Teknillinen korkeakoulu Apiola
Matematiikka

**Mat-1.192 Numeerinen ja symbolinen laskenta kevät 2004**
`http://www.math.hut.fi/teaching/numsym/04/H/`

**Laskuharjoitus 6** (viikko 13–14 , 23 – 30.3 2004)

Päivitetty 26.3.04 (BeulerFixP korjattu)

**Huom:** Tehtävään 5 lisäiltiin hiukan.

Nämä tehtävät käydään läpi **ti 30.3.**

Annetaan vielä tehtävät 7 ennen pääsiäistä, sitten onkin loppuprojektien vuoro.

Sopivia harj7-tehtäviä ovat mm. Moler: Ch 11: PDE: teht. 11.1, 11.2, 11.6

`http://www.math.hut.fi/teaching/numsym/04/maple/ns04.mpl` Maple-kokoelma
`http://www.math.hut.fi/teaching/numsym/04/matlab/dfield5.m`
LAODE-funktio (muistathan vielä `pplane5`:n)

1. Aiheena kankeat diffyhtälöt. Jatketaan harj.5 teht. 3:a. Yhtälöhän on

$$y' = -1000(y - \sin t) + \cos t.$$

   Jotta voitaisiin paremmin havainnollistaa kuvilla, lievennetään kankeutta ottamalla vakion 1000 tilalle vaikkapa 50.

   Muodosta Maplella yleinen ratkaisu antamalla alkuehto muodossa $y(a) = ya$.

   Piirrä ratkaisufunktioita vaikka ohjeen mukaan.

   Määrittele sopiva grafiikkafunktio ohjeen tyylisesti ja piirtele sillä joitakin kuvia ja rakenna sen avulla ratkaisukäyräparvi. (Piirrä vaikka alapuoli sinisellä ja yläpuoli punaisella.)

   Tee suunakenttä- ja ratkaisuparvipiirros Matlab-(LAODE)-funktiolla `dfield5`.

2. (a) Ratkaise edellinen yhtälö Euerin menetelmällä välillä $[0, \pi]$. Tiedostossa `.../maple/ns04.mpl` on `Eulerh`, listaus `harj5`-paperissa. Kokeile erityisesti askelpituuksia $h_{max}$, $h_{max}/2$ ja esim. $1.2hmax$.

   (b) Tutustu `../maple/ns04.mpl`-tiedoston funktioon `BeulerFixP`, selvitä itsellesi sen toiminta ja sovella sitä samaan tehtävään. Suppenemista voit tarkkailla vaikka tyyliin:
   `hmax:=...; BEF:=BeulerFixP(F,[0,Pi],1,0.9*hmax,5): %[-1];`
   Kun saat järkevää, voit piirtää: `plot(BEF);`

   **Varoitus!** Edellä olevan tyylisissä kutsuissa on syytä varmistaa, että kaikilla parametreilla on järkevät arvot. Jos vaikka F:n määrittely unohtuu, niin Maple ryhtyy ikuiseen jauhantaan, jolloin ollaan täysin **stop**-nappulan armoilla.

   (c) Selitä tehtävien 1) ja 2) kuvien perusteella, mistä tässä on kyse!

3. Kirjoita Maple-funktio `BeulerNewt`, johon on hyvä pohja ja ohjeet olemassa.

   Tee ensin versio, jossa on `niter` parametrinä aivan kuten "Fix":ssä.

   Voit terästää funktiosta "aidon implisiittimenetelmän" iteroimalla toleranssiin saakka. Lopetusehtona vosi olla esim.

$$|y_{n+1}^{(k+1)} - y_{n+1}^{(k)}| \leq 10^{-6} |y_{n+1}^{(k+1)}|.$$

   (jos suht. toleranssina on $10^{-6}$.) Olkoon terästys "vapaaehtoista", "niter-versiokin" toimii varsin hyvin.

   Testaa tätä menetelmää eri askelpituuksilla.

   Kirjoita vaikka grafiikkafunktio:

   `BenKuva:=h->plot(BeulerNewt(F,[0,Pi],1,h,5))` (jos käytät "niter-versiota")

   Selvitä taas. mistä on kyse.

4. Tarkastele lineaarista yhtälösysteemiä $y' = Ay$, missä $A = \begin{bmatrix} 998 & 1998 \\ -999 & -1999 \end{bmatrix}$

   Alkuarvotehtävän ratkaisu voidaan kirjoittaa muotoon $y(t) = e^{At}y(0)$, joka Maplella voidaan tehdä alla kerrattuun tyyliin.

   Tässä on havainnollisempaa laskea ratkaisu ominaisarvojen avulla ominaisvektorikannassa:
   $y(t) = C_1 e^{\lambda_1 t} v_1 + C_2 e^{\lambda_2 t} v_2$.

   Selvitä tämän muodon perusteella, mistä systeemin kankeus johtuu.

   Laske ja piirrä (faasikuvana) jokin Euler-askel AA-tehtävälle, $y(0) = [1, 0]^T$. Tämän voit tehdä joko Maplella tai Matlabilla.

Maple-ratkaisuun voit etsiä valmiin `EulerV`:n `..maple/ns04.mpl`:stä. (Ei ole pahitteeksi, vaikka kirjoittaisit koodin yhdenmukaisesti edellä esiintyneiden Euler-versioiden kanssa.) Mielellään voit piirtää myös faasikuvan `DEplot`:lla

Matlab-ratkaisussa voit piirtää faasitasokuvan `pplane5`:llä ja samaan kuvaan joitakin Eulerin polkuja. ( Kokeile `figure(1); figure(2) ...` sitten `hold on` ja piirtoa rohkeasti.
Kuvissa voit taas skaalata siten, että arvon 1000 sijasta käytät vaikka arvoa 50.

Jos kokeilet arvoa 1000 pplane5-piirroksessa, niin kankeuden pitäisi näkyä hitautena ratkaisukäyrien piirrossa, koska pplane5 käyttää ei-kankeaa ratkaisijaa ode45.

5. Tutki kolmen aineen kemiallisia reaktioita kuvaavaa yhtälösysteemiä:

$$\begin{cases} y_1' = -0.04y_1 + 10^4 y_2 y_3 \\ y_2' = 0.04y_1 - 10^4 y_2 y_3 - 3\times 10^7 y_2 \\ y_3' = 3\times 10^7 y_2^2 \end{cases}$$

Muodosta Maplella systeemin Jakobiaani y-muuttujien suhteen ja määritä sen ominaisarvot. Päättele niiden perusteella systeemi kankeaksi (ainakin alla olevassa alkuarvopisteessä).

Ratkaise AA-tehtävä, jossa alkuarvona on vaikkapa $[1, 0, 0]^T$, välillä $[0, 3]$. Vertaa ratkaisijoita `ode45` ja `ode15s`. Tutki askelten lukumääriä, havainnollista piirroksin, voit käyttää myös optiota: `odeset('Stats','on')`.

(Jos tutkiskelet jaettuja materiaaleja, voit tehdä hyviä löytöjä.)

**Huom!** ke 24.3. tein pieniä lisäyksiä tähän tehtävään: Ota Maplella generoitu jakobiaani Matlab-editoriin ja tee siitä Matlab-funktio vaikkapa "jakobiaani". Suorita `optiot=odeset('Jacobian',@jakobiaani)`.

Jakobiaania ei tarvita ode45:ssä, mutta ei se haittakaan. Sensijaan saattaa käydä niin, että ode45 laskee toivottoman kauan. Jotta lasekennasta saataisiin jotain tietoa (ennen `CTR-C`:tä), kannattaa laittaa globaali muuttuja, vaikkapa `kutsut` diffyhtälösysteemiin laskuriksi.

```
function yp=kemikaalit(t,y)
global kutsut
kutsut=kutsut+1
...
```

Istunnossa (pääohjelmassa) ennen ode45-kutsua kirjoitetaan `global kutsut` ja `kutsut=0`.

## Ohjeita

Yleisesti ottaen kannattaa palautella mieliin harj2:n ja harj3:n tehtäviä ja ohjeita.

Teht. 1: Diffyhtälön analyyttinen ratkaisuhan saadaan tyyliin
`dsolve({dyht,y(0)=ya},y(t));`
Ratkaisulausekkeen poimiminen on kaikkein yksinkertaisinta näin:
`Y:=rhs(%);` Funktioksi muuttaminen: `Yf:=unapply(Y,t);`

"Mielenkiintoinen"lisäpiirre tulee tässä, kun Maple haluaa lausua tuloksen cosh :n ja sinh :n avulla. Nimittäjän $\cosh(kt) - \sinh(kt)$ aiheuttaa nollalla jaon pienilläkin $t$:n arvoilla ainakin, jos $k = 1000$.

Kannattaa leikata/liimata nimittäjä ja sanoa sille: `nim:=convert(nim,exp);`. Outputista saa helposti leikkaus/liimaus-tekniikalla input-muodon, jota sitten sopivasti editoidaan. (Varmasti helpompaa kuin sopivan `subs`-komennon miettiminen.) Pieni nimittäjä poistuu (itse asiassa nimittäjään ei jää mitään).

Kankeusasian ymmärtämiseksi kannattaa ensin piirtää ratkaisukäyräparvi analyyttistä ratkaisua käyttäen. Tähän tyyliin voisit edetä:

```
plot(subs(a=0.10,ya=1,Y),t=0..0.5,y=0..2);
plot(subs(a=0.10,ya=1,Y),t=0..Pi,y=-1.5..1.5);
```

Muistellaan tehokasta Maple-tekniikkaa sopivia grafiikka-arvoisia ("kertakäyttö")funktioita määritellen. (Funktion määrittelyä editoidaan aina kulloisenkin modifikaatiotarpeen mukaan.)

```
kuvaf:=(A,Tvali,Ya,vari)->plot(subs(a=A,ya=Ya,Y),t=Tvali[1]..Tvali[2],
y=-1.5..1.5,color=vari);

display(seq(kuvaf(A,[0,Pi],1,red),A=[seq(k*0.1,k=0..30)]));
... blue ...
display(%,%%);
```

Kuvajonosta saadaan animaatio lisäämällä display-funktion valitsimeksi `insequence=true`.

## Implisiittinen Euler

$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1})$.

Ratkaistava $y = y_{n+1}$ yhtälöstä $F(y) = y$, missä $F(y) = y_n + hf(t_{n+1}, y)$.

Tehtävä on suoraan kiintopisteiteraatiomuotoa (kts. [HAM] ss. 66–68.)

Menetelmän Maple-implementointi on nautinnollista, määritellään lokaali iterointifunktio $F$. Kiintopisteiteraation suppenemisehto on $|F'(x)| < 1$.

Tässä ote tiedostosta `.../maple/ns04.mpl` (päivitetty 22.3.04) Kommenteissa on selitys menetelmän rajoituksesta, joka johtuu yllä mainitusta suppenemisehdosta.

---

*Backward Euler, eli implisiittinen Euler. Tässä iteroidaan kiintopisteiteraatiolla. Suppeneminen vaatii askelpituusrajoituksen $|hf_y| < 1$ Käytännössä aika kelvoton, eikä omaa implisiittisen menetelmän pääetua: askelpituusrajoituksesta vapautumista. Helppo impelemntoida.*

---

```
 BeulerFixP:=proc(f,vali,y0,h,niter)
local a,b,T,Y,F,s,n,N;
F:=unapply(Y[n]+h*f(T[n+1],s),s);
a:=evalf(vali[1]); b:=evalf(vali[2]);
N:=ceil((b-a)/h);
Y[0]:=evalf(y0);
T[0]:=a;
for n from 0 to N do
  T[n+1]:=T[n]+h;
  Y[n+1]:= (F@@niter)(Y[n]+h*f(T[n],Y[n]));
end do;
[seq([T[n],Y[n]],n=0..N)];
end:
```

Implisiittisestä Eulerista on juttua [HAM] ss. 123 – 124.

**Iterointi Newtonin menetelmällä**

Newtonin menetelmä ei kärsi yllä olevasta rajoituksesta, ainoa pieni epämääräisyys on, että alkuarvauksen on oltava "riittävän hyvä", jotta menetelmä suppenee, jolloin se yleensä suppenee nopeasti. "Oikea" implisiittimenetelmä saadaan iteroimalla siihen saakka, kunnes toleranssiraja saavutetaan.

4) Newtonia käyttävä implisiittinen Euler saadaan kiintopisteEulerista helposti. [HAM]-versiossa (`BEN`) (joka on myös `ns04.mpl`:ssä), on turhaan viety paikallisten funktioiden määrittelyt for-silmukan sisään. Tee yllä olevan `BeulerFixP`:n henkinen versio.

Systeemin $y' = Ay$ ratkaisu: $y = e^{At}y_0$:

```
> with(LinearAlgebra):with(linalg):
> A:=<<a11 | a12>,<a21 | a22>>;
> eAt:=Matrix(exponential(A,t));
```

Huomaa, että `exponential` on `linalg-` funktio, jos/kun haluamme käsitellä tulosta `LinearAlgebra`- rakenteena, täytyy soveltaa `Matrix`:ia. (Ääneen lukien tuo kuullostaa juuri siltä, kuin pitääkin.)

Tässä on erityisen hyödyllistä käyttää `Eigenvectors`- komentoa, jolloin saadaan johtopäätöksiin parhaiten sopiva muoto (esitys ominaisvektorikannassa, kun sellainen on tarjolla).

5) Jakobiaanin muodostamiseen tarvitaan vanhaa linalg:ia, siis `with(linalg):` `?jacobian`. `lprint` antaa Matlabiin siirrettävän muodon. Kätevää on ensin muuntaa matriisi listojen listaksi: `convert(J,listlist);`
Jos leikkau/liimaus ei onnistu, voit vaikka komentaa:

```
> writeto("jac.txt"):
> lprint(...);
> writeto(terminal):
```