Övning 3.

Ordinära differentalekvationer (ODE)

— kort teori

— modul 3 – game

— program (~solve) (kod på tavlan). Jobba själv ☺

— hur testar vi om vi inte förstår?

ODE: (*) $\frac{du}{dt}=f(t;u), \quad u(t_0)=u_0$

Lösning till (*)

Exempel:

$$\frac{du}{dt}=-u, \quad u(0)=1$$

$$u(t)=e^{-t}$$

Lös med hjälp av tidsstegning

Eulers metod:

$$u^{n+1}=u^n+dt\, f(t^n;u^n), \quad n=0, 1, 2, \ldots$$

Om vi vill lösa $\frac{du}{dt}=f(t;u), \quad u(t_0)=u_{0,} \quad t_0 \le t \le T$

n = 0, 1, 2, …, N

N?

Tidssteg dt ger N:

$$N=\frac{T-t_0}{dt}$$

System av ODE:er: (Jämför övning 1)

$$(T)\begin{cases}\dfrac{dv}{dt}=-\cos(t) & v(0)=0\\ \dfrac{dx}{dt}=v(t) & x(0)=0\end{cases}$$

$$\vec{u}=\begin{bmatrix}u_1(t)\\u_2(t)\end{bmatrix},\quad \vec{f}=\begin{bmatrix}f_1(t;u_1;u_2)\\f_2(t;u_1;u_2)\end{bmatrix}\qquad \left(\vec{u}=\coprod_{\forall u}u(t),\quad \vec{f}=\coprod_{\forall u}\left(t;\overbrace{\coprod_{\forall u}u}^{\vec{u}}\right)\right)$$

$$\left.\begin{array}{l}\vec{u}=\begin{bmatrix}v(t)\\x(t)\end{bmatrix},\quad \vec{f}=\begin{bmatrix}-\cos(t)\\v(t)\end{bmatrix}\\[2ex] \dfrac{d\vec{u}}{dt}=\vec{f},\quad \vec{u}(0)=\begin{bmatrix}0\\0\end{bmatrix}\end{array}\right\}\triangleq (**)$$

($\triangleq$ användes på tavlan)

$$\frac{d\vec{u}}{dt}=\vec{f}(t;\vec{u}),\quad \vec{u}(t_0)=\vec{u}^0$$

Bra (!) nummeriska metoder (tidsstegning) fungerar på detta sätt!

Eulers metod

$$\vec{u}^{n+1}=\vec{u}^n+dt\,\vec{f}(t^n;\vec{u}^n)$$

Egenskaper hos olika metoder:

$$\frac{du}{dt}=f(t;u),\quad t_0\le t\le T$$

Euler framåt: $u^{n+1}=u^n+dt\,f(t_n;u^n)$

Noggrannhetsordning: 1 $\quad |\underbrace{u(T)}_{dt}-\underbrace{\tilde{u}(T)}_{dt/2}|\approx k\cdot dt^1$

k — konstant

felet avtar linjärt med dt

Explicit metod (givet u^n kan vi beräkna u^{n+1} direkt!)

Ej energibevarande: (varför är utanför kursen)

energin stiger.

Euler bakåt: $u^{n+1} = u^n + dt\, f\left(t_{n+1};\, u^{n+1}\right)$

Noggrannhetsordning: 1

Implicit metod (givet u^n kan ej beräkna u^{n+1} direkt, extra åtgärder krävs)

Ej energibevarande:
energin avtar.

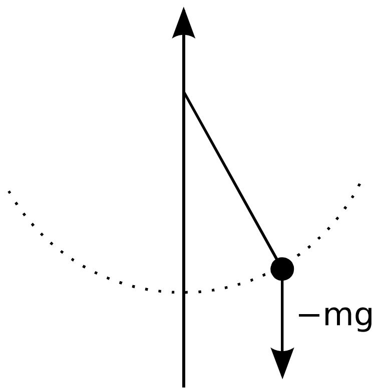
Trapetsmetoden: $u^{n+1} = u^n + \frac{dt}{2}\left[f\left(t_n;\, u^n\right) + f\left(t_{n+1};\, u^{n+1}\right)\right]$

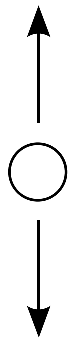
Implicit metod

Energibevarande

Noggrannhetsordning: 2 $\left|u(T) - \tilde{u}(T)\right| \approx k \cdot dt^2$

$$(T)\begin{cases} \frac{dx}{dt} = v(t) \\ \frac{dv}{dt} = -x(t) \end{cases} \left| \begin{array}{l} x(0)=0 \\ v(0)=1 \end{array} \right.$$





Harmonisk oscillation:

Svänger för evig,
energin är bevarad.

Totala energin i systemet

$$E(t) = \frac{v(t)^2}{2} + \frac{x(t)^2}{2}$$

kvadraterna och halveringarna spelar inge roll
när man analyserar om enegin bevaras eller ej.

ode_solve.py

Kommer läggas upp på Katarinas nada-sida
http://www.nada.kth.se/~katarina/

```
from pylab import *
import ode

def f(t, u):

     #Tidsintervall
     I = [0, 2]
     t0 = I[0]
     T = I[1]
     dt = 1 #Steglängd

     #Antal stag
     N = (T - d0) / dt + 1

     #Vektor med tidspunkter, (t⁰, t¹, …, tᴺ)
     tarray = linespace(t0, T, N) #[0 1 2]

     #Begynnelsedata
     u0 = 1 #Ses som array av storleken 1

     #Definiera en array för lösningen
     uarray = zeros([size(u0), size(tarray)])    #[0 0 0]

     uarray[:, 0] = u0  #kolonn 0 för alla rader i uarray = u0

     un = u0
     i = 1
     for tn in tarray[0, N - 1]:
          u = ode.timestep(f, tn, un, dt, "Euler")
          uarray[:, i] = u   #[1 u¹ 0]
          i += 1
          un = u   #Uppdatera un

     plot(tarray, uarray[0])
```