OH till Föreläsning 11, Numme O1, 120229

Hela GKN-boken & hela kursen!

God programmeringsteknik

⋆ **Tänk efter före:**
- Definiera problemet (VAD skall göras?)
- Bestäm algoritm (och lagrings-struktur)
- Dela upp i små delar

⋆ **Skriv sedan kod**
- Dela upp i små delar
- Bra variabelnamn
- Lägg data i variabler/konstanter
- Använd färdig kod/rutiner
- Använd FOR- och WHILE-slingor! Undvik kod-upprepning!
- Använd FOR då antalet upprepningar är känt.
- Använd WHILE då antalet upprepningar inte är känt.
- Kommentera koden medan du skriver den.
- Indentera - Snygg kod oftast rätt!
- Överför info mellan programdelarna som parametrar. Undvik GLOBAL.

⋆ **Debugga sedan koden**
- Debugga både delarna och helheten!
- Om något fel envisas: Handjaga detaljerna, 'antag inget'!
- Fundera igenom vilka fel som kan uppstå.
  (30% av ovana programmerares programrader exekveras aldrig).

När sedan programmet är färdigt — skriv om det. Det är normalt först **tredje** versionen som blir bra.
(Att tänka på både vid köp och försäljning av kod!).

⋆ **Effektivitet**
- Använd färdig-debuggad kod!
- Använd bra metoder/algoritmer.
- Tag ut onödiga beräkningar ur loopar.
  - Förkompilera koden.
  - Förallokera vektorer.
  - Använd Matlabs punktnotation i stället för FOR-slinga.

Tänk dock på att de allra flesta program körs färre än 10 gånger — och det är den totala tiden som skall optimeras.

Grundläggande idéer: räta linjer och upprepning

Approximera funktionen en liten bit med en rät linje

Men olika val av linjer ger olika svar, en del är bättre än andra. Oftast, men inte alltid, blir det bättre ju kortare linjer man har. Exempel:

$$f'(x) \approx \frac{\Delta y}{\Delta x} \qquad \text{gjord som} \qquad \begin{array}{ll} \text{enkelsidig} & f'(x) \approx \dfrac{f(x+h)-f(x)}{h} \qquad \text{med} \quad E_{trunk} \approx c\,h \\ \text{centrerad} & f'(x) \approx \dfrac{f(x+h)-f(x-h)}{2h} \qquad \text{med} \quad E_{trunk} \approx c\,h^2 \end{array}$$

Det fel som vi har infört genom att approximera den riktiga funktionen med en rät linje kompenserar vi sedan för genom någon form av upprepning. På så vis blir trunkeringsfelet till slut acceptabelt litet. Upprepningen hjälper oss också att se felets storlek. Upprepningen sker i form av iteration eller rekursion:

Iteration

T.ex. Newton-Raphson:

$$x_{n+1} = x_n - f(x_n)/f'(x_n)$$

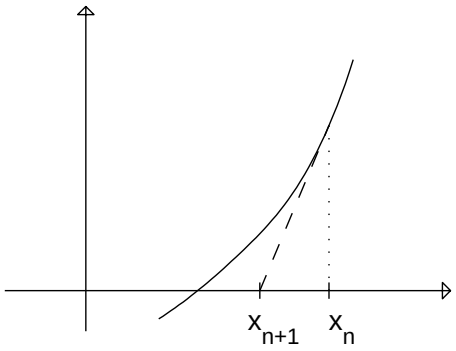
x_n och x_{n+1} beskriver (allt bättre) samma värde.

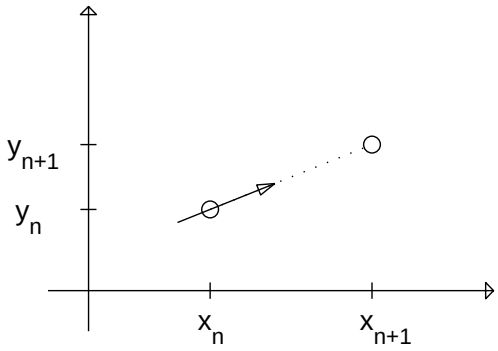
Rekursion

T.ex. Eulers metod:

$$\begin{cases} y_{n+1} = y_n + h\,y'(x_n, y_n) \\ x_{n+1} = x_n + h \end{cases}$$

y_n och y_{n+1} beskriver olika värden. y_n behövs för att beräkna y_{n+1}.





Beteckningar

x betecknar exakta värdet. Det är sällan känt.
$\tilde{x}$ betecknar närmevärdet. Det är det vi räknar ut, vår approximation!

$E_x = \max|\tilde{x} - x|$ = gränsen för absoluta felet.
$R_x = |\frac{E_x}{x}| \approx |\frac{E_x}{\tilde{x}}|$ = gränsen för relativa felet.

E_{tab} osäkerhet i utdata pga osäkra indata.
E_{ber} osäkerhet i utdata pga avrundade mellanresultat.
E_{trunk} osäkerhet i utdata pga kapad Taylorutveckling (eller kapad iteration eller ändligt steg).
E_{pres} osäkerhet i utdata pga egen slutlig presentationsavrundning.

Tabellfelet, E_{tab}

Tabellfelet uppstår genom felfortplantning. Om indata är osäkra blir även resultatet osäkert:

$$w = f(x, y, z, \ldots) \quad \text{där} \quad x = \tilde{x} \pm E_x,\ y = \tilde{y} \pm E_y,\ z = \tilde{z} \pm E_z,\ \text{osv} \quad \Longrightarrow \quad w = \tilde{w} \pm E_w$$

Tabellfelet skattas med tex:

Allmänna felfortplantningsformeln

$$\tilde{w} = f(\tilde{x}, \tilde{y}, \tilde{z}) \qquad E_w = \left|\frac{\partial f}{\partial x}\right| E_x + \left|\frac{\partial f}{\partial y}\right| E_y + \left|\frac{\partial f}{\partial z}\right| E_z + \ldots \qquad \text{där derivatorna beräknas i punkten } (\tilde{x}, \tilde{y}, \tilde{z})$$

Störningsräkning (dvs AFFF med numeriskt skattade derivator — praktisk!)

$$\tilde{w} = f(\tilde{x}, \tilde{y}, \tilde{z}) \qquad E_w = |f(\tilde{x}+E_x, \tilde{y}, \tilde{z}) - f_o| + |f(\tilde{x}, \tilde{y}+E_y, \tilde{z}) - f_o| + |f(\tilde{x}, \tilde{y}, \tilde{z}+E_z) - f_o| + \ldots \qquad \text{där } f_o = \tilde{w}$$

Min-och-max-räkning

Finn maximala och minimala resultatet, $f_{\max}$ och $f_{\min}$, tex genom att beräkna funktionsvärdet med alla möjliga kombinationer av störda parametrar. Denna metod blir mycket arbetssam om man har många parametrar!

$$\tilde{w} = (f_{\max} + f_{\min})/2 \qquad E_w = (f_{\max} - f_{\min})/2$$

Beräkningsfelet, E_{ber}

Detta fel är svårskattat. Datorn avrundar ju alla sina mellanresultat till det antal siffror den räknar med. Försök att undvika *kancellation* och *utskiftning* så minimeras beräkningsfelet.

Ekvationslösning

Förbehandling:

- Bakgrundsinformation.
- Grafisk teknik och uppdelning.
- Matematik (antag x liten eller stor och förenkla, kolla tecken hos derivatan, mm)
- Intervallhalvering.

Fortsättningsmetoder:
De är alla iterativa metoder: gissa x_0, förbättra enligt $x_{n+1} = x_n - t_n$, fortsätt tills t blir tillräckligt liten. Om $t_{n+1}/t_n \to \textit{konstant}$ så är konvergensen linjär. Om $t_{n+1}/t_n^2 \to \textit{konstant}$ så är konvergensen kvadratisk.

En ekvation med en obekant

• Newton-Raphson	$t_n = f(x_n)/f'(x_n)$	$t_{n+1} \approx K \cdot t_n^2$
• Sekantmetoden	$t_n = f(x_n) \cdot \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}$	$t_{n+1} \approx K \cdot t_n \cdot t_{n-1}$
• Fixpunktsmetoden		$t_{n+1} \approx K \cdot t_n, \quad \lvert K\rvert \ll 1$

Ickelinjära ekvationssystem

Sökt $\bar{x}$ så att $\bar{f}(\bar{x}) = \bar{0}$. Den enda metod vi lär oss är iterativ, Newtons metod:

1) Gissa $\bar{x}_0$.
2) Beräkna vektorn $\bar{f} = \bar{f}(\bar{x}_n)$ och matrisen $J = \frac{\partial \bar{f}}{\partial \bar{x}}(\bar{x}_n)$.
3) Lös ekvationssystemet $J\,\bar{t} = \bar{f} \quad \Longrightarrow \quad \bar{t}_n = J_n^{-1}\bar{f}_n$
4) Bättre approximation $\bar{x}_{n+1} = \bar{x}_n - \bar{t}_n$
5) Om inte $||\bar{t}_n||$ tillräckligt liten, upprepa från punkt 2 med $\bar{x}_{n+1}$.

Ett vanligt sätt att skatta trunkeringsfelet är att titta på sista använda korrektionen t. Oftast är detta en grov överskattning, men om konvergensen är långsam, $t_n/t_{n-1} > 0.5$, så är det inte ens tillräckligt! Trunkeringsfelet kan då vara större än sista korrektionen! Det är bla därför man måste titta på hurdan konvergens man har. Om den inte är som teorin säger är *något* fel.

Linjära ekvationssystem

$$\begin{aligned} a_{11}\,x_1 + a_{12}\,x_2 + \cdots a_{1N}\,x_N &= b_1 \\ a_{21}\,x_1 + a_{22}\,x_2 + \cdots a_{2N}\,x_N &= b_2 \\ \vdots \qquad\qquad\qquad\qquad &= \vdots \\ a_{N1}\,x_1 + a_{N2}\,x_2 + \cdots a_{NN}\,x_N &= b_N \end{aligned}$$

$Ax = b$ har en lösning om A^{-1} existerar (dvs om $\det(A) \neq 0$ så är $x = A^{-1}b$). Annars har ekvationssytemet ingen eller oändligt antal lösningar.

Normer = ett mått på vektorer

$\|\|x\|\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_N^2}$	″vanliga″
$\|\|x\|\|_\infty = \max_{1\leq i\leq N} \|x_i\|$	största elementet
$\|\|A\|\|_\infty = \max_{1\leq i\leq N} \sum_{j=1}^{N} \|a_{ij}\|$	tyngsta raden

Konditionstalet, $\kappa(A) = ||A||\,\left|\left|A^{-1}\right|\right|$

$\kappa(A) \gg 1 \iff$ Illakonditionerat $\iff$ Känsligt för störningar. Praktisk skattning : $\kappa(A) \simeq \dfrac{R_{ut}}{R_{in}}$

Tidsåtgång

Bandmatris : $T \propto N^1$ Triangulär matris : $T \propto N^2$ Full matris : $T \propto N^3$

Minstakvadratmetoden

Används då man har fler villkor än obekanta, dvs överbestämda ekvationssystem.

Överbestämda linjära ekvationssystem

$$Ac = b$$

där A har fler rader än kolumner. MKV löser systemet så att $||r||_2 = ||b - Ac||_2$ minimeras. Löses som

$$A^TA\,c = A^Tb \quad \Longrightarrow \quad c = \left(A^TA\right)^{-1} A^Tb$$

A^TA har lika många rader som kolumner. Antalet rader i A^TA är detsamma som antalet kolumner i A som är detsamma som antalet obekanta parametrar. Löses i ett steg (dvs direkt, inga iterationer).
Kontroll: $A^Tr = 0$.

Överbestämda ickelinjära ekvationssystem

$$\bar{f}(\bar{c}) = \bar{0}$$

där $\bar{f}$ har fler komponenter än $\bar{c}$. Löses med MKV (Gauss-Newtons metod) så att $\left|\left|\bar{f}\right|\right|_2$ minimeras. Iterativ metod!

1) Gissa $\bar{c}_0$.
2) Beräkna vektorn $\bar{f} = \bar{f}(\bar{c}_n)$ och matrisen $J = \frac{\partial \bar{f}}{\partial \bar{c}}(\bar{c}_n)$.
3) Lös det överbestämda linjära ekvationssystemet $J\,\bar{t} = \bar{f} \quad \Longrightarrow \quad \bar{t}_n = \left(J_n^T J_n\right)^{-1} J_n^T \bar{f}_n$
4) Bättre approximation $\bar{c}_{n+1} = \bar{c}_n - \bar{t}_n$
5) Om inte $||\bar{t}_n||$ tillräckligt liten, upprepa från punkt 2 med $\bar{c}_{n+1}$.

Interpolation

Allmänt: Givet tabell

x	x_1	x_2	$\cdots$	x_N
y	y_1	y_2	$\cdots$	y_N

Sökt $y(x)$ för punkter däremellan.

Polynominterpolation

Naiv ansats	$p(x) = c_1 + c_2\,x + c_3\,x^2 + c_4\,x^3 + \ldots$
Newtons ansats	$p(x) = c_1 + c_2\,(x-x_1) + c_3\,(x-x_1)(x-x_2) + c_4\,(x-x_1)(x-x_2)(x-x_3) + \ldots$
Centrerad ansats	$p(x) = c_1 + c_2\,(x-m) + c_3\,(x-m)^2 + c_4\,(x-m)^3 + \ldots$

- Koefficienterna bestäms med $p(x_i) = y_i$ vilket ger ett linjärt ekvationssystem.
- Med naiva ansatsen får ekvationssystemet ofta stort konditionstal.
- Med Newtons ansats blir ekvationssystemet lättlöst och får ofta lågt konditionstal.
- De olika ansatserna ger olika koefficienter men det erhållna polynomet är detsamma.
- Vid höga gradtal uppträder Runges fenomen.

E_{trunk} skattas genom att göra beräkningar med olika gradtal på polynomet.

Styckvis interpolation

Grundidé: Lägg ett nytt polynom i varje intervall $[x_i, x_{i+1}]$.

Linjär IP: Förstagradspolynom. Behöver y i varje punkt x_i. Drar rät linje mellan varje punktpar.

Hermite: Tredjegradspolynom. Behöver y och y' i varje punkt x_i.

Splines: Tredjegradspolynom. Behöver bara y i varje punkt x_i.
Metoden beräknar/skattar/gissar y' och y'' enligt kravet att de är kontinuerliga.

Integraler $\qquad I = \int_a^b f(x)dx$

Trapetsregeln

$$I \simeq T(h) = h\left\{\frac{1}{2}f(a) + f(a+h) + f(a+2h) + \cdots + f(b-h) + \frac{1}{2}f(b)\right\}$$

$$E_{trunk} = |T(h) - I| = \left|c_1\,h^2 + c_2\,h^4 + c_3\,h^6 + \ldots\right| \simeq |T(h) - T(2h)|$$

$T(h)$ är bra skattning av integralen om
- h tillräckligt litet för att följa kurvan
- $\frac{T(2h)-T(4h)}{T(h)-T(2h)} \simeq 2^2 = 4$

Rombergs metod = Trapetsregeln + Richardsonextrapolation

$$\begin{aligned} T(h) &= I + c_1\,h^2 + c_2\,h^4 + \ldots \\ T(2h) &= I + 4\,c_1\,h^2 + 64\,c_2\,h^4 + \ldots \\ T(4h) &= I + 16\,c_1\,h^2 + 256\,c_2\,h^4 + \ldots \end{aligned} \quad \Longrightarrow \quad \begin{aligned} \hat{T}(h) &= T(h) + \frac{T(h)-T(2h)}{3} = I + c_1 h^4 + \ldots \\ \hat{T}(2h) &= T(2h) + \frac{T(2h)-T(4h)}{3} = I + 16\,c_1 h^4 + \ldots \end{aligned}$$

$$\Longrightarrow \quad \hat{\hat{T}}(h) = \hat{T}(h) + \frac{\hat{T}(h) - \hat{T}(2h)}{15} = I + c_1 h^6 + \ldots$$

$\hat{T}(h)$ bra om $\quad \dfrac{T(2h)-T(4h)}{T(h)-T(2h)} \simeq 2^2 = 4 \quad$ (dvs felet i $T \sim c\,h^2$)

$\hat{\hat{T}}(h)$ bra om $\quad \dfrac{\hat{T}(2h)-\hat{T}(4h)}{\hat{T}(h)-\hat{T}(2h)} \simeq 2^4 = 16 \quad$ (dvs felet i $\hat{T} \sim c\,h^4$)

Matlab: `I=quad8('funk',a,b,tol), Etrunk=abs(tol*I)  % OK om kurvan snäll`
Matlab: `I=quadl('funk',a,b,tol), Etrunk=tol  % OK om kurvan snäll`

Förbehandling:	• Substitution	• Uppdelning	• Kapning	• Partiell integration

Ordinära differentialekvationer — begynnelsevärdesproblem (BV)

Våra metoder klarar "bara" första ordningens differentialekvationer, men det är inget problem ety högre ordningens differentialekvationer kan alltid skrivas om till ett system av första ordningen.

Skriv på standardform : $\dfrac{d\bar{y}}{dx} = \bar{f}(x, \bar{y})$ $\bar{y}(a) = c, \quad \bar{y}(b) = ?$

Lösningsidé: Approximera kurvan (en kort bit) med en rät linje. Linjens lutning ges av differentialekvationen (och metoden, olika metoder ger olika lutning).

Eulers metod

Linjens lutningen beräknas till derivatan i startpunkten:

$$\begin{cases} y_{n+1} = y_n + h\, f(x_n, y_n) & y_0 = c \\ x_{n+1} = x_n + h & x_0 = a \end{cases}$$

$$E_{trunk} = |y(x;h) - y(x)| \simeq c_1\, h + c_2\, h^2 + c_3\, h^3 + c_4\, h^4 + \ldots \simeq |y(x;h) - y(x;2h)|$$

Metoden är lätt att programmera men har låg noggrannhet.

Runge-Kuttas metod

Linjens lutningen beräknas som ett viktat medelvärde av derivatan i startpunkten, mittpunkten och slutpunkten:

$$\begin{aligned} k_1 &= h\, f(x_n, y_n) \\ k_2 &= h\, f(x_n + h/2, y_n + k_1/2) \\ k_3 &= h\, f(x_n + h/2, y_n + k_2/2) \\ k_4 &= h\, f(x_n + h, y_n + k_3) \end{aligned}$$

$$\begin{cases} y_{n+1} = y_n + (k_1 + 2k_2 + 2k_3 + k_4)/6 & y_0 = c \\ x_{n+1} = x_n + h & x_0 = a \end{cases}$$

$$E_{trunk} = |y(x;h) - y(x)| \simeq c_1\, h^4 + c_2\, h^5 + c_3\, h^6 + c_4\, h^7 + \ldots \simeq |y(x;h) - y(x;2h)|$$

Runge-Kutta är lite klurigare att programmera än Eulers metod men den ger mycket bättre svar vid samma mängd beräkningar.

I Matlab finns de två ODE-lösarna `ode23` och `ode45` som bygger på Runge-Kutta och adaptiv steglängd. Trunkeringsfelet skattas genom att jämföra resultatet från två beräkningar med olika steglängd:

```
y0=c; tolval=odeset('RelTol', 1e-6); tolval2=odeset('RelTol', 1e-9);
[xut,yut]=ode45('dydx',[a b],y0,tolval);  n1=length(xut); y1=yut(n1,1);
[xut,yut]=ode45('dydx',[a b],y0,tolval2); n2=length(xut); y2=yut(n2,1);
plot(xut,yut);
svar=y2  % Eftersom sökt y(b) och y2 beräknad med minsta steget.
if n1 ~= n2; Etrunk=abs(y1-y2), end;
```

Ordinära differentialekvationer — randvärdesproblem (RV)

En metod (+en extra) att välja på:

Finita-Differens-Metoden FDM (förr Bandmatrismetoden)

1) Ersätt **alla** derivator med differenser (dvs både i differentialekvationen och randvillkoren).
2) Diskretisera integrations-intervallet.
3) Sätt in differenserna i alla diskretiseringspunkter ⇒ ekvationssystem.
4) Lös ekvationssystemet (med Newtons metod om icke-linjärt).
5) Upprepa från 2 med dubbla antalet diskretiseringspunkter (dvs halva steget) om felet är för stort.

E_{trunk} kommer från vald steglängd (och hur länge man itererar i Newton om man hade ett icke-linjärt ekvationssystem) och skattas genom att jämföra lösningen beräknad med dubbla steget. $E_{trunk} \simeq |y(x;h) - y(x;2h)|$

$$y'(x) \approx \frac{y(x+h) - y(x-h)}{2h} \qquad y''(x) \approx \frac{y(x+h) - 2y(x) + y(x-h)}{h^2}$$

Inskjutningsmetoden (även kallad provskottsmetoden)

1) Gissa det saknade startvärdet.
2) Lös diffekvationen med en metod för begynnelsevärdesproblem (BV).
3) Kolla hur stort felet blev i slutet genom att jämföra med det givna randvillkoret.
4) Gissa ett nytt startvärde.
5) Lös ånyo med BV-metoden.
6) Kolla hur stort felet blev denna gång. Om tillräckligt litet, stanna: vår lösning är OK. Om för stort:
7) Bestäm nytt startvärde utgående från de senaste två gissningarna vi gjort (sekantmetoden).
8) Upprepa från 5.

E_{trunk} kommer från vald steglängd i BV-metoden och hur länge man itererar för att hitta det rätta startvärdet.

Slut på denna nummekurs.
Lycka till med det fortsatta "nummandet"!

Exempel på inskjutningsmetoden.

Beräkna $y(0.0), y(0.1), y(0.2)$ och $y(0.3)$. Använd inskjutningsmetoden och Eulers metod med steget $h = 0.1$.

$$y'' = \left(x^2 + 6y'\right)\, y \qquad \begin{matrix} y(0.3) = 0.7 \\ y'(0.0) = 0.4 \end{matrix}$$

Lösning

Det är ett RV-problem eftersom randvillkoren är givna i olika punkter. Får att kunna använda Eulers metod (som är en BV-metod) måste vi ha alla randvärdena givna i samma punkt. Vi skriver om till standardform:

$$\begin{matrix} u_1 = y(x) \\ u_2 = y'(x) \end{matrix} \quad \Longrightarrow \quad \begin{matrix} u_1' = y'(x) = u_2 \\ u_2' = y''(x) = \left(x^2 + 6y'\right) y = \left(x^2 + 6u_2\right) u_1 \end{matrix}$$

vilket skrivet i vektorform blir : $\quad \bar{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \qquad \bar{u}' = \begin{pmatrix} u_2 \\ \left(x^2 + 6u_2\right) u_1 \end{pmatrix}$

Eulers metod blir då $\quad \begin{cases} (\bar{u})_{n+1} = (\bar{u})_n + h \cdot (\bar{u}')_n \\ x_{n+1} = x_n + h \end{cases}$ med startvärden $\qquad (\bar{u})_0 = \begin{pmatrix} u_1(x_0) \\ u_2(x_0) \end{pmatrix} \equiv \begin{pmatrix} u_{1_0} \\ u_{2_0} \end{pmatrix}$

I vårt fall har vi $u_{2_0} = 0.4$ men vi saknar ett värde på u_{1_0}. I stället har vi fått värdet på y i punkten $x = 0.3$. Med steget $h = 0.1$ och våra beteckningar betyder detta att vi vet värdet på u_1 efter tre steg, $u_{1_3} = 0.7$

Det inskjutningsmetoden innebär är att vi måste gissa ett värde på u_{1_0} och räkna fram vad vi får på u_{1_3}. Om vi får 0.7 betyder det att vi gissade rätt, annars får vi göra om med ett nytt startvärde på u_{1_0}.

Gissning 1

Jag gissar $u_{1_0} = 0.7$ (i brist på fantasi så antar jag att $u_1(x) = y(x)$ inte ändras alltför mycket mellan $x = 0$ och $x = 0.3$). Jag skall nu lösa begynnelsevärdesproblemet; tre steg med Eulers metod ty $x_0 = 0, x_1 = 0.1, x_2 = 0.2, x_3 = 0.3$ (upp till fyra decimaler redovisas):

Steg 1:

$$\bar{u}_0 = \begin{pmatrix} u_{1_0} \\ u_{2_0} \end{pmatrix} = \begin{pmatrix} 0.7 \\ 0.4 \end{pmatrix} \quad \Longrightarrow \quad \bar{u}_0' = \begin{pmatrix} u_{2_0} \\ \left(x_0^2 + 6u_{2_0}\right) u_{1_0} \end{pmatrix} = \begin{pmatrix} 0.4 \\ 1.68 \end{pmatrix}$$

$$\bar{u}_1 = \bar{u}_0 + h \cdot \bar{u}_0' = \begin{pmatrix} 0.7 \\ 0.4 \end{pmatrix} + 0.1 \cdot \begin{pmatrix} 0.4 \\ 1.68 \end{pmatrix} = \begin{pmatrix} 0.7 \\ 0.4 \end{pmatrix} + \begin{pmatrix} 0.04 \\ 0.168 \end{pmatrix} = \begin{pmatrix} 0.74 \\ 0.568 \end{pmatrix}$$

Steg 2:

$$\bar{u}_1 = \begin{pmatrix} u_{1_1} \\ u_{2_1} \end{pmatrix} = \begin{pmatrix} 0.74 \\ 0.568 \end{pmatrix} \quad \Longrightarrow \quad \bar{u}_1' = \begin{pmatrix} u_{2_1} \\ \left(x_1^2 + 6u_{2_1}\right) u_{1_1} \end{pmatrix} = \begin{pmatrix} 0.5680 \\ 2.5293 \end{pmatrix}$$

$$\bar{u}_2 = \bar{u}_1 + h \cdot \bar{u}_1' = \begin{pmatrix} 0.74 \\ 0.568 \end{pmatrix} + 0.1 \cdot \begin{pmatrix} 0.568 \\ 2.5293 \end{pmatrix} = \begin{pmatrix} 0.74 \\ 0.568 \end{pmatrix} + \begin{pmatrix} 0.0568 \\ 0.2529 \end{pmatrix} = \begin{pmatrix} 0.7968 \\ 0.8209 \end{pmatrix}$$

Steg 3:

$$\bar{u}_2 = \begin{pmatrix} u_{1_2} \\ u_{2_2} \end{pmatrix} = \begin{pmatrix} 0.7968 \\ 0.8209 \end{pmatrix} \quad \Longrightarrow \quad \bar{u}_2' = \begin{pmatrix} u_{2_2} \\ \left(x_2^2 + 6u_{2_2}\right) u_{1_2} \end{pmatrix} = \begin{pmatrix} 0.8209 \\ 3.9566 \end{pmatrix}$$

$$\bar{u}_3 = \bar{u}_2 + h \cdot \bar{u}_2' = \begin{pmatrix} 0.7968 \\ 0.8209 \end{pmatrix} + 0.1 \cdot \begin{pmatrix} 0.8209 \\ 3.9566 \end{pmatrix} = \begin{pmatrix} 0.7968 \\ 0.8209 \end{pmatrix} + \begin{pmatrix} 0.0821 \\ 0.3957 \end{pmatrix} = \begin{pmatrix} 0.8789 \\ 1.2166 \end{pmatrix}$$

Denna Euler-räkning kan redovisas kompaktare med följande tabell

n	x_n	$(\bar{u}_n)^T$		$(\bar{u}_n')^T$		$h \cdot (\bar{u}_n')^T$	
0	0.0	0.7000	0.4000	0.4000	1.6800	0.0400	0.1680
1	0.1	0.7400	0.5680	0.5680	2.5293	0.0568	0.2529
2	0.2	0.7968	0.8209	0.8209	3.9566	0.0821	0.3957
3	0.3	0.8789	1.2166				

Jag fick $u_{1_3} = 0.8789$, inte 0.7, alltså har jag gissat fel. Ny gissning behövs.

Gissning 2

Eftersom jag fick ett för stort slutvärde så chansar jag nu på ett lite mindre, jag gissar nu $u_{1_0} = 0.6$. Nu har jag vad jag behöver för att göra en ny Euler-räkning från $x_0 = 0$ till $x_3 = 0.3$ Den redovisas i tabellform:

n	x_n	$(\bar{u}_n)^T$		$(\bar{u}'_n)^T$		$h \cdot (\bar{u}'_n)^T$	
0	0.0	0.6000	0.4000	0.4000	1.4400	0.0400	0.1440
1	0.1	0.6400	0.5440	0.5440	2.0954	0.0544	0.2095
2	0.2	0.6944	0.7535	0.7535	3.1673	0.0754	0.3167
3	0.3	0.7698	1.0703				

Nu fick jag $u_{1_3} = 0.7698$, bättre men inte riktigt bra.

Vidare räkningar

Ett nytt startvärde behövs men denna gång gissar jag inte vilt utan använder sekantmetoden på de värden jag har. Med startgissningen 0.7 blev felet i slutet $0.8789 - 0.7 = 0.1789$ och med startgissningen 0.6 blev felet i slutet $0.7698 - 0.7 = 0.0698$:

$$\text{Sekantmetoden}: \quad z_{n+1} = z_n - f_n \frac{z_n - z_{n-1}}{f_n - f_{n-1}} \qquad \Longrightarrow \qquad 0.6 - 0.0698 \cdot \frac{0.6 - 0.7}{0.0698 - 0.1789} = 0.5361$$

Min nya startgissning blir alltså $u_{1_0} = 0.5361$ Den nya Euler-beräkningen redovisas i tabellform igen:

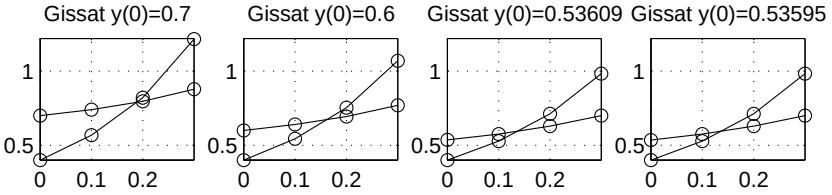
n	x_n	$(\bar{u}_n)^T$		$(\bar{u}'_n)^T$		$h \cdot (\bar{u}'_n)^T$	
0	0.0	0.5361	0.4000	0.4000	1.2866	0.0400	0.1287
1	0.1	0.5761	0.5287	0.5287	1.8331	0.0529	0.1833
2	0.2	0.6290	0.7120	0.7120	2.7119	0.0712	0.2712
3	0.3	0.7002	0.9832				

Slutvärdet är inte riktigt bra, jag kör ett varv till. Nytt startvärde blir $0.5361 - 0.0002 \cdot \frac{0.5361 - 0.6}{0.0002 - 0.7698} = 0.5359$ Den nya Euler-beräkningen redovisas ånyo i tabellform:

n	x_n	$(\bar{u}_n)^T$		$(\bar{u}'_n)^T$		$h \cdot (\bar{u}'_n)^T$	
0	0.0	0.5359	0.4000	0.4000	1.2863	0.0400	0.1286
1	0.1	0.5759	0.5286	0.5286	1.8325	0.0529	0.1833
2	0.2	0.6288	0.7119	0.7119	2.7110	0.0712	0.2711
3	0.3	0.7000	0.9830				

Nu stämmer slutvärdet (och kör man sekantmetod-formeln ett varv till får man korrektionen till startvärdet $1.2 \cdot 10^{-7}$ och samma värden med fyra decimaler). De sökta värdena är således $y(0) = 0.5359$, $y(0.1) = 0.5759$, $y(0.2) = 0.6288$ och (givetvis) $y(0.3) = 0.7000$.

(Trunkeringsfelet i svaret beror av när vi stannar i sekantmetoden samt steglängden i Euler. Här dominerar felet från Euler! Gör man om beräkningarna med steget $h = 0.05$ får man $y(0) = u_{1_0} = 0.5179$, $y(0.1) = u_{1_2} = 0.5610$, $y(0.2) = u_{1_4} = 0.6192$ och $y(0) = u_{1_6} = 0.7000$. (För att få fyra säkra decimaler var jag tvungen att ha $h < 0.0001$, dvs över 3000 steg i varje Euler-beräkning! När jag bytte ut Euler mot ODE45 fick jag svaret med en tiondel så mycket arbete. Använd inte Euler!))



- Rätta värdena med fyra decimaler är $y(0) = 0.4943$, $y(0.1) = 0.5411$, $y(0.2) = 0.6060$ och $y(0.3) = 0.7000$.
- (Om man gör ett stegs Rich.-extr. på Euler-värdena får man rätta svaret med 75% av ODE45-arbetet.)
- (Om man väljer RK med $h = 0.1$ och gör ett stegs Rich. får man rätta svaret med 25% av ODE45-arb.)