problem $\mathbb{A}x = b$ ges nu till

$$x = \mathbb{A}^{+} b$$

Konditionstal för $\mathbb{A}_{(m \times n)}$ med $\text{rang}(\mathbb{A}) = n$

$\text{cond}(\mathbb{A}) = \|\mathbb{A}\|_2 \cdot \|\mathbb{A}^{+}\|_2$ där $\text{cond}(\mathbb{A}) = \infty$ då $\text{rank}(\mathbb{A}) < n$

P.s.s. som att $\text{cond}(\mathbb{A})$ för $\mathbb{A}_{(n \times n)}$ mäter hur nära $\mathbb{A}$ är singulär, mäter $\text{cond}(\mathbb{A})$ för $\mathbb{A}_{(m \times n)}$ hur nära $\mathbb{A}$ är otillräcklig i rang.

3.4.1 Problem transformationer - Normal ekvationer

Då vi har matrisen $\mathbb{A}$ av full rang dvs $n \times n$, så löser vi ekvationen

$$\mathbb{A}^T \mathbb{A} x = \mathbb{A}^T b$$

Om vi istället har $\mathbb{A}_{(m \times n)}$, använder vi Cholesky faktorisering

$\mathbb{A}^T \mathbb{A} = \mathbb{L} \mathbb{L}^T$ Sedan löser vi ekv. syst.

$$\begin{cases} \mathbb{L} y = \mathbb{A}^T b \\ \mathbb{L}^T x = y \end{cases}$$

Rektangulär $\longrightarrow$ Kvadratisk $\longrightarrow$ Triangulär

3.4.2 Förstärkta system

Definitionen av residualvektor $\mathbb{r}$ + kravet på att $\mathbb{r}$ skall vara ortogonal mot kolonnerna i $\mathbb{A}$ ger följande system.

$$\begin{cases} \mathbb{r} + \mathbb{A}\mathbb{x} = \mathbb{b} \\ \mathbb{A}^T\mathbb{r} = \mathbb{0} \end{cases}$$

Detta skrivs nu som ett förstärkt system

$$\begin{bmatrix} I\alpha & \mathbb{A} \\ \mathbb{A}^T & \mathbb{0} \end{bmatrix} \begin{bmatrix} \frac{\mathbb{r}}{\alpha} \\ \mathbb{x} \end{bmatrix} = \begin{bmatrix} \mathbb{b} \\ \mathbb{0} \end{bmatrix}$$

Där α används för att kontrollera relationen mellan elementen i de två undersystemen. En god idé är att ta

$$\alpha = \max_{i,j} |a_{ij}| / 1000$$

3.6 **Singulärvärdesuppdelning, SVD**

Singulärvärdesuppdelningen av en (mxn) matris A har formen

$$A = U\Sigma V^T$$

där U: mxm ortogonal matris. Kolonnerna u_i vänster sing.ve

V: nxn ortogonal matris. Kolonnerna v_i höger sing. vel

Σ: mxn diagonal matris med de singulära värdena

σ_i i diagonalen. Med

$$\sigma_{ij} = \begin{cases} 0 & \text{för } i \neq j \\ \sigma_i \geq 0 & \text{för } i = j \end{cases}$$

Om vi har en SVD: $A = U\Sigma V^T$ så att $A_{(3\times 2)}$

$$\underset{3\times 3}{\begin{bmatrix} & & \end{bmatrix}} \underset{3\times 2}{\begin{bmatrix} \sigma_1 & \\ & \sigma_2 \\ & \end{bmatrix}} \underset{2\times 2}{\begin{bmatrix} & \\ & \end{bmatrix}}$$

Rangen av matrisen A är lika med antalet $\sigma \neq 0$.

Vi kan få lite olika form på vår SVD:

A är en mxn matris av rang $(A) = n \Rightarrow$

$$A = U\Sigma V^T = [U_1\ U_2]\begin{bmatrix}\Sigma \\ 0\end{bmatrix}V^T = U_1\Sigma_1 V^T$$

där U_1 (mxn) och Σ_1 (nxn) icke-singulär, utgör den reducerade SVD av A.

Lösningen till SVD av A ges av:

$$x = V\Sigma_1^{-1}U_1^T b$$

SVD är mycket användbart för illa konditionerade problem.

3.6.1 Tillämpningar av SVD

Euklidisk matrisnorm:

$$\|A\|_2 = \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2} = \sigma_{max}$$

Euklidiskt konditionstal:

Konditionstalet för en godtycklig matris A ges av:

$$\text{cond}(A) = \frac{\sigma_{max}}{\sigma_{min}}$$

Ett alternativt sätt att skriva SVD är:

$$A = U \Sigma V^T = \sigma_1 E_1 + \sigma_2 E_2 + \ldots + \sigma_n E_n$$

där $E_i = u_i v_i^T$

4.1 **Egenvärden och egenvektorer**

Ett egenvärdesproblem består av en given (nxn) matris $\mathbb{A}$ som representerar en linjär transformation på ett n-dim vektorrum. Vi söker en vektor $\mathbb{x} \neq \mathbb{0}$ och en skalär så att:

$$\mathbb{A}\mathbb{x}_i = \lambda \mathbb{x}_i$$

där $\mathbb{x}_i$ är egenvektor och λ tillhörande egenvärde. Högeregenve

En vänsteregenvektor av $\mathbb{A}$ är en högeregenvektor av $\mathbb{A}^T$ d

$$\mathbb{x}_i^T \mathbb{A} = \lambda \mathbb{x}_i^T$$

Hela uppsättningen av egenvärden $\lambda_1, \ldots, \lambda_n$ till $\mathbb{A}$, $\lambda(\mathbb{A})$ kallas för A:s spektrum.

4.2.1 **Karakteristiska polynomet**

A:s egenvärden fås ur determinanten

$$\det(\mathbb{A} - \lambda \mathbb{I}) = 0$$

A:s egenvektorer till respektive λ fås ur

$$(\mathbb{A} - \lambda \mathbb{I})\mathbb{x} = 0$$

En (nxn) matris har alltid n:st λ.

4.2.2 **Multiplicitet och diagonaliserbarhet**

Multiplicitet kan tolkas:

algebraiskt: en rot till en karakteristisk ekvation.

geometriskt: Antalet egenvektorer $\binom{\text{linj.}}{\text{obe.}}$ som tillhör samma λ.

Diagonaliserbarhet:

Om $\mathbb{A}^{(n\times n)}$ har egenvärdena $\lambda_1, \ldots, \lambda_n$ och tillhörande egenvektorer $\mathbb{P} = (x_1, \ldots, x_n)$ och multipliciteten hos varje $\lambda = 1$ så är $\mathbb{A}$ diag. bar och kan skrivas som $\mathbb{D} = \mathbb{P}^{-1}\mathbb{A}\mathbb{P}$

Där $\mathbb{D}$ är $\begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_n \end{bmatrix}$ och $\mathbb{P} = [\mathbb{x}_1, \ldots, \mathbb{x}_n]$

4.2.3 Egenrum och icke konstanta underrum

En egenvektor till egenvärdet λ är inte unik ty om vi multiplicerar egenvektorn $\mathbf{x}$ med γ fås

$$\mathbb{A}(\gamma\mathbf{x}) = \lambda(\gamma\mathbf{x})$$

4.2.4 Egenskaper hos matriser och egenvärdesproblem

Egenskaper hos en nxn matris $\mathbb{A}$:

- Diagonal: $a_{ij} = 0$ för $i \neq j$

$$\mathbb{A} = \begin{bmatrix} a_{11} & 0 & 0 & 0 \\ 0 & a_{22} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & a_{nn} \end{bmatrix}$$

- Tridiagonal: $a_{ij} = 0$ för $|i-j| > 1$

$$\mathbb{A} = \begin{bmatrix} a_{11} & a_{12} & & \\ a_{21} & a_{22} & \ddots & \\ & \ddots & \ddots & \\ & & & \end{bmatrix}$$

- Triangulär: $a_{ij} = 0$ för $i > j$ (över triangulär)
  $i < j$ (nedre triangulär)

$$\mathbb{A} = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ 0 & a_{22} & \vdots \\ \vdots & & \ddots \\ 0 & \cdots & 0 \; a_{nn} \end{bmatrix} \qquad \mathbb{A} = \begin{bmatrix} a_{11} & & 0 \\ \vdots & a_{22} & \\ & & \ddots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}$$

- Ortogonal: $\mathbb{A}^T\mathbb{A} = \mathbb{A}\mathbb{A}^T = \mathbb{I}$
- Symmetrisk: $\mathbb{A}^T = \mathbb{A}$

4.5 **Beräkning av egenvärden och egenvektorer**

4.5.3 **Rayleigh kvot iteration**

Låt x vara en approximativ egenvektor till A. För att då bestämma det bästa motsvarande egenvärde λ kan vi lösa $(n \times 1)$ minsta-kvadrat problemet:

$$x\lambda \cong Ax$$

Från normalekvationerna $x^T x\lambda \cong x^T Ax$ fås

$$\lambda = \frac{x^T Ax}{x^T x}$$

Vilket kallas Rayleigh kvot. Kan användas för att accelerera konvergensen för en metod.

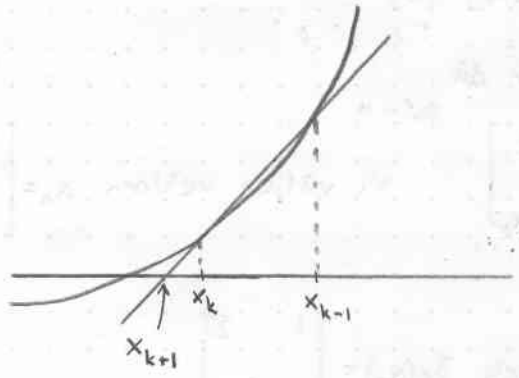
5.5.4 **Sekant metoden** - Löser ickelinjära problem

Sekant metoden ger en approximation som bygger på differenser av tidigare iterationer:

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

Metoden kräver alltså två startvärden x_0, x_1 som antingen tas fram ur en annan numerisk metod eller uppskattas.

Fig.



5.6 Ickelinjära ekvationssystem

5.6.1 Fix-punkts iteration

För ett problem där $g_j: \mathbb{R}^n \to \mathbb{R}^n$ går ett fixpunkt-problem till g_j ut på att finna en vektor $x \in \mathbb{R}^n$ så att

$$x = g_j(x)$$

Motsvarande fixpunktsiteration är alltså

$$x_{k+1} = g_j(x_k) \quad \text{för en given startvektor } x_0$$

5.6.2 Newtons metod

Newtons metod är den mest populära och effektiva metoden för ickelinjära ekv system i n-dim. Denna metod bygger på de trunkerade Taylor serierna

$$f(x+s) \approx f(x) + J_f(x)\, s$$

där $J_f(x)$ är Jacobian matrisen för f $\{J_f(x)\}_{ij} = \dfrac{\partial f_i(x)}{\partial x_j}$

Om s uppfyller det linjära systemet:

$$J_f(x)s = -f(x) \Rightarrow x + s \approx \text{Nollställe till } f.$$

Ex

$$f(x) = \begin{bmatrix} x_1 + 2x_2 - 2 \\ x_1^2 + 4x_2^2 - 4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Jacobianen blir då

$$J_f(x) = \begin{bmatrix} 1 & 2 \\ 2x_1 & 4x_2 \end{bmatrix} \qquad \text{Vi väljer vektorn } x_0 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

Vi får alltså

$$f(x_0) = \begin{vmatrix} 3 \\ 13 \end{vmatrix} \quad \text{och} \quad J_f(x_0) = \begin{vmatrix} 1 & 2 \\ 2 & 8 \end{vmatrix}$$

Vi löser nu bara ut s_0 ur

$$J_f(x_0)\, s_0 = -f(x_0): \quad \begin{vmatrix} 1 & 2 \\ 2 & 8 \end{vmatrix} s_0 = \begin{vmatrix} -3 \\ -13 \end{vmatrix}$$

Sedan har vi

$$x_1 = x_0 + s_0$$

5.6.3 Sekant-uppdaterings metoder

Broydens metod: $B_{k+1}(x_{k+1} - x_k) = f(x_{k+1}) - f(x_k)$

Ex (samma som föregående)

$$f(x) = \begin{bmatrix} x_1 + 2x_2 - 2 \\ x_1^2 + 4x_2^2 - 4 \end{bmatrix} = \begin{vmatrix} 0 \\ 0 \end{vmatrix} \quad \text{Vi låter även} \quad x_0 = \begin{vmatrix} 1 \\ 2 \end{vmatrix} \Rightarrow f(x_0) = \begin{vmatrix} 3 \\ 13 \end{vmatrix}$$

Nu har vi att

$$B_0 = J_f(x_0) = \begin{vmatrix} 1 & 2 \\ 2 & 16 \end{vmatrix}$$

Detta leder till att vi nu får

$$B_0 s_0 = -f(x_0)$$

5.6.4 Dämpad Newtons metod

Att dämpa Newtons metod innebär en försiktighets åtgärd för ett n-dim problem. Denna åtgärd ser ut såhär:

$$x_{k+1} = x_k + \alpha_k s_k$$

där α_k är en vald skalär, såatt x_{k+1} är en bättre approximation till lösningen än x_k.

6. Optimering

6.1 Optimeringsproblem

Ett allmänt kontinuerligt optimeringsproblem har formen

$$\min_{x} f(x) \quad \text{under villkoren} \quad g_j(x) = 0$$
$$h_i(x) \leq 0$$

där $f: \mathbb{R}^n \to \mathbb{R}$

$g: \mathbb{R}^n \to \mathbb{R}^m$

$h: \mathbb{R}^n \to \mathbb{R}^p$

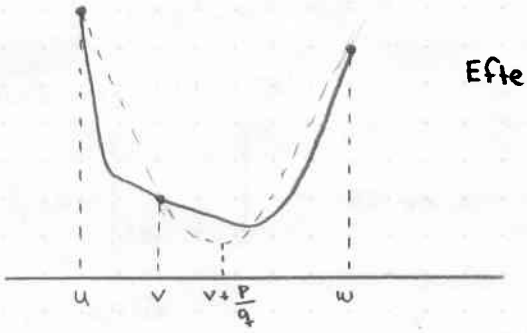
Optimeringsproblemet kategoriseras efter hur f, g_j och h_i förhåller sig till varandra.

6.4.2 Parabel interpolisering

Givet tre punkter u, v, w med tillhörande funktionsvärden f_u, f_v, f_w där v är den bäst approximerade minimipunkten än så länge, ges minimum av parabelinpoliseringen av de tre punkterna av: $v + \frac{p}{q}$

där $p = \pm\left((v-u)^2(f_v - f_w) - (v-w)^2(f_v - f_u)\right)$

$q = \mp 2\left((v-u)(f_v - f_w) - (v-w)(f_v - f_u)\right)$



Efter första iterationen

6.4.3 **Newtons metod**

$$f(x+h) \approx f(x) + f'(x)h + \tfrac{1}{2} f''(x)h^2 \qquad \text{(Taylor)}$$

där $h = -f'(x)/f''(x)$

Newtons metod konvergerar kvadratiskt.

6.5.3 **Newtons metod i flera variabler**

Taylor ger nu

$$f(* + s) \approx f(*) + \nabla f(*)^T s + \tfrac{1}{2} s^T \mathbb{H}_f(*)\, s$$

där $\mathbb{H}_f(*)$ står för Hessians matris bestående av partiella andraderivator av f: $\{\mathbb{H}_f(*)\}_{ij} = \dfrac{\partial^2 f(*)}{\partial x_i \partial x_j}$

Denna kvadratiska form av s är minimal

då

$$\mathbb{H}_f(*)\, s = -\nabla f(*)$$

6.5.4 **Kvasi-Newtons metod** (kvasi = nästan)

$$*_{k+1} = *_k - \alpha_k \mathbb{B}_k^{-1} \nabla f(*_k) \qquad \text{Undviker Jacobian}$$

där $\mathbb{B}_k$ är en approximation av $J(*_k)$

6.6 Icke-linjära minsta kvadrat problem

Man kan betrakta detta som ett optimeringsproblem, där vi givet punkterna (t_i, y_i) $i = 1, \ldots, m$ söker vektorn $x \in \mathbb{R}^n$ som ger den bästa approximationen till punkterna.

Vi definierar komponenterna till residualfunktionen:

$$r : \mathbb{R}^n \to \mathbb{R}^m$$

$$r_i(x) = y_i - f(t_i, x) \quad i = 1, \ldots, m$$

För att få residualen minimal vill vi nu minimera funktionen

$$\Phi(x) = \frac{1}{2} r(x)^T r(x)$$

Gradientvektorn och Hessianen för Φ ges av

$$\nabla \Phi(x) = J^T(x) r(x)$$

$$H_\Phi(x) = J^T(x) J(x) + \sum_{i=1}^{m} r_i(x) H_{r_i}(x)$$

Detta ger likt tidigare att om vi applicerar Newtons metod får vi:

$$H_\Phi(x_k) s_k = - \nabla \Phi(x_k)$$

6.6.1 Gauss-Newtons metod

Observera att i H_Φ multipliceras varje Hessianmatris H_r med motsvarande residual komponent r_i. Denna komponent bör vara relativt liten för en relativt god approximation.

Detta leder till Gauss-Newtons metod, där vi bortser från H_r, och får approximationen

$$(J^T(x_k) J(x_k)) s_k = - J^T(x_k) r(x_k)$$

Anm: Detta känns igen som normalekvationerna till ett mxn linjärt minsta-kvadrat problem

$$J(x_k) s_k \cong - r(x_k)$$ sedan blir nästa approx.

$$x_{k+1} = x_k + s_k$$

Ex Vi vill finna en approximation till den icke linjära funktio

$$f(t, x) = x_1 e^{x_2 t}$$

För datapunkterna

t	0.0	1.0	2.0	3.0
y	2.0	0.7	0.3	0.1

Först söker vi Jacobianen för residualfunktionen r.

$$\{J(x)\}_{i,1} = \frac{\partial r_i(x)}{\partial x_1} = -e^{x_2 t_i} \quad \{J(x)\}_{i,2} = \frac{\partial r_i(x)}{\partial x_2} = -x_1 t_i e^{x_2 t_i}$$

där $i = 1, \ldots, 4$

Vi tar $x_0 = [1 \ 0]^T$ som startpunkt

Då erhålls följande minsta kvadrat lösning

$$J(x_0)\, s_0 = \begin{vmatrix} -1 & 0 \\ -1 & -1 \\ -1 & -2 \\ -1 & -3 \end{vmatrix} s_0 \cong \begin{vmatrix} -1 \\ 0.3 \\ 0.7 \\ 0.9 \end{vmatrix} = -r(x_0)$$

Detta ger $s_0 = [0.69 \ -0.61]^T$ som i sin tur leder fram till $x_1 = x_0 + s_0 = [1 \ 0]^T + [0.69 \ -0.61]^T = [1.69 \ -0.61]^T$

Detta förlopp fortlöper nu tills konvergens nåtts.

Anm: Även denna metod, baserad på Newtons metod, kan misslyckas att konvergera då man startar för långt ifrån ett nollställe.

6.7.3 Linjära optimerings problem med tvångsvillkor

Detta innebär att både funktionen och villkoren måste vara linjära. En av de standardforme för ett sådant problem är

$$\min_{\mathbf{x}} f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \quad \text{under villkoren} \quad \begin{cases} \mathbf{A}\mathbf{x} = \mathbf{b} \\ \mathbf{x} \geq \mathbf{0} \end{cases}$$

där $m<n$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, $\mathbf{c}, \mathbf{x} \in \mathbb{R}^n$

Problemet löses med hjälp av "simplex" metod.

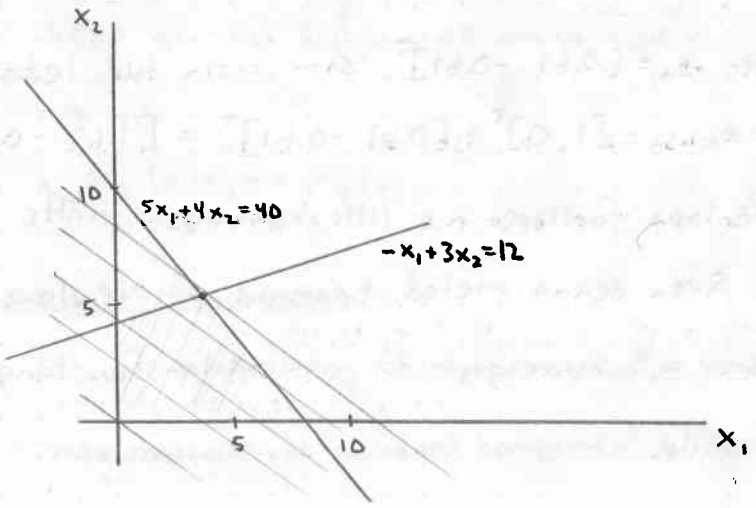
Ex Lös optimeringsproblemet

$$\min_{\mathbf{x}} f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} = -8x_1 - 11x_2$$

under bivillkoren

$$\begin{cases} 5x_1 + 4x_2 \leq 40 \\ -x_1 + 3x_2 \leq 12 \end{cases}$$

$$x_1 \geq 0, \quad x_2 \geq 0$$



7. **Interpolation** - Anpassa en funktion till givna data så att funktionen har exakt samma värde som den givna datan.

7.1 **Interpolation** - En grundsten för numerisk analys

Allmänt gäller att interpolationsproblem i en-dim har följande form:

För given data

(t_i, y_i) där $i=1,\dots,m$ med $t_1 < t_2 < \dots < t_m$

söker vi en funktion $f: \mathbb{R} \to \mathbb{R}$ så att

$f(t_i) = y_i$ för $i=1,\dots,m$

Vi talar om att f är interpolant för den givna datan.

Användning av interpolation:

1) Approximation mellan punkter.
2) Mjuka kurvor mellan punkter.
3) Approximera besvärliga funktioner med enklare.
4) Grund för metoder för integraler och derivator.
5) Polynomapproximation vid optimering.

7.2 **Existens, unika lösningar och konditionering**

En interpolationsfunktion uttrycks som ett linjärt ekvationssystem och har därför samma egenskaper avseende existens och unika lösningar som ett linj. ekv. sysl. Detta innebär att om antalet obekanta parametrar > antalet ekvationer, saknar interpolationen lösning, på motsvarande sätt är lösningen inte unik om systemet är överbestämt.

För att ta fram en interpolant, använder vi en uppsättning med basfunktioner $\phi_1(t), \ldots, \phi_n(t)$

Nu fås det linjära ekvationssystemet

$$f(t) = \sum_{j=1}^{n} x_j \phi_j(t)$$

där vi vill bestämma de okända parametrarna x_j.

Att nu kräva att f interpolerar data punkterna (t_i, y_i) innebär alltså att

$$f(t_i) = \sum_{j=1}^{n} x_j \phi_j(t_i) = y_i \qquad i = 1, \ldots, m$$

Detta innebär ett system av formen $\mathbb{A}x = b$ att lösa.

Lämpligt är att välja antalet basfunktioner n till att vara lika med antalet data punkter m för att få ett kvadratiskt system som kan, om $\mathbb{A}$ är icke singulär, lösas exakt.

7.3 Polynom interpolisering - Den vanligaste interpoliseringen

För ett givet heltal $k \geq 0$ betecknar vi $\mathbb{P}_k$ som uppsättningen av alla polynom av grad $\leq k$ definierat på ett givet intervall.

Valet av bas att använda i polynomen avgör mycket hur kostsam och långa uträkningarna blir.

7.3.1 Monomial bas

För att interpolera n st punkter väljer vi $k = n-1$ för att kommande system skall få en unik lösning. Den mest naturliga basen att tillämpa för $\mathbb{P}_{n-1}$ är uppbyggd av de n första monomialerna

$$\phi_j(t) = t^{j-1} \quad \text{för } j = 1, \ldots, n$$

dvs vi får

$$p_{n-1}(t) = x_1 + x_2 t + x_3 t^2 + \ldots + x_n t^{n-1}$$

Detta ger oss följande system att lösa: Punkter: (t_i, y_i) $i=1,\dots,n$

$$\mathbb{A}x = \begin{vmatrix} 1 & t_1 & \dots & t_1^{n-1} \\ 1 & t_2 & \dots & t_2^{n-1} \\ \vdots & & & \\ 1 & t_n & \dots & t_n^{n-1} \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{vmatrix} = \begin{vmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{vmatrix} = y$$

Denna matris går även under benämningen "Vandermonde matris"

Att använda denna monomialbas innebär dock en risk man behöver känna till. För stora värden på n blir baselementen mer och mer parallella, vilket medför att Vandermondes matris blir illa konditionerad.

7.3.3 Newton interpolation

För en given uppsättning datapunkter (t_i, y_i) $i=1,\dots,n$ ges Newtons bas funktioner för $\mathbb{P}_{n-1}$ av

$$\pi_j(t) = \prod_{k=1}^{j-1} (t - t_k) \quad j=1,\dots,n$$

Med Newtons bas får en given polynombas formen

$$p_{n-1}(t) = x_1 + x_2(t-t_1) + x_3(t-t_1)(t-t_2) + \dots + x_n(t-t_1)(t-t_2)\dots(t-t_n$$

Vi ser utifrån definitionen att

$$\pi_j(t_i) = 0 \quad \text{för } i < j$$

Detta ger det positiva med basen, nämligen att $\mathbb{A}$ med element $a_{ij} = \pi_j(t_i)$, blir undertriangulär.

På nästa sida följer ett ex. på Newton interpolisering.

Ex Vi har datapunkterna

t	-2	0	1
y	-27	-1	0

Med Newtons bas fås följande undertriangulära system

$$\begin{vmatrix} 1 & 0 & 0 \\ 1 & t_2-t_1 & 0 \\ 1 & t_3-t_1 & (t_3-t_1)(t_3-t_2) \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \end{vmatrix} = \begin{vmatrix} y_1 \\ y_2 \\ y_3 \end{vmatrix}$$

Alltså med datapunkterna

$$\begin{vmatrix} 1 & 0 & 0 \\ 1 & 2 & 0 \\ 1 & 3 & 3 \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \end{vmatrix} = \begin{vmatrix} -27 \\ -1 \\ 0 \end{vmatrix}$$

Löses detta får vi $x = [-27 \quad 13 \quad -4]^T$

Vilket alltså innebär att det interpoliserings polynom vi söker är

$$p(t) = -27 + 13(t+2) - 4(t+2)t$$

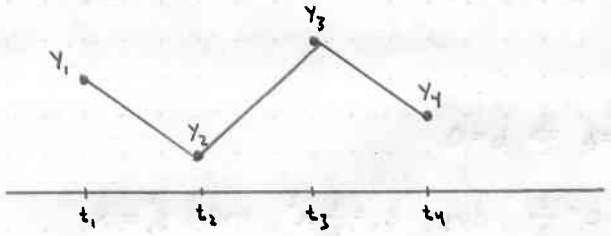
7.4 Styckvis polynom interpolering

För att minska svårigheterna att ta fram interpolanter av relativt låg grad, kan man dela upp sitt intervall som man vill interpolera i flera delintervall. Enligt följande

Givet en uppsättning datapunkter (t_i, y_i) $i = 1, \ldots, n$ med $t_1 < t_2 < \ldots < t_n$ använder man olika interpolerings funktioner för varje delintervall. $[t_i, t_{i+1}]$

Punkterna för vilka vi byter delintervall kallar vi brytpunkter.

Man kan själv välja vilket gradtal man vill ha på de "splines" man får mellan punkterna. Den enklaste är bara räta linjer mellan punkterna.



Ju högre grad man väljer på dessa splines, desto mjukare övergång får man till nästa spline.

7.4.2 Kubiska splines interpolisering

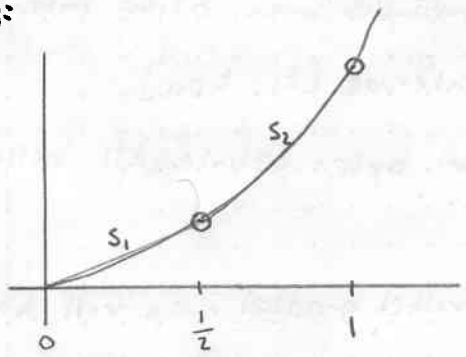
En spline är ett styckvis polynom av grad k som är kontinuerligt differentierbar k-1 gånger. Detta innebär alltså att kubiska splines är differentierbara 2 ggr. Att bestämma splines till ett givet antal mätdata går ut på att använda samma derivator och (för kubiska splines) även andraderivator i övergångarna mellan olika splines. Genom att sätta andraderivatan till 0 i ändpunkterna fås en naturlig spline. Se exemplet på nästa sida.

Ex Kvadratiska splines

Bestäm kvadratiska splines som interpolerar $f(x)=x^3$ i punkterna $0, \frac{1}{2}, 1$ som uppfyller $s'(0)=0$

Ledning: Ansätt på ett smart sätt (Jämför Newtons form vid interpolering)

Lösning:



Ansats:

$s_1 = a + bx + cx^2$ $\quad s_1(0)=0$ ger $a=0$

$s_1' = b + 2cx$

$s'(0)=0$ ger $s_1'(0)=b \Rightarrow b=0$

$s_1(\frac{1}{2}) = \frac{c}{4} = \frac{1}{8} \Rightarrow c=\frac{1}{2}$ dvs $s_1 = \frac{1}{2}x^2$ med $s_1' = x$

Nu ansätts s_2 smart

$s_2 = \frac{1}{8} + d(x-\frac{1}{2}) + e(x-\frac{1}{2})^2$

↑ Vi får ur s_1 att $s_2(\frac{1}{2}) = \frac{1}{8}$

$s_2' = d + 2e(x-\frac{1}{2})$

$s_2'(\frac{1}{2}) = d \underset{\text{splinevillkor}}{=} s_1'(\frac{1}{2}) = \frac{1}{2} \Rightarrow d=\frac{1}{2}$

$s_2(1) = \frac{1}{8} + \frac{1}{2}(\frac{1}{2}) + e(\frac{1}{2})^2 = 1 \Rightarrow e=\frac{5}{2}$

Svar: $s_1 = \frac{1}{2}x^2$ och $s_2 = \frac{1}{8} + \frac{1}{2}(x-\frac{1}{2}) + \frac{5}{2}(x-\frac{1}{2})^2$

8. Numerisk integrering och differentiering

8.3 Numerisk kvadratur $I = \int_a^b f(x)\,dx \approx \sum_{i=1}^{n} w_i f(x_i)$

Numerisk approximation av begränsade integraler kallas numerisk kvadratur.

För att approximera integraler använder vi ett viktat antal integrandvärden för ett ändligt antal mätpunkter inom integrationsintervallet. Integralen $I(f)$ är approximerad av n-punkts kvadraturs regel av formen

$$Q_n(f) = \sum_{i=1}^{n} w_i f(x_i) \qquad \text{på intervallet } [a, b]$$

där $a \leq x_1 < x_2 < \ldots < x_n \leq b$

Punkterna x_i i vilka integranden f är beräknad kallas noder och w_i kallas vikter.

Det finns en regel som säger att en kvadratur är:

öppen: Då $a < x_1$ och $b > x_n$

stängd: Då $a = x_1$ och $b = x_n$

Vår mål är att välja noder och vikter för att få en önskvärd nivå av noggrannhet för överkomlig beräkningsmängd.

8.3.1 Newton-Cotes kvadratur

Enklast är att välja en placering av noder så att avståndet mellan två noder är konstant över hela intervallet $[a, b]$. Detta gör Newton-Cotes kvadratur.

En n-punkters öppen Newton-Cotes regel har noderna

$$x_i = a + i(b-a)/(n+1) \quad i = 1, \ldots, n$$

och en stängd regel enligt Newton-Cotes:

$$x_i = a + (i-1)(b-a)/(n-1) \quad i = 1, \ldots, n$$

Detta ger oss tre mycket användbara regler

Mittpunktsregel: En punkts öppen Newton-Cotes regel

$$M(f) = (b-a) f\left(\frac{a+b}{2}\right)$$

Trapetsregeln: Två ändpunkter med poly. av grad 1 ger ett tvåpunkters stängd Newton-Cotes regel.

$$T(f) = \frac{b-a}{2}\left(f(a) + f(b)\right)$$

Simpsons regel: Två ändpunkter och mittpunkt med polynom av grad 2. Detta ger tre-punkters stängd Newton-Cotes regel.

$$S(f) = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)$$

8.6 Numerisk differentiering

För en given steglängd h, betraktar vi Taylor utvecklingen

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \frac{f'''(x)}{6}h^3 + \dots$$

och

$$f(x-h) = f(x) - f'(x)h + \frac{f''(x)}{2}h^2 - \frac{f'''(x)}{6}h^3 + \dots$$

Den första utvecklingen ger lösningen på $f'(x)$ genom framåt differens formeln

$$f'(x) = \frac{f(x+h)-f(x)}{h} - \frac{f''(x)}{2}h + \dots \approx \frac{f(x+h)-f(x)}{h}$$

Den andra utvecklingen ger på motsv. sätt lösningen genom bakåt differens formeln:

$$f'(x) = \frac{f(x)-f(x-h)}{h} + \frac{f''(x)}{2}h + \dots \approx \frac{f(x)-f(x-h)}{h}$$

För att erhålla en mer noggrann numerisk approximation av derivatan, använder vi centrerad differens:

$$f'(x) = \frac{f(x+h)-f(x-h)}{2h} - \frac{f'''(x)}{6}h^2 + \dots \approx \frac{f(x+h)-f(x-h)}{2h}$$

Vi kan även nu erhålla en approximation för $f''(x)$:

$$f''(x) = \frac{f(x+h)-2f(x)+f(x-h)}{h^2} - \frac{f^{(4)}(x)}{12}h^2 + \dots \approx$$

$$\approx \frac{f(x+h)-2f(x)+f(x-h)}{h^2}$$

8.7 Richardson extrapolation

Eftersom numerisk integrering och derivering alltid måste baseras på en specifik steglängd h, när vi i själva verket vill låta $h \to 0$, kom Richardson med en lösning.

Låt $F(h)$ betecknа värdet givet med steg längd h. Om vi beräknar ett värde på F för steg $h \neq 0$ så kan vi beräkna ett teoretiskt uppförande på F då $h \to 0$. Detta kallas för att <u>extrapolera</u> från ett givet värde för att få ett korrekt värde på $F(0)$.

Låt

$$F(h) = a_0 + a_1 h^p + \mathcal{O}(h^r) \quad \text{där } h \to 0 \text{ för ngt } r > p$$

a_0 och a_1 är okända.

<u>$F(0) = a_0$ är det vi söker.</u>

Vi beräknar nu F för två steglängder h och h/q där q positivt tal.

Vi har nu

$$F(h) = a_0 + a_1 h^p + \mathcal{O}(h^r)$$

och

$$F\left(\frac{h}{q}\right) = a_0 + a_1 \left(\frac{h}{q}\right)^p + \mathcal{O}(h^r)$$

Vi får alltså

$$\boxed{a_0 = F(h) + \frac{F(h) - F(h/q)}{q^{-p} - 1} + \mathcal{O}(h^r)}$$

Detta förfarande kan upprepas tills tillräckligt noggrann lösning erhållits.

9. Begynnelsevärdesproblem för ODE

9.1 ODE

En differentialekvation ålägger en relation mellan en okänd tillståndsfunktion $y(t)$ och en eller flera av y:s derivator med avseende på t.

9.3 Numeriska lösningar av ODEs

9.3.1 Euler's metod

Den enklaste metoden att göra en numerisk approximation till en ODE är Eulers metod där lösningen vid tiden

$$t_{k+1} = t_k + h_k$$

ges av

$$y_{k+1} = y_k + h_k f(t_k, y_k)$$

Denna metod härleds ur Taylorserien:

$$y(t+h) = y(t) + h y'(t) + \frac{1}{2} h^2 y''(t) + \ldots$$

genom att ta $t = t_k$, $h = h_k$, $y'(t_k) = f(t_k, y_k)$ och bortser från termer av ordning ≥ 2.

Ändlig differens approximation

Om vi byter ut $y'(t)$ i ODE $y' = f(t, y)$ mot första ordningens differensapproximation fås den algebraiska ekvationen

$$\frac{y_{k+1} - y_k}{h_k} = f(t_k, y_k)$$

Vilket innebär Eulers metod för lösningar av y_{k+1}

9.3.2 Noggrannhet och stabilitet

Några begrepp	Orsaker
Avrundningsfel	Flyttalsaritmetikens ändliga noggrannhet.
Trunkeringsfel	Vald metod bryter iterationerna.
Globalt fel $e_k = y_k - y(t_k)$	där y_k är den exakta lösningen och $y(t_k)$ lösningen av ODE genom initial punkten (t_0, y_0)
Lokalt fel $l_k = y_k - u_{k-1}(t_k)$ (u_{k-1} lösningen av ODE genom punkten (t_{k-1}, y_{k-1})	det fel som görs i ett steg i den numeriska metoden
Noggrannhet för numerisk metod	$l_k = O(h_k^{p-1})$

9.3.3 **Implicita metoder**

Bättre stabilitetsområde fås om man har möjlighet att använda information vid tiden t_{k+1} vilket gör metoden implicit.

Euler bakåt

$$y_{k+1} = y_k + hf(t_{k+1}, y_{k+1})$$

För att kunna använda en implicit metod behöver man veta var man skall börja. Den informationen hämtas lättast ur en explicit metod t.ex Euler framåt.

Ex Ta problemet

$$\begin{cases} y' = -y^3 \\ y(0) = 1 \end{cases}$$

Vi väljer steglängd $h = 0.5$

$$y_1 = y_0 + 0.5y_1^3$$

Nu tillämpar vi Euler framåt

$$y_1 = y_0 - 0.5y_0^3 = 0.5 \quad \text{(startgissning)}$$

Detta ger $y_0 = 1$

$$y_1 \approx 0.7709$$

Vi bestämmer nu stabiliteten av Euler bakåt:

ODEs scalär test $y' = \lambda y$

$$y_{k+1} = y_k + h\lambda y_{k+1}$$

$$y_k = y_{k+1}(1 - h\lambda)$$

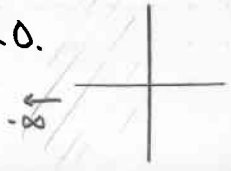
$$y_k = \left(\frac{1}{1-h\lambda}\right)^k y_0$$

För att nu Euler bakåt skall vara stabil måste $\left|\frac{1}{1-h\lambda}\right| \leq 1$

Vilket gäller $\forall h > 0$ när $\mathrm{Re}(\lambda) < 0$.

Stabil metod.

Ovillkorligt stabil metod.



10. Gränsvärdes problem för ODEs

Ett begynnelsevärdes problem för ODE innebär att de villkor som ges, avser endast en punkt. Gränsvärdesproblem avser däremot mer än en punkt. Oftast intervall ändpunkterna.

Ett allmänt två punkts gränsvärdes problem för en ODE av första ordningen har formen

$$y' = f(t, y) \qquad a < t < b$$

med gränsvillkoren

$$g(y(a), y(b)) = 0$$

där $f: \mathbb{R}^{n+1} \to \mathbb{R}^n$ och $g: \mathbb{R}^{2n} \to \mathbb{R}^n$

Gränsvillkoren sägs vara separata då given komponent av g enbart innehåller värden av a eller b, men ej båda. Respektive linjärt om det har formen

$$B_a y(a) + B_b y(b) = c$$

där $B_a, B_b \in \mathbb{R}^{n \times n}$, $c \in \mathbb{R}^n$

10.3 Inskjutningsmetoden

Ex Vi har följande tvåpunkts randvärdesproblem (RVP)

$$\begin{cases} \alpha y'' + \beta y' + \gamma y = f(t) & \text{där } 0 \le t \le 1 \\ y(0) = c_1, \quad y(1) = c_2 \end{cases}$$

Vi söker $y(t)$ givet $f(t), \alpha, \beta, \gamma, c_1, c_2$

Lösning med inskjutningsmetoden:

Skriv om (RVP) som system av 1:a ordningen (BVP)

$$\begin{cases} y_1 = y \\ y_2 = y' \end{cases}$$

Vi får då (BVP)

$$\begin{cases} y_1' = y_2 \\ y_2' = \frac{1}{\alpha}\left[\beta y_2 - \gamma y_1 + f(t)\right] \\ y_1(0) = c_1 \\ y_2(0) = s \leftarrow \text{Obs} \end{cases}$$

Här är s en parameter (variabel)

Vi väljer nu s så att lösningen

$y_1(1, s) = c_2$ (en ekvation i s)

Vi har alltså en ekvation i variabeln s att lösa

$g(s) \equiv y_1(1, s) - c_2 = 0$

Denna löses t. ex. med sekantmetoden

$$s_{k+1} = s_k - \frac{\left[y(1, s_k) - c_2\right](s_k - s_{k-1})}{y_1(1, s_k) - y_1(1, s_{k-1})}$$

Vi behöver alltså två startapproximationer s_0, s_1