TMA4135 Matematikk 4D
Kompendium i numerikk

Eirik Refsdal

2. august 2005

En mangel ved dagens "autorative" kompendium i matematikk 4, er at numerikk-biten i matematikk 4D er fullstendig utelatt. Dette er ikke et forsøk på å lage et dyptpløyende og fantastisk kompendium, men er kun ment som en oppsummering av de metodene i Kreyszigs *Advanced Engineering Mathematics* (8th Ed.) som er pensum i TMA4135 Matematikk 4D. Samt at notatet om Newtons metode for systemer er tatt med, og forsøkt gjort lesbart.

Merk også at det er skrevet fint lite om konvergenshastigheter og feilvurderinger her. Det får bli en annen gang :)

Dette dokumentet er tilgjengelig i PDF- og LaTeX-format fra
http://www.refsdal.no/eirik/dokumenter/

Innhold

1 Numerical Methods in General [K17]

1.1 Introduction [K17.1]

Det eneste interessante som står her er egentlig avsnittet om feilverdier ved numeriske metoder, som egentlig er opplagt.

$$a = \tilde{a} + \varepsilon, \text{True value = Approximation + Error.} \tag{1}$$

De nevner også den **relative feilen** ε_r, som er definert ved

$$\varepsilon_r = \frac{\varepsilon}{a} = \frac{a - \tilde{a}}{a} = \frac{\text{Error}}{\text{True value}} \tag{2}$$

Avslutningsvis poengterer de at enhver numerisk metode bør oppgis sammen med et feilestimat.

1.2 Solution of Equations by Iteration [K17.2]

Konseptet med metodene i dette kapittelet er at vi har en likning (typisk $f(x) = 0$), kjører gjennom metoden med en gjettet inputverdi x_0, bruker den utregnede verdien som ny inputverdi og gjentar denne prosedyren (naturligvis med unntak av å gjette inputverdi, siden vi i alle steg unntatt steg én bruker den utregnede verdien fra forrige iterasjon) til vi har fått et akseptabelt svar.

1.2.1 Fixed-Point Iteration for Solving Equations $f(x) = 0$

$$x_{n+1} = g(x_n) \tag{3}$$

Eks: K17.2.E1: An interation process, $f(x) = x^2 - 3x + 1 = 0$. Det første vi må gjøre her er å skrive om likningen til å ha bare x på venstre side. Vi ser at dette kan gjøres ved å dele enten på x eller 3. Under vises utregning med deling på 3.

$$\begin{aligned} x &= \frac{1}{3}(x^2 + 1) \\ x_{n+1} &= \frac{1}{3}(x_n^2 + 1) \end{aligned}$$

Nå er det bare å sette inn initialverdien for x og kjøre det ønskede antall iterasjoner.

1.2.2 Newton's Method for Solving Equations $f(x) = 0$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4}$$

Også kjent som Newton-Raphsons metode. Hele poenget her er å ta uttrykket du måtte få oppgitt, skrive det om til $f(x) = 0$ på et eller annet vis, derivere det og bare fylle inn i formelen over.

Eks: K17.2.11: Design a Newton iteration for cube roots and compute $\sqrt[3]{7}$.

$$\begin{aligned} x &= \sqrt[3]{7} \\ x^3 &= 7 \\ f(x) &= x^3 - 7 = 0 \end{aligned}$$

Her har vi nå en fin $f(x)$ som vi kan derivere og få $f'(x) = 3x^2$. Så er det bare å sette inn i formelen og vips er vi i mål.

1.2.3 Secant Method for Solving Equations $f(x) = 0$

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} \tag{5}$$

Sekantmetoden er mer eller mindre lik Newtons metode, men den deriverte i uttrykket er, som vi ser av formelen over, byttet ut med et uttrykk for sekanten mellom punktene $(x_n, f(x_n))$ og $(x_{n-1}, f(x_{n-1}))$.

1.3 Interpolation [K17.3]

Målet med de følgende interpolasjonsmetodene er å finne et polynom, p_n, som passer inn med et sett gitte punktet. Motivasjonen for å finne et polynom er at dette er kontinuerlig og fint og både kan deriveres og integreres.

1.3.1 Lagrange Interpolation

$$f(x) = p_n(x) = \sum_{k=0}^{n} L_k(x) f_k = \sum_{k=0}^{n} \frac{l_k(x)}{l_k(X_k)} f_k \tag{6}$$

Over er den generelle formelen for Lagrangeinterpolasjon, og under følger måten man faktisk må sette det opp på når man skal gjøre oppgaver.

$$\begin{aligned} p_1(x) &= L_0(x) f_0 + L_1(x) f_1 \\ &= \frac{x - x_1}{x_0 - x_1} \times f_0 + \frac{x - x_0}{x_1 - x_0} \times f_1 \end{aligned}$$

For p_1 er det to ledd i uttrykket (og altså to L-uttrykk, L_0 og L_1), for p_2 er det tre, osv. Disse L-uttrykkene settes opp som følger:

- Hvis det er 2 L-uttrykk, skal det være ett "ledd" i teller og ett i nevner, hvis det er 3 L-uttrykk, skal det være to "ledd" i teller og to i nevner, osv.
- Telleren skal være $\prod(x - x_j)$ der x_j er alle verdier fra 0 til n, unntatt det tallet i rekket som vi lager en L-verdi for akkurat nå. Altså: Gitt at vi har tre ledd (0, 1 og 2), så skal vi for L_1 ha telleren $(x - x_0)(x - x_2)$. Vi hopper over 1, men tar alle de andre.
- Nevneren i den tilsvarende situasjonen vil være $(x_1 - x_0)(x_1 - x_2)$. Altså omtrent som telleren, men vi bruker x_0 i stedet for bare x.

Se f.eks. eksempel 1 side 850 i Kreyzsig.

En ulempe med Lagrange er at vi må lage et helt nytt uttrykk så fort vi skal kjøre en iterasjon til.

1.3.2 Newton's Divided Difference Interpolation

$$\begin{aligned} f(x) \approx\ & f_0 + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] + \cdots \\ & + (x - x_0)(x - x_1) \cdots (x - x_{n-1} f[x_0, \cdots, x_n] \end{aligned}$$

Det viktige her (og for de to etterfølgende metodene) er å sette opp en differansetabell (se tabellen under, som er hentet fra side 855 i Kreyszig). Poenget med denne tabellen er å sette inn oppgitte x-verdier med tilhørende $f(x)$ (de to

kolonnene lengst til venstre). Kolonne 3 er $f[x_0, x_1]$ fra uttrykket over, kolonne 4 er $f[x_0, x_1, x_2]$, osv.

Verdiene i kolonne 3 regnes ut ved å ta differansen av $f(x)$-verdiene i kolonnen til venstre og dele på differansen av de tilhørende x-verdiene. Pass på at du tar rette x-verdier. Det skal alltid være "ytterverdiene". I kolonne 5 skal du feks. ta differansen mellom de to tallene fra kolonne 4, men dele på differansen mellom 11.0 og 8.0.

x_j	$f_j = f(x_j)$	$f[x_j, x_{j+1}]$	$f[x_j, x_{j+1}, x_{j+2}]$	$f[x_j \cdots x_{j+3}]$
8.0	2.079 442			
		0.117 783		
9.0	2.197 225		-0.006 433	
		0.108 134		0.000 411
9.5	2.251 292		-0.005 200	
		0.097 735		
11.0	2.397 895			

Så er det bare å sette verdiene inn i uttrykket over.

$$\begin{aligned} f(x) \approx p_3(x) &= f_0 + (x - x_0)f[x_0, x_1] + (x - x_0)(x - x_1)f[x_0, x_1, x_2] + \\ &\quad (x - x_0)(x - x_1)(x - x_2)f[x_0, x_1, x_2, x_3] \\ &= 2.079442 + (x - 8.0) \times 0.117783 + (x - 8.0)(x - 9.0) \times \\ &\quad (-0.006433) + (x - 8.0)(x - 9.0)(x - 9.5) \times 0.000411 \end{aligned}$$

Så er det bare å sette inn verdien for x, feks. 9.2, og regne ut.

Det fine med denne metoden er at vi slipper å lage et helt nytt uttrykk når vi skal kjøre en iterasjon til. Vi kan bare legge til et nytt ledd til det gamle uttrykket, og kjøre på.

I tillegg er det også verdt å merke seg at den takler tilfeldig avstand mellom punktene, sånn som rekken 8.0, 9.0, 9.5, 11.0 over.

1.3.3 Newton's Forward Difference Formula

$$\begin{aligned} f(x) \approx p_n(x) &= \sum_{s=0}^{n} \binom{r}{s} \Delta^s f_0 \\ &= f_0 + r\Delta f_0 + \frac{r(r-1)}{2!} + \cdots + \\ &\quad \frac{r(r-1)\cdots(r-n+1)}{n!} \Delta^n f_0 \end{aligned}$$

For praktisk utregning er det viktig å merke seg at $r = \frac{x-x_0}{h}$, hvor x er verdien vi skal finne, x_0 er det første punktet vi har i det oppgittet datasettet vårt og h er steglengden (avstanden mellom hvert datapunkt).

For utregning i praksis setter man opp en differansetabell og setter inn tallene fra den i uttrykket. Se side 857 i Kreyzsig for et eksempel. Husk for all del å trekke fra 1 mer i r i telleren for hvert ledd. De mystiske Δ^s-uttrykkene er tallene fra differansetabellen, som lett vist på side 857.

Denne metoden virker kun på datasett hvor det er lik avstand mellom datapunktene.

1.3.4 Newton's Backward Difference Formula

Metoden er i grunnen helt lik den forrige, men i r-uttrykkene i telleren legger vi til 1 for hver gang i stedet for å trekke fra. Se eksempel side 858.

1.4 Numerical Integration and Differentiation [K17.5]

1.4.1 Rectangular Rule. Trapezoidal Rule

Rectangular rule først:

$$J = \int_a^b f(x)dx \approx h[f(x_1*) + f(x_2*) + \cdots + f(x_n*)] \tag{7}$$

Hvor h i uttrykket over er $\frac{b-a}{n}$.

Så trapezoidal rule:

$$J = \int_a^b f(x)dx \approx h[\frac{1}{2}f(a) + f(x_1) + f(x_2) + \cdots + f(x_{n-1} + \frac{1}{2}f(b)] \tag{8}$$

1.4.2 Simpson's Rule of Integration

$$\int_a^b f(x)dx \approx \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots 2f_{2m-2} + 4f_{2m-1} + f_{2m}) \qquad (9)$$

hvor $h = (b-a)/2m$ og $f_j = f(x_j)$.

Denne er også oppgitt i Rottmann, side 174.

1.4.3 Adaptive Integration

Adaptiv integrering vil si at man prøver å tilpasse h til forandringen i $f(x)$; lave verdier av h der forandringen i $f(x)$ er stor og store verdier av h der forandringen i $f(x)$ er liten. Konseptet kan brukes sammen med hvilken som helst metode for numerisk integrasjon. Se side 876 i Kreyszig for et eksempel.

2 Numerical Methods in Linear Algebra [K18]

2.1 Linear Systems: LU-Factorization, Matrix Inversion [K18.2]

Cluet med de følgende metodene er å faktorisere en matrise A til de to matrisene L og U, henholdsvis *lower triangular* og *upper triangular*.

Vi bruker her eksempel K18.2.1 (side 895-896) for å illustrere de generelle poengene ved metodene.

$$A = \begin{bmatrix} 3 & 5 & 2 \\ 0 & 8 & 2 \\ 6 & 2 & 8 \end{bmatrix} = LU = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & 2u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}$$

Over ser vi rent visuelt hvor de to matrisene har fått navnene sine fra. Videre kan vi også sette opp følgende uttrykk:

$$Ax = LUx = b \tag{10}$$

2.1.1 Doolittle's Method

Denne metoden går ut på at vi tar det generelle grunnlaget fra forrige avsnitt, og sier følgende:

$$Ly = b, \text{ hvor } y = Ux \tag{11}$$

Vi løser så først for y og deretter for x. Se resten av eksempel K18.2.1 for en fullstendig gjennomgang av metoden.

2.1.2 Crout's Method

Denne metoden er helt like Doolittles metode, rent bortsett fra at det er matrisen U som har en diagonal bestående av kun enere.

2.1.3 Cholesky's Method

Choleskys metode er svært lik de to forrige metodene, men den lille forskjellen her er at $U = L^T$. Se Kreyszig side 897 for eksempler.

2.2 Linear Systems: Solution by Iteration [K18.3]

Metodene i forrige kapittel er såkalte *direkte metoder*, som krever en rekke utregninger før det til slutt kommer ut et svar. Metodene i dette kapittelet er såkalte *indirekte* eller *iterative metoder*, altså metoder hvor vi begynner med et estimat og får mer og mer nøyaktige svar for hver runde med utregnigner vi foretar.

Iterative metoder brukes hvis konvergenshastigheten er stor eller systemene inneholder svært mange nuller.

2.2.1 Gauss-Seidel Iteration Method

Trikset i denne metoden er å skrive om et likningsett av typen

$$\begin{array}{rcrcrcrcl} x_1 & - & 0.25x_2 & - & 0.25x_3 & & & = & 50 \\ -0.25x_1 & + & x_2 & & & - & 0.25x_4 & = & 50 \\ -0.25x_1 & & & + & x_3 & - & 0.25x_4 & = & 25 \\ & - & 0.25x_2 & - & 0.25x_3 & + & x_4 & = & 25 \end{array}$$

til formen

$$\begin{array}{rcrcrcrcl} x_1 & = & & 0.25x_2 & + & 0.25x_3 & & + & 50 \\ x_2 & = & 0.25x_1 & & & & + \; 0.25x_4 & + & 50 \\ x_3 & = & 0.25x_1 & & & & + \; 0.25x_4 & + & 25 \\ x_4 & = & & 0.25x_2 & + & 0.25x_3 & & & 25 \end{array}$$

Deretter velger vi estimater for alle x-verdiene, feks. $x_1^{(0)} = 100, x_2^{(0)} = 100, x_3^{(0)} = 100, x_4^{(0)} = 100$, og begynner den iterative prosessen.

Først regner vi ut en ny x_1, $x_1^{(1)}$, setter inn denne for x_1 i tabellen, regner ut en ny x_2, $x_2^{(2)}$, setter inn denne i tabellen, osv. Slik fortsetter vi til ønsket presisjon er oppnådd eller antallet ønskede iterasjoner er gjennomført.

2.2.2 Jacobi Iteration

Jacobi-iterasjon er helt lik Gauss-Seidel-iterajon, med ett bittelite unntak; ingen av de utregnede x-verdiene benyttes før man har gjennomført en hel iterasjon (altså regnet ut samtlige x-verdier).

3 Numerical Methods for Differential Equations [K19]

3.1 Methods for First-Order Differentical Equations [K19.1]

Disse metodene er såkalte "steg for steg"-metoder, hvor vi først regner ut $y_0 = y(x_0)$ og deretter fortsetter å regne ut verdier for $y(x)$ ved x-punkter som følger:

$$\begin{aligned} x_1 &= x_0 + h \\ x_2 &= x_0 + 2h \\ x_3 &= x_0 + 3h \\ &\vdots \\ x_N &= x_0 + Nh \end{aligned}$$

3.1.1 Euler Method (Euler-Cauchy)

$$y_{n+1} = y_n + hf(x_n, y_n) \tag{12}$$

3.1.2 Improved Euler Method (Heun's Method)

I denne metoden regner vi først ut hjelpeverdien y^*_{n+1},

$$y^*_{n+1} = y_n + hf(x_n, y_n) \tag{13}$$

og deretter den nye verdien,

$$y_{n+1} = y_n + \frac{1}{2}h\left[f(x_n, y_n) + f(x_{n+1}, y^*_{n+1})\right] \tag{14}$$

3.1.3 Runge-Kutta Methods

Metoden gir en løsning på initialverdiproblemet $y' = f(x, y), y(x_0) = y_0$ ved ekvidistante punkter, $x_1 = x_0 + h, x_2 = x_0 + 2h, \cdots, x_N = x_0 + Nh$.

Først regner vi ut fire hjelpeverdier, k_1, k_2, k_3 og k_4,

$$
\begin{aligned}
k_1 &= hf(x_n, y_n) && (15)\\
k_2 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) && (16)\\
k_3 &= hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) && (17)\\
k_4 &= hf(x_n + h, y_n + k_3) && (18)\\
& && (19)
\end{aligned}
$$

Deretter regner vi ut den nye verdien, y_{n+1},

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{20}$$

3.1.4 Runge-Kutta-Fehlberg

RKF kan umulig bli gitt på eksamen med mindre de oppgir formlene, simpelthen fordi det er alt for mange uinteressante koeffisienter å holde rede på. Ikke minst vil det være et sant helvete å skulle gjennomføre metoden vha. en HP30S.

3.2 Methods for Systems and Higher Order Equations [K19.3]

3.2.1 Runge-Kutta Methods for Systems

Her brukes bare de samme formlene som i vanlig Runge-Kutta, men alt i alt er det likevel litt mer grisete. Se eksempel K19.3.2 for en full gjennomgang av et faktisk regneeksempel.

4 Andre metoder

4.1 Newtons metode for systemer av ikke-lineære likninger

Newtons (eller Newton-Raphsons) metode er en numerisk metode for å finne numeriske løsninger av et system av ligninger:

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= 0 \\ f_2(x_1, \ldots, x_n) &= 0 \\ &\vdots \\ f_n(x_1, \ldots, x_n) &= 0 \end{aligned}$$

Dette er egentlig ikke en spesielt komplisert metode, men pensumnotatet som prøver å beskrive metoden er ikke akkurat definisjonen på forståelig. Her følger derfor en mer "rett på sak"-fremgangsmåte for å faktisk løse oppgaver av denne typen. Det gjør vi like gjerne med et eksempel.

Ifølge faglærer har det hendt at det har blitt gitt systemer med tre likninger til eksamen, så det er kanskje en fordel å om nødvendig friske opp matrisekunnskapene sine til å dekke også denslags.

Eks: Eksamen, august 2003, oppgave 7c

Gjør én iterasjon med Newtons metode på det ikke-lineære likningssystemet

$$\begin{aligned} 10y_1 - y_2 - 20 &= 0 \\ y_1 + (9 + y_1^2)y_2 &= 0 \end{aligned}$$

Bruk $y_1 = 2$ og $y_2 = 0$ som startverdier.

Løsning

Først og fremst må man ha klart for seg hva vi er ute etter å finne, og det er de neste verdiene av y_1 og y_2. Vi kaller initialverdiene for $y_{1,0}$ og $y_{2,0}$, og skal finne $y_{1,1} = y_{1,0} + \Delta y_{1,0}$ og $y_{2,1} = y_{2,0} + \Delta y_{2,0}$.

Det første vi må gjøre er å sette opp en Jacobimatrise for likningssettet, og den er på formen,

$$J = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

Som vi ser øker x-verdiene langs den horisontale aksen og f-verdiene langs den vertikale. Vi ser altså at det er snakk om å derivere likning f_n med hensyn på variabel x_n for de forskjellige cellene/elementene i matrisen.

I vårt tilfelle blir dette

$$J(y_{1,0}, y_{2,0}) = \begin{bmatrix} 10 & -1 \\ 1 + 2y_1y_2 & 9 + y_1^2 \end{bmatrix}$$

For å finne $y_{1,1}$ og $y_{2,1}$ må vi så løse likningen

$$J(y_{1,0}, y_{2,0}) \begin{pmatrix} \Delta y_{1,0} \\ \Delta y_{2,0} \end{pmatrix} = - \begin{pmatrix} 10y_{1,0} - y_{2,0} - 20 \\ y_{1,0} + (9 + y_{1,0}^2)y_{2,0} \end{pmatrix}$$

med hensyn på $(\Delta y_{1,0}\ \Delta y_{2,0})^{\mathrm{T}}$,

Vi setter så inn initialverdiene våre, altså tallverdier for $y_{1,0}$ og $y_{2,0}$, og får

$$\begin{pmatrix} 10 & -1 \\ 1 & 13 \end{pmatrix} \begin{pmatrix} \Delta y_{1,0} \\ \Delta y_{2,0} \end{pmatrix} = - \begin{pmatrix} 0 \\ 2 \end{pmatrix}$$

Dette gir videre

$$\begin{aligned} \begin{pmatrix} \Delta y_{1,0} \\ \Delta y_{2,0} \end{pmatrix} &= - \begin{pmatrix} 10 & -1 \\ 1 & 13 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 2 \end{pmatrix} \\ & - \frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix} \begin{pmatrix} 0 \\ 2 \end{pmatrix} \\ & - \frac{1}{131} \begin{pmatrix} 13 & 1 \\ -1 & 10 \end{pmatrix} \begin{pmatrix} 0 \\ 2 \end{pmatrix} \end{aligned}$$

Enkel matrisemultiplikasjon gir $(13 \times 0) + (1 \times 2) = 2$ og $(-1 \times 0) + (10 \times 2) = 20$, vi multipliserer videre med faktoren $-\frac{1}{131}$ og ender opp med at

$$\begin{pmatrix} \Delta y_{1,0} \\ \Delta y_{2,0} \end{pmatrix} = - \begin{pmatrix} 2 \\ 20 \end{pmatrix}$$

Det aller siste vi må gjøre er å regne ut de nye verdiene $y_{1,1}$ og $y_{2,1}$:

$$\begin{aligned} y_{1,1} = y_{1,0} + \Delta y_{1,0} = 2 - 0,02 &= \underline{\underline{1,98}} \\ y_{2,1} = y_{2,0} + \Delta y_{2,0} = 0 - 0,15 &= \underline{\underline{-0,15}} \end{aligned}$$