F7

- System av ODE - initialvärdesproblem
- Existens & entydighet, Lipschitz, lösningsskaror, fasrum
- Euler, konvergens, fel, konvergensordning
- Stabilitet, känslighet
- Styva ekvationer & Implicita metoder

Initialvärdesproblem, I: Matematik

Lösningars egenskaper

1. En första ordningens linjär ekvation med konstanta koefficienter

$$dy/dt = ay,\ y(0) = c\text{: } y(t) = ce^{at}$$

Oberoende variabel t, ett begynnelsevillkor som väljer ut en lösning ur lösningsskaran. Lösningen är unik och existerar för alla t.

2. En första ordningens o-linjär ekvation (*separabel:* $dy/y^2 = dt$)

$$dy/dt = y^2,\ y(0) = c > 0\text{: } y = c/(1–ct) \text{ för } 0 < t < 1/c$$

Lösningen är unik men existerar bara i ett begränsat intervall.

3. En annan olinjär ekvation:

$$dy/dt = \sqrt{y},\ y(0) = c \geq 0 : y = (\sqrt{c} + t/2)^2 \quad (?)$$

Uppenbarligen är $y = 0$ en lösning för $c = 0$. Men också

$$y(t) = t^2/4, \text{ alla } t > 0$$

Lösningen är inte unik för $c = 0$.

Initialvärdesproblem, II: Matematik

IVP för system av första ordnings differentialekvationer

$$d\mathbf{y}/dt = \mathbf{f}(t,\mathbf{y}), \mathbf{y}(a) = \mathbf{c}, \mathbf{y}, \mathbf{c} \in R^n$$

är normalformen i numeriska metoder för ordinära differentialekvationer. *t* är den oberoende variabeln (ofta tid), ett system av *n* första ordningens differential-ekvationer med begynnelsevärden (initialvärden) givna vid $t = a$ (ofta 0). Kolumnvektorn **y** kallas tillstånds-vektorn. De *n* funktionerna

f_i, $i = 1,2,\ldots,$ n av **y** kan kallas högerledet.

Ex. Newton, massa-fjäder-dämpare

$$\begin{cases} \dot{x} = v \\ m\dot{v} + Kx + Dv = \sin t \end{cases} \Rightarrow m\ddot{x} + Kx + D\dot{x} = f(t)$$

$$\mathbf{y} = \begin{pmatrix} x \\ v \end{pmatrix} : \frac{d}{dt}\mathbf{y} + \mathbf{A}\mathbf{y} = \mathbf{f}, \mathbf{A} = \begin{pmatrix} 0 & -1 \\ \frac{K}{m} & \frac{D}{m} \end{pmatrix}, \mathbf{f} = \begin{pmatrix} 0 \\ \frac{\sin t}{m} \end{pmatrix}$$

Initialvärdesproblem, III: Unicitet

Def. Lipschitz-kontinuitet: **f** kallas L-kontinuerlig i en domän D om det finns en konstant L så att

$$\left\|\mathbf{f}(t,\mathbf{x})-\mathbf{f}(t,\mathbf{y})\right\| \le L\cdot\left\|\mathbf{x}-\mathbf{y}\right\|$$

för alla **x**,**y** i D.

Om **f** är deriverbar duger $L = \max_{\mathbf{x},\mathbf{y}\in D}\left\|\frac{\partial \mathbf{f}(t,\mathbf{y})}{\partial \mathbf{y}}\right\|$

Ex. $\sqrt{y}$ är kontinuerlig men inte L-kontinuerlig i $y = 0$

(på tavlan)

Sats. Om **f**(t,**y**) är L-kontinuerlig i **y** i en omgivning till initialvärdet **c** så existerar en unik lösning en kort stund.

Man kan bevisa satsen genom att visa, att en numerisk lösning konvergerar när steglängden går mot noll.

Initialvärdesproblem, IV: Fasrum mm.

Exempel: satellit. En masspunkt (x,y) med hastighet (u,v) rör sig runt en mycket större massa under inverkan av gravitation.

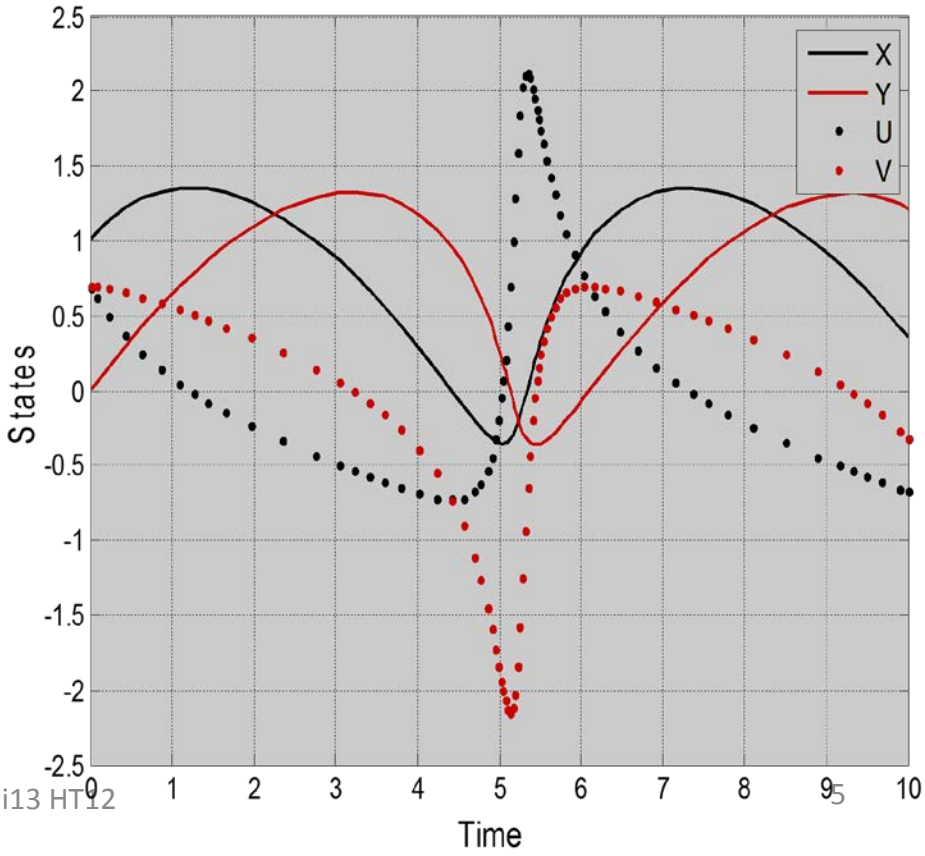
Tillståndrum 4D (x,y,u,v) = fasrum (fas*plan* i 2D)

$$\begin{cases} \dot{u} = -kx/r^3, \dot{x} = u \\ \dot{v} = -ky/r^3, \dot{y} = v \end{cases}; r = \sqrt{x^2 + y^2}$$

Visualisering av lösningar

1. ”Plotta” komponenterna i tillståndsvektorn mot tiden.

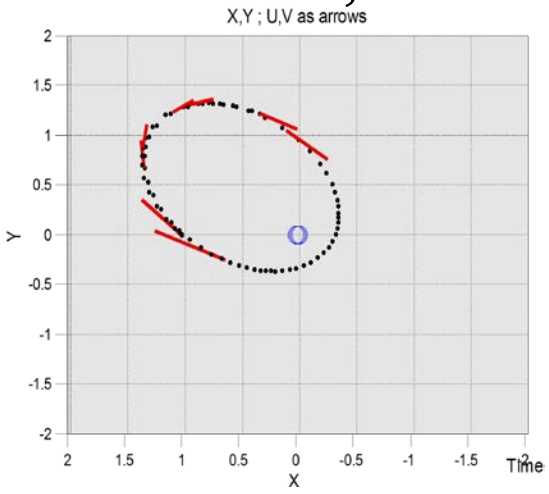
Fyra periodiska kurvor $(x(t),y(t),u(t),v(t))$.



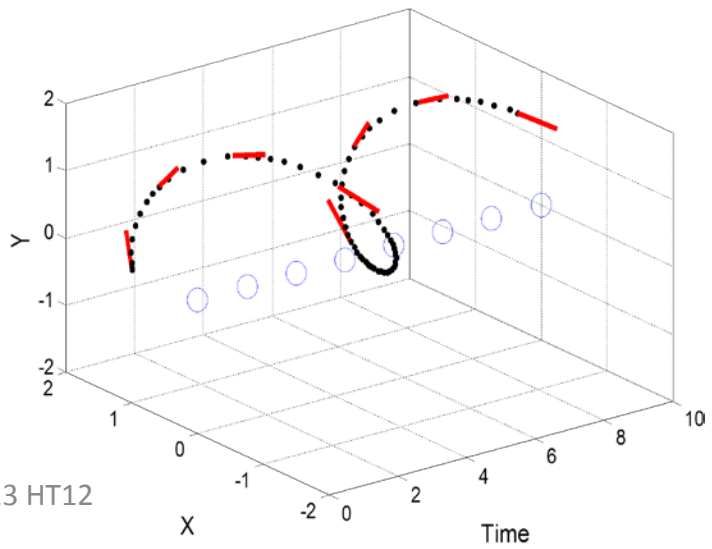
Initialvärdesproblem, V: Fasrum mm.

Mera geometrisk insikt: tillståndsvektorn som en kurva i ett n+1-dimensionellt rum: en dimension för varje komponent och en för tiden. I satellitmodellen är $n = 4$ så det är svårt, men man kan

2. Projicera på t ex ett 2-D underrum (x,y), banan som parameterkurva



3. Använda 3D (x,y,t).
   dx/dt och dy/dt
   visas som en hastighets”pil”
   (även i 2.)



Initialvärdesproblem, VI: Eulers metod

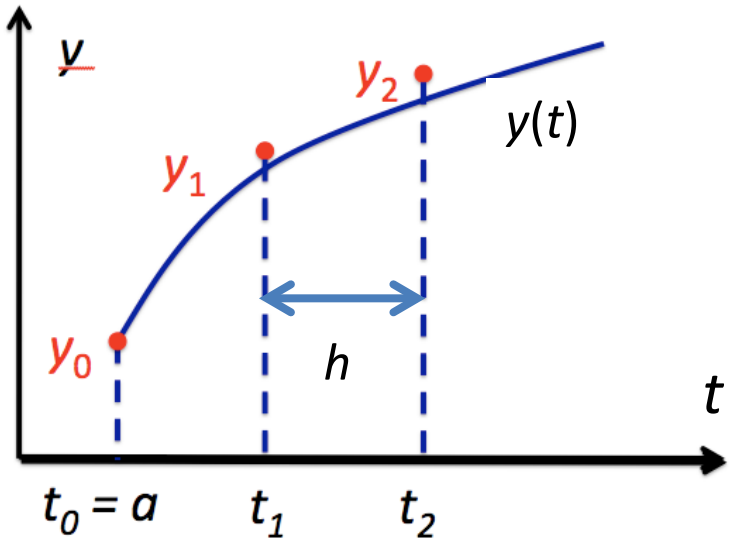
Man kan approximera lösningen med en steg-metod som ger y_k som approximation till $y(t_k)$, $t_k = 0 + kh, k=0,1,\ldots$

Approximera derivatan med differenskvot,

$$\left.\frac{dy}{dt}\right|_{t_n} \approx \frac{y(t_{n+1}) - y(t_n)}{h} \rightarrow \frac{y_{n+1} - y_n}{h}$$

så får vi ***Eulers*** metod

$$t_n = a + nh$$
$$y_{n+1} = y_n + h \cdot f(t_n, y_n), n = 0,1,\ldots$$
$$y_0 = c$$



Initialvärdesproblem, VII: Fel

$$\begin{cases} y(t_{n+1}) = y(t_n) + hy'(t_n) + \dfrac{h^2}{2} y''(\tau) \\ y_{n+1} = y_n + hf(t_n, y_n) \end{cases}$$

$$E_{n+1} = E_n + h(f(t_n, y(t_n)) - f(t_n, y_n)) + \frac{h^2}{2} y''(\tau)$$

$$|E_{n+1}| \le |E_n| + hL|E_n| + \underbrace{\frac{h^2}{2}|y''(\tau)|}_{\text{Lokalt fel}}$$

Om man tar N steg från 0 till $t_N = b$ är $h = b/N$

Totalt N fel summeras,

$$N \cdot \frac{h^2}{2} \max|y''| = b \cdot \frac{h}{2} \max|y''|$$

Fel proportionellt mot h^1 : noggrannhetsordning 1

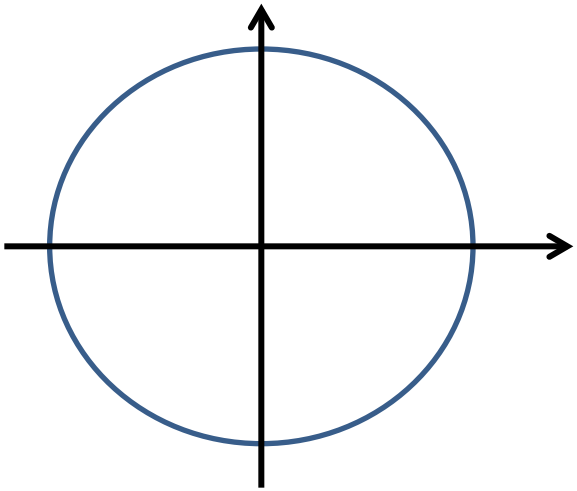
Initialvärdesproblem, VIII: Hur fungerar Euler?

Exempel: Rita en enhetscirkel! Parameterkurvan $x = \cos t$, $y = \sin t$ ger

$$dx/dt = -\sin t = -y,$$
$$dy/dt = \cos t = x$$

Eulers metod med steg $h = 2\pi/M$ ger

$$x_{k+1} = x_k - h\, y_k, \quad x_0 = 1$$
$$y_{k+1} = y_k + h\, x_k, \quad y_0 = 0$$



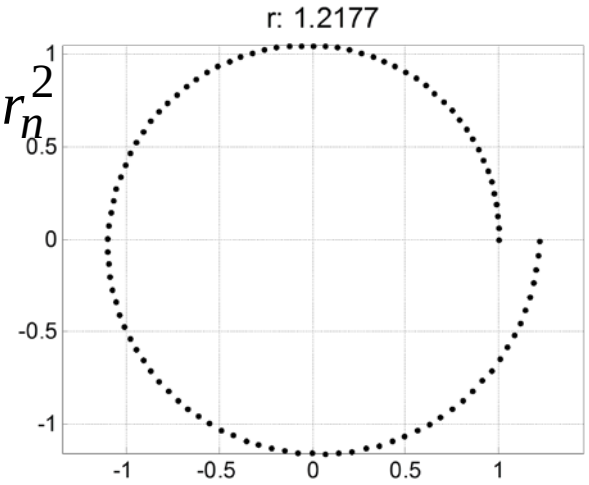
$M = 100$: *Spiral !*

$$r_{n+1}^2 = \mathbf{u}_{n+1}^T \mathbf{u}_{n+1} = \ldots = (1+h^2)\mathbf{u}_n^T \mathbf{u}_n = (1+h^2) r_n^2$$

$$r_M^2 = (1+h^2)^M = e^{M(h^2 + 1/2 h^4 + \ldots)} =$$

$$= e^{2\pi h}(1 + O(h^3)) =$$

$$= 1 + 2\pi h + O(h^2) \approx 1.4$$



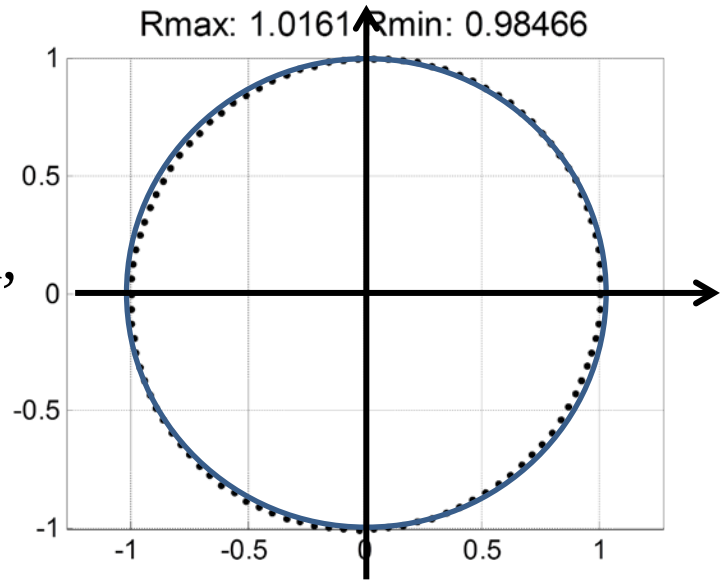
Initialvärdesproblem: Bättre metoder I

O(h) fel för stort, men spiral kan enkelt åtgärdas:

Tag senaste *x*-värde i *y*-uppdateríngen:

$$x_{k+1} = x_k - h\, y_k, \quad x_0 = 1$$
$$y_{k+1} = y_k + h\, \boldsymbol{x_{k+1}}, \quad y_0 = 0$$

Ellips med halvaxlar $\dfrac{1}{\sqrt{1 \pm (h/2)^2}}$



Detta används för Hamiltoniansk dynamik, t ex molekyldynamik, Maxwells ekvationer: Verlet's metod

$$\mathbf{M}\mathbf{v}_{n+1} = \mathbf{M}\mathbf{v}_n + h\, \mathbf{F}(\mathbf{x}_n)$$
$$\mathbf{x}_{n+1} = \mathbf{x}_n + h\, \mathbf{v}_{n+1}$$

Initialvärdesproblem: Bättre metoder II

Trapetsregeln => trapetsmetoden för IVP

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} y'(s)ds = \int_{t_n}^{t_{n+1}} f(s, y(s))ds =$$

$$= \frac{t_{n+1} - t_n}{2}\left(f(t_{n+1}, y(t_{n+1})) + f(t_n, y(t_n))\right) + O(\left|t_{n+1} - t_n\right|^3)$$

ger

$$y_{n+1} - y_n = \frac{t_{n+1} - t_n}{2}\left(f(t_{n+1}, y_{n+1}) + f(t_n, y_n)\right)$$

- Andra ordningens fel: noggrannaste *linjära enstegs*-metoden
- IMPLICIT – ekvationslösning i varje steg
- Eulers metod EXPLICIT

- Bakåt Euler: $y_{n+1} - y_n = hf(t_{n+1}, y_{n+1})$
  - IMPLICIT, ordning 1

Initialvärdesproblem: Runge-Kuttametoder I

Ersätt y_{n+1} i trapetsmetoden med sin Euler-approximation så blir metoden explicit:

$$y_{n+1} - y_n = \frac{h}{2}\left(f(t_n, y_n) + f(t_{n+1}, y_n + hf(t_n, y_n)) \right)$$

och har fel $O(h^2)$. Konstruktionen implementeras som

$$k_1 = hf(t_n, y_n)$$
$$k_2 = hf(t_n + h, y_n + k_1)$$
$$y_{n+1} = y_n + \tfrac{1}{2} k_1 + \tfrac{1}{2} k_2$$

en två-stage explicit Runge-Kuttametod
Ännu högre ordning?

Initialvärdesproblem: Runge-Kuttametoder II

- … genom att evaluera $f(.,.)$ $k > 2$ gånger (k stages) i steget
- För k-stage-metoder
  - Ordning <= k
  - Ordning k bara för k = 1,2,3,4
- Klassisk Runge-Kutta, fyra stages, ordning 4, NAM RK4

$$
\begin{aligned}
k_1 &= h\, f(t_n, y_n) \\
k_2 &= h\, f(t_n + 1/2\, h, y_n + 1/2\, k_1) \\
k_3 &= h\, f(t_n + 1/2\, h, y_n + 1/2\, k_2) \\
k_4 &= h\, f(t_n + \quad h, y_n + \quad k_3) \\
y_{n+1} &= y_n + 1/6\,(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

MATLAB **ode23** och **ode45**

Initialvärdesproblem: Styva problem I

Exempel Radioaktivt sönderfall

A sönderfaller till B som sönderfaller till C. $A \xrightarrow{k_1} B \xrightarrow{k_2} C$:

k_i kallas hastighetskoefficienter. a, b och c koncentrationerna

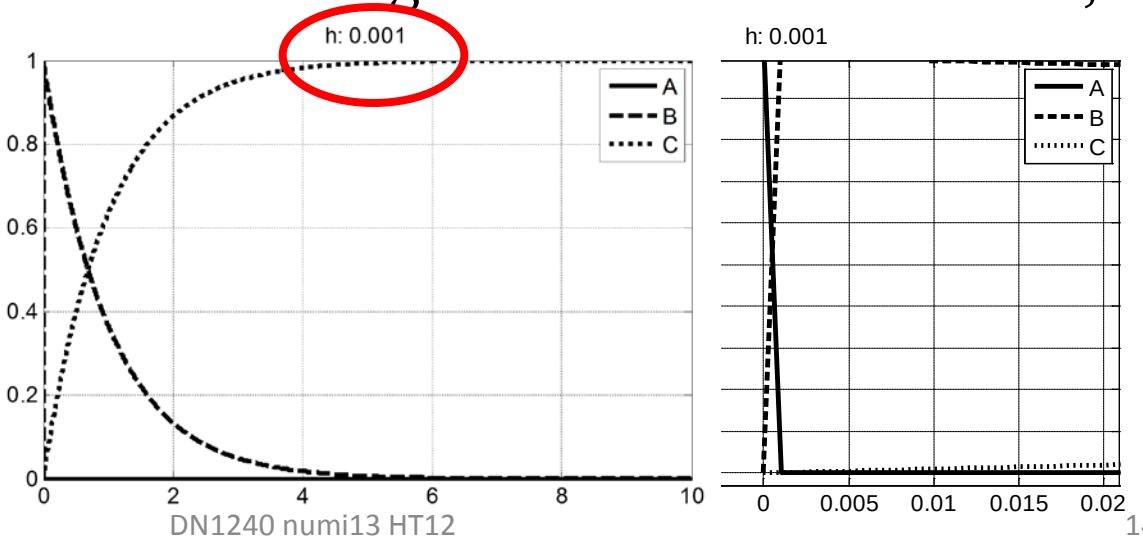
$$\begin{cases} \dot{a} = -k_1 a \\ \dot{b} = k_1 a - k_2 b \\ \dot{c} = k_2 b \end{cases} \quad \frac{d\mathbf{y}}{dt} = \mathbf{A}\mathbf{y}, \mathbf{y} = \begin{pmatrix} a \\ b \\ c \end{pmatrix}, \mathbf{A} = \begin{pmatrix} -k_1 & 0 & 0 \\ k_1 & -k_2 & 0 \\ 0 & k_2 & 0 \end{pmatrix}, \mathbf{y}(0) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

Varje A som försvinner blir ett B, och varje B som försvinner blir ett C: $a + b + c$ är konstant. Småningom försvinner all A och B, och C har samlat allt.

k_1 = 1000 (sort: 1/s),

k_2 = 1,

h = 0.001, Euler

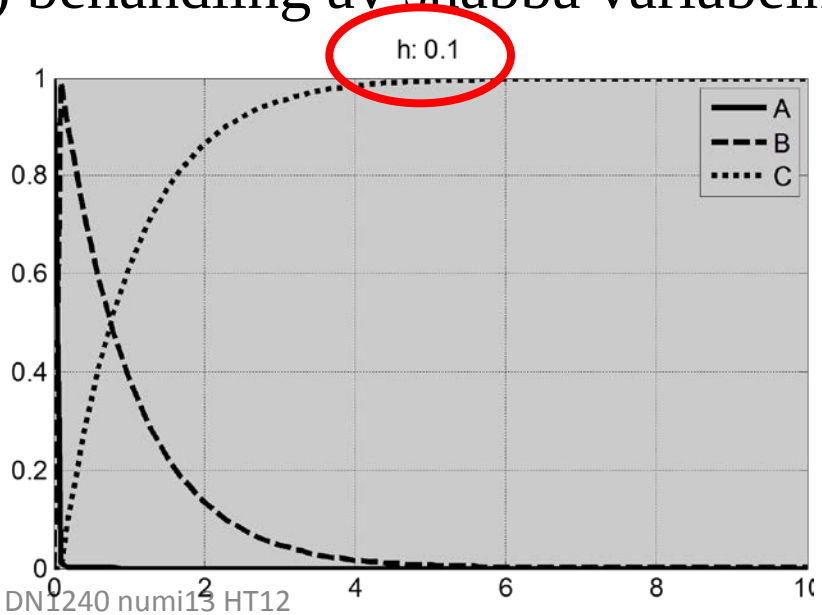


Initialvärdesproblem: Styva problem II

$$a_{n+1} = a_n - h\,k_1\,a_n = (1 - h\,k_1)a_n$$
$$b_{n+1} = b_n + h\,(k_1\,a_n - k_2\,b_n)$$
$$c_{n+1} = c_n + h\,k_2\,b_n$$

För $h > 0.002$ växer a exponentiellt, $(1 - h\,k_1) < -1$,
numerisk ***In***stabilitet. L blir 1000!
För $t > 0.005$, säg, är a nästan 0 och lösningens tidsskala är 1 sek.
Trots det krävs en tusendels sek. tidssteg: ***STYVT*** problem
Fix: Implicit (Bakåt Euler) behandling av snabba variabeln a:

$a_{n+1} = a_n - h\,k_1\,\boldsymbol{a_{n+1}}$**:**
$\boldsymbol{a_{n+1}} = a_n/(1+ hk_1)$
$b_{n+1} = b_n + h\,(k_1\,a_n - k_2\,b_n)$
$c_{n+1} = c_n + h\,k_2\,b_n$



Initialvärdesproblem: Styva problem III

I allmänhet är det inte så lätt att identifiera de snabba variablerna och då får man använda (t.ex) Bakåteuler på hela systemet. Det blir till att lösa ekvationssystemet

$$(\mathbf{I} - h\mathbf{A})\mathbf{y}_{k+1} = \mathbf{y}_k$$

i varje steg.

- Automatiskt val av steglängd viktigt

Vårt exempel: start med 0.0001, öka till säg 0.1

- Om icke-linjärt, lös

$$\mathbf{z} - h\mathbf{f}(t_{k+1},\mathbf{z}) = \mathbf{y}_k,\ \mathbf{y}_{k+1} = \mathbf{z}$$

med Newton.

- Jacobian-matrisen $d\mathbf{f}/d\mathbf{z}$ behövs.
- Bra startgissning finns t ex $\mathbf{z}^{(0)} = \mathbf{y}_k$
- Effektiv hantering av gles matris viktigt för stora system

MATLAB, I

• Många (för många?) metoder finns.
En lämplig standardmetod är
`[tout,yout] = ode45(fun,tspan,y0);`
Dormand-Princes RK4 och RK5. Evaluering med två olika metoder i varje steg användes för feluppskattning och för automatiskt val av steget *h*. **`ode23`** ordning 2 & 3
• **`dydt = fun(t,y)`** är högerledet i ekvationen, reell, komplex, skalär eller vektorform, **`tspan`** är vektor som börjar med t0 och fortsätter med de *t*-värden där lösningen önskas, **`y0`** är begynnelsevärdesvektorn: **`dydt,y0`** KOLUMNER!
• För styva problem: **`ode15(fun,tspan,y0)`**
• Högre ordnings system måste alltså först skrivas som första ordnings system. Förekommer alltid på tenta och är enkelt.

MATLAB, II

Automatisk reglering av steglängden:
Avses styras så, att ***lokala felet över varje steg*** hålls mindre än den givna toleranser: **`reltol`** och **`abstol`**, sätts med

```
opts = odeset('reltol',1e-5);
[tout,yout] = ode45(fun,tspan,y0,opts);
```

- Observera: skillnaden mellan exakt lösning och numerisk uppskattas ***INTE***.
- **`tspan`** påverkar inte steglängdsreglering annat än **`tspan`**(1) & **`tspan`**(end)
- Man förväntar sig konvergens så att för ett givet problem felet blir proportionellt mot toleransen.
    - kvoten kan inte lätt uppskattas,
    - felet som funktion av tolerans blir mindre regelbundet än vid global, successiv steghalvering.

MATLAB, III

Vi undersöker $y(4)$ beräknad med **ode23** och **ode45**

toleranser $10^{-k/2}$, $k = 1,2,\ldots,22$.

$$\frac{dy}{dx} = -10y^3 + \cos x,\ y(0) = 1$$

- Exakt resultat = strängaste toleransen.
- Observerad konvergens-ordning p (fel = Ch^p) blir

  ode23: $p = 2.7$ ---- ode23

  fel/tolerans från 1 vid $h = 0.3$ till > 100 vid $h = 0.001$.

  ode45: $p = 4.7$ ---- ode45. ode45 tar minst 10 steg.

  fel/tolerans ungefär 0.1 för alla fall.