Matemaattiset ohjelmistot MS-E1999

Tavalliset differentiaaliyhtälöt

1 Yleisiä differentiaaliyhtälökäsitteitä

Differentiaaliyhtälö tarkoittaa sellaista yhtälöä, jossa esiintyy tuntematon funktio ja sen derivaattoja. Tässä käsittelemme vain *tavallisia differentiaaliyhtälöitä* ja -systeemejä. Ratkaistavat funktiot ovat yhden muuttujan funktioita.

Yhtälösysteemin *kertaluku* on korkeimman siinä esiintyvän derivaatan kertaluku.

Alkuarvotehtävässä on annettu ratkaisufunktion ja sen derivaattojen arvot samassa pisteessä (ajanhetkellä t_0). Systeemin tapauksessa on annettu ratkaistavien funktioiden arvot ao. alkupisteessä.

Reuna-arvotehtävässä on annettu funktion tai derivaatan arvoja eri pisteissä.

Tiettyjen säännöllisyysehtojen vallitessa voidaan valitussa alkupisteessä antaa yhtä monta ehtoa kuin yhtälön kertaluku osoittaa ja saadaan jossakin alkupisteen ympäristössä yksikäsitteinen alkuarvotehtävän ratkaisu.

:n differentiaaliyhtälötaidot tiivistyvät komentoon **dsolve**. Lisäksi kirjastopakkauksessa `DEtools` on erityisesti funktio `DEsolve`, jolla voidaan piirtää suuntakenttiä, approksimaatioita eri pisteisiin asetetuille ratkaisukäyrille jne. :n avustusjärjestelmässä on runsaasti perustietoa differentiaaliyhtälöistä. Esimerkkejä sopivista avustuskomennoista tai valintareiteistä ovat

```
?ode, ?odeadvisor, ?dsolve
```

`Mathematics->Differential Equations-> Classifying ODE->Overview`

Käsittelemme esimerkkien valossa :n differentiaaliyhtälöratkaisijoiden käyttöä, ratkaisujen jälkiprosessointia, kuten piirtämistä ja taulukointia.

Tarkastelemme kolmea eri näkökulmaa, analyyttistä (symbolista), numeerista ja kvalitatiivista käsittelytekniikkaa. Näihin kaikkiin on :ssa työkaluja tarjolla. Ratkaisujen jatkokäsittelytekniikka on periaatteessa hyvin samanlaista kuin tavallisten yhtälöiden kohdalla. Kannattaa siksi palauttaa mieleen ja tehdä vertailuja lukuun **??** erityisesti kohtaan **??** .

2 Analyyttinen ratkaiseminen, komento `dsolve`

Komennon `dsolve` avustusteksti antaa neljä muotoa, joista esimerkeissämme käsittelemme näitä kolmea useimmiten vielä niin, että osa `extra_args` puuttuu. dsolve

```
dsolve(ODE,y(x),extra_args);
dsolve({ODE,ICs},y(x),extra_args);
dsolve({sysODE,ICs},{funcs},extra_args);
```

Parametri `ODE` selviää esimerkeistä. `ICs` tarkoittaa alkuehtoja, ne annetaan jonona muodossa `y(t0)=y0, D(y)(t0)=dy0, (D@@2)(t0)=d2y0, ...;`

Usein on hyödyllistä ja opettavaista seurata ratkaisuprosessin etenemistä. Sitä varten voidaan asettaa `infolevel[dsolve]:=3:` infolevel

Otamme ensimmäiseksi esimerkiksi toisen kertaluvun lineaarisen vakiokertoimisen yhtälön, jonka varmasti osaisimme ratkaista käsin.

$$y'' - y' - 2y = x$$

Palautamme mieleen, että ratkaisulla tarkoitetaan kahdesti jatkuvasti derivoituvaa funktiota $x \to y(x)$, joka jollain reaaliakselin välillä toteuttaa identtisesti yhtälön.

$$y''(x) - y'(x) - 2y(x) = x$$

Muuttujaa x nimitämme riippumattomaksi muuttujaksi ja merkitsemme sitä usein x:n sijasta t:llä.

Määritämme tehtävälle yleisen ratkaisun :n **dsolve**-komennolla dsolve ja sijoitamme saadun ratkaisun takaisin yhtälöön.

[t]14cm

```
> dyht:=diff(y(x),x,x)-diff(y(x),x)-2*y(x)=x;
```

$$dyht := \frac{d^2}{dx^2}y(x) - \frac{d}{dx}y(x) - 2\,y(x) = x$$

```
> ratk:=dsolve(dyht,y(x));
```

$$ratk := y(x) = 1/4 - 1/2\,x + \_C1\,e^{-x} + \_C2\,e^{2\,x}$$

```
> eval(subs(ratk,dyht));
                                x = x
```

- Differentiaaliyhtälö esitetään kirjoittamalla -syntaksin mukaisesti yhtälö jälkimmäisessä muodossa, siis niin, että riippumaton muuttuja on kaikkialla mukana.
- Komento `dsolve` palauttaa ratkaisun yhtälömuodossa aivan samoin kuin `solve`. Kun emme sisällyttäneet alkuehtoja, saamme kaksi mielivaltaista vakiota, joille antaa alaviivaalkuiset nimet `_C1, _C2` .
- Muuttuja `ratk` edustaa sijoitussääntöä, joka voidaan suoraan antaa `subs`-komennolle argumentiksi. Huomaa, että differentiaaliyhtälössä esiintyvät derivoinnit saadaan suoritetuksi `eval`-komennon avulla.

file=ode1.ps, height=5.5cm, width=10cm

Kuva 1: Ratkaisukäyräparvi

2.0.1 Ratkaisun käsittely lausekkeena

Haluamme kenties piirtää ratkaisukäyräparven. Silloin riittää ottaa ratkaisu*lauseke* käsittelyyn. Tässä yksinkertaisessa tapauksessa voimme tietysti leikata ja liimata työarkilla. Yleispätevä tyyli on jälleen seuraavanlainen (vrt. yhtälöiden käsittelyyn)

```
> Ylaus:=subs(ratk,y(x));
```

$$Ylaus := 1/4 - 1/2\,x + \_C1\,e^{-x} + \_C2\,e^{2\,x}$$

Yhden yhtälön tapauksessa voimme tietysti turvallisesti käyttää myös tyyliä

```
> Ylaus:=rhs(ratk):
```

Kuvan **??** ratkaisukäyräparvi syntyy kätevästi kahdella sisäkkäisellä `seq`- komennolla.

```
> plot([seq(seq(Ylaus,_C1=-2..2),_C2=-1..1)],x=-3..2,y=-10..10);
```

2.0.2 Alkuarvot, ratkaisufunktio

Ensi töiksemme määrittelemme ratkaisun funktioksi. Muistamme, että lausekkeesta saadaan funktio `unapply`-komennolla.

```
> Y:=unapply(Ylaus,x);
```

$$Y := x \mapsto 1/4 - 1/2\,x + \_C1\,e^{-x} + \_C2\,e^{2\,x}$$

Jos haluamme alkuehdot $y(0) = -1,\ \ y'(0) = 1$ toteuttavan ratkaisun, voimme käyttää `solve`-komentoa.

```
> C12:=solve({Y(0)=-1,D(Y)(0)=1},{_C1,_C2});
                    C12 := {_C1 = 1/12, _C2 = -4/3}
> Ylaus:=subs(C12,Ylaus);
```

$$Ylaus := 1/4 - 1/2\,x + 1/12\,e^{2\,x} - 4/3\,e^{-x}$$

```
> Y:=subs(C12,op(Y));
```

$$Y := x \mapsto 1/4 - 1/2\,x + 1/12\,e^{2\,x} - 4/3\,e^{-x}$$

Huomaa muoto `subs(C12,op(Y))`, jolle kehitimme sivulla **??** selkeän ajattelutavan. Jos kirjoitetaan `Y;` niin vastaa `Y`, jolloin `subs`-komento kohtelee sitä pelkkänä symbolina `Y`. Jos kirjoitetaan `op(Y)`, nähdään `Y`:n sisältö, johon haluammekin `subs`- komennon kohdistaa.

Nyt on helppoa laskea ratkaisufunktion ja derivaatan arvoja halutuissa pisteissä ja taulukoida.

```
> Digits:=3:
> h:=0.1:xlista:=[seq(i*h,i=0..5)]:
> derivY:=D(Y); # Derivaattafunktio
> taulukko:=seq([x,Y(x),derivY(x)],x=xlista):
> matrix([taulukko]);
```

$$\begin{bmatrix} 0 & -1 & 1 \\ 0.1 & -0.908 & 0.913 \\ 0.2 & -0.816 & 0.838 \\ 0.3 & -0.736 & 0.791 \\ 0.4 & -0.657 & 0.765 \\ 0.5 & -0.582 & 0.762 \end{bmatrix}$$

2.1 Alkuarvot suoraan dsolve-komennossa

Jos haluamme ratkaista tietyn alkuarvotehtävän, emmekä ole kiinnostuneita yleisestä ratkaisusta, on helpointa antaa alkuehdot suoraan `dsolve`-komennolle. Edellinen tehtävä hoidettaisiin näin.

```
> aaratk:=dsolve({dyht,y(0)=-1,D(y)(0)=1},y(x));
```

$$aaratk := y(x) = 1/4 - 1/2\,x - 4/3\,e^{-x} + 1/12\,e^{2\,x}$$

alkuehdot Tässä kannattaa huomata derivaattaehdon antamistapa derivoimisoperaattorin `D` avulla. Jos käsittelisimme korkeamman kertaluvun yhtälöä, voisimme hyödyntää derivaattaoperaattorin iterointia, eli toista derivaattaa koskeva alkuehto $y''(0) = 3$ voitaisiin kirjoittaa `(D@@2)(y)(0)=3` jne.

Tämä ja muita differentiaaliyhtälöihin liittyviä -istuntoja on nähtävillä ja saatavilla kotisivumme `http://www.math.hut.fi/~apiola/maple/opas/` kohdasta *differentiaaliyhtälöitä.*

2.2 Differentiaaliyhtälösysteemit

Komento `dsolve` ratkaisee yhtä hyvin differentiaaliyhtälösysteemejä.

Ensimmäisen kertaluvun DY-systeemi on muotoa

$$\begin{cases} y_1' = f_1(t, y_1, \ldots, y_n) \\ \vdots \\ y_n' = f_n(t, y_1, \ldots, y_n) \end{cases}$$

Jokainen n:nnen kertaluvun differentiaaliyhtälö voidaan palauttaa 1. kl:n systeemiksi ottamalla y ja sen derivaatat tuntemattomiksi funktioiksi.

Niinpä koko differentiaaliyhtälöiden teoria palautuu ensimmäisen kertaluvun systeemien tutkimiseen. Numeeriset ohjemistot vaativat useimmiten käyttäjää kirjoittamaan ja koodaamaan differentiaaliyhtälönsä ensimmäisen kertaluvun systeemiksi. :n numeeriset ratkaisijat eivät sitä vaadi.

Otamme esimerkiksi jälleen helpon lineaarisen systeemin

$$\begin{cases} x' + x + 2y = 0 \\ y' + 3x + 2y = 0 \end{cases}$$

Yleinen ratkaisu saadaan komennolla

```
> dsolve({diff(x(t),t)+x(t)+2*y(t) = 0,
        3*x(t)+diff(y(t),t)+2*y(t) = 0},{x(t),y(t)});
```

ja alkuehdot $x(0) = 3, \;\; y(0) = 2$ toteuttavat ratkaisut suoraan komennolla

```
> aaratk:=dsolve({dy1,dy2,x(0)=3,y(0)=2},{x(t),y(t)});
```

Tässä on selkeyden vuoksi otettu nimet yhtälöille ja asetettu siis ensin:

```
> dy1:=diff(x(t),t)+x(t)+2*y(t) = 0;
> dy2:=3*x(t)+diff(y(t),t)+2*y(t) = 0;
```

Koko istunto on nähtävissä (kirjan kotisivulla näkyvässä) viitteessä

`http://www.math.hut.fi/teaching/k3/luentomateriaali/lindys1.html`

Komento `dsolve` toimii kaikenlaisille yhtälösysteemeille, joissa saa esiintyä korkeammankin kertaluvun yhtälöitä.

2.2.1 Milloin voidaan ratkaista analyyttisesti

Puhumme *symbolisesta*, *analyyttisesta* tai *suljetussa muodossa olevasta* ratkaisusta, jos se voidaan esittää tunnettujen alkeisfunktioiden avulla. :n `dsolve`-komento sisältää suuren joukon menetelmiä, joilla se yrittää löytää analyyttisen ratkaisun.

Käytännössä vain harvoin saadaan ratkaisu ns. suljetussa muodossa. Tosin tämä käsite on viime aikoina laajentunut käsittämään tavanomaisten alkeisfunktioden lisäksi joukon matemaattisen fysiikan erikoisfunktioita.

tarjoaa kiintoisan mahdollisuuden ratkaisuyritysten seurantaan niin `dsolve`- kuin minkä tahansa muunkin komennon kohdalla. Tämä tapahtuu komennolla `infolevel[komento]:=n:` missä n on kokonaisluku. Mitä suurempi arvo n:llä on (tiettyyn rajaan saakka), sitä enemmän kertoo prosessin vaiheista

Katsomme muutamaa esimerkkiä.

Annamme ensin oikein helpon tehtävän.

```
> infolevel[dsolve]:=3;
> dsolve(diff(y(t),t)+sin(y(t))=t,y(t));
Methods for first order ODEs:
trying to isolate the derivative dy/dx
successful isolation of dy/dx
 -> trying classification methods
trying a quadrature
trying 1st order linear
1st order linear successful
```

$$y(t) = t - 1 + e^{-t}\_C1$$

Vaikeutetaan vähän, otetaan epälineaarisuutta.

```
> dsolve(diff(y(t),t)+y(t)^2=t,y(t));
Methods for first order ODEs:
trying to isolate the derivative dy/dx
successful isolation of dy/dx
 -> trying classification methods
trying a quadrature
trying 1st order linear
trying Bernoulli
trying separable
trying inverse linear
trying simple symmetries for implicit equations
trying homogeneous types:
trying Chini
trying exact
 -> trying 2nd set of classification methods
```

```
trying Riccati
trying Riccati Special
Riccati Special successful
```

$$y(t) = \frac{\_C1\ AiryAi(1,t) + AiryBi(1,t)}{\_C1\ AiryAi(t) + AiryBi(t)}$$

Ratkaisu voitiin lausua erikoisfunktioiden avulla.

Tehdään nyt vielä pahemmin epälineaariseksi.

```
> dsolve(diff(y(t),t)+sin(y(t))=0,y(t));
Methods for first order ODEs:
trying to isolate the derivative dy/dx
successful isolation of dy/dx
 -> trying classification methods
trying a quadrature
trying 1st order linear
trying Bernoulli
trying separable
separable successful
```

$$y(t) = \arctan(2\,\frac{e^{-t-\_C1}}{1+e^{-2\,t-2\,\_C1}}, \frac{-e^{-2\,t-2\,\_C1}+1}{1+e^{-2\,t-2\,\_C1}})$$

Muuttujat voitiin erottaa. Lisätään vielä alkuperäinen epähomogeenisuustermi.

```
> dsolve(diff(y(t),t)+sin(y(t))=t,y(t));
Methods for first order ODEs:
trying to isolate the derivative dy/dx
successful isolation of dy/dx
 -> trying classification methods
trying a quadrature
trying 1st order linear
trying Bernoulli
trying separable
trying inverse linear
trying simple symmetries for implicit equations
trying homogeneous types:
trying Chini
trying exact
 -> trying 2nd set of classification methods
trying Riccati
trying Abel
 -> trying Lie symmetry methods
```

palauttaa pelkkää tyhjää sen merkiksi, ettei mikään menetelmä johtanut tulokseen.

Kun haluamme luopua selityksista, sijoitamme `infolevel[dsolve]:=0:`

file=dirfield1.ps, height=5cm, width=10cm

Kuva 2: Yhtälön $y' = t^2 - y^2$ suuntakenttä

3 Kvalitatiivista teoriaa

Kvalitatiivinen menetelmä tarkoittaa lyhyesti sanottuna pyrkimystä saada tietoa ratkaisujen laadullisesta käyttäytymisestä suoraan yhtälöstä.

3.1 Ensimmäisen kertaluvun yhtälö

Tarkastelemme siis aluksi yhtä ensimmäisen kertaluvun yhtälöä

$$y' = f(t, y).$$

Jos laskemme funktion f arvoja (t, y)-tason pisteissä, saamme kokoelman ratkaisufunktion derivaatan arvoja. Piirtämällä kuhunkin valittuun (t, y)- pisteeseen lyhyen janan, jolla on kulmakertoimena $f(t, y)$, saamme *suuntakentän*, jonka avulla saamme karkean kuvan ratkaisujen käyttäytymisestä tason eri osissa.

Suuntakentän piirtämiseen on `DEtools`- pakkauksessa `DEplot`- komento, jonka avulla voidaan myös piirtää kuvaan numeerisesti laskettuja ratkaisukäyriä haluttuihin pisteisiin.

Tarkastellaan yhtälöä $y' = t^2 - y^2$.

```
> with(DEtools,DEplot): # Lataamme vain funktion DEplot.
> with(plots):
> dyht:=diff(y(t),t)=t^2-y(t)^2:
> DEplot(dyht,y(t),t=-4..4,y=-3..3);
```

Saamme täsmennetyksi kuvaa piirtämällä *isokliinejä*. Nämä ovat käyriä, joilla ratkaisukäyrien derivaatoilla on sama arvo, ts. käyriä, jotka yhdistävät samansuuntaisia viivaelementtejä. Ne ovat siis funktion $f(t, y)$ tasa-arvokäyriä. Sellaisia voidaan piirtää joko `contourplot`- tai `implicitplot`- komennolla. Piirrämme isokliinit, joilla kulmakerroin on $-1, 0$ ja 1 ja yhdistämme samaan kuvaan suuntakentän kanssa.

Aloitamme toistamalla edellä olleen suuntakentän piirtokomennon. Tällä kertaa sijoitamme tulokseksi saatavan grafiikkaesityksen muuttujaan `suuntak`. **Muista** lopettaa tällainen komento **kaksoispisteeseen** eikä puolipisteeseen. Jos et usko, niin kokeile, mutta “omalla vastuulla”.

```
> suuntak:=DEplot(dyht,y(t),t=-4..4,y=-3..3):
> isokl:=implicitplot({t^2-y^2=-1,t^2-y^2=0,t^2-y^2=1},t=-4..4,
                       y=-3..3,grid=[25,25]):
> display({suuntak,isokl});
```

file=isokl1.ps, height=5.5cm, width=10cm

Kuva 3: Suuntakenttä ja isokliinit $-1, 0, 1$

file=dirfield2.ps, height=5.5cm, width=10cm

Kuva 4: Ratkaisukäyrät pisteiden $(-2, 0), (0, 0.5), (0, 1), (1, 0)$ kautta

Komennolla `DEplot` voimme piirtää suuntakentän lisäksi myös numeerisesti laskettuja ratkaisukäyriä haluamiimme pisteisiin. Halutut pisteet ilmaistaan muodossa `[[y(t1)=y1],[y(t2)=y2],...]`. Siis listojen listana, jossa sisälistat ilmaistaan yhtälöilla.

```
> DEplot(dyht,y(t),t=-4..4,y=-3..3,[[y(-2)=0],[y(0)=0.5],
  [y(0)=1],[y(1)=0]]);
```

Komennolla `DEplot` on lukuisa määrä lisävalitsimia, jotka saadaan selville `?DEplot`-avustuskomennolla. Siinä on myös monipuolinen esimerkkivalikoima.

Mainittakoon erityisesti optiot **dirgrid**, **stepsize** ja **method**.

Edellinen annetaan muodossa `dirgrid=[20,20]`. Tämä on oletusarvo ja ilmaisee, että suuntakentän piirtämisessä käytetään kummassakin suunnassa jakoa 20 osaan muodostettaessa laskentahilaa viiva-alkioille. Jälkimmäinen on muotoa `stepsize=0.05` ja ilmaisee ratkaisuapproksimaatiossa käytettävän numeerisen menetelmän askelpituuden. Oletusarvona on annetun piirtovälin pituus jaettuna 20:llä ja oletusmenetelmänä neljännen asteen *Runge–Kutta* tällä kiinteällä askelpituudella. Tarkempia ratkaisukäyriä saadaan valisemalla `method=rkf45` ja/tai pienentämällä askelpituutta.

Koska `DEplot` ei käytä adaptiivista askelpituuden säätöä, eivät ratkaisukäyrät ole välttämättä kovin tarkkoja.

3.2 Autonominen 2×2-systeemi

Yleinen ensimmäisen kertaluvun 2×2 systeemi on muotoa.

$$\begin{cases} x' = f(t, x, y) \\ y' = g(t, x, y) \end{cases}$$

Jos riippumaton muuttuja t ei eksplisiittisesti esiinny yhtälöissä, systeemiä sanotaan *autonomiseksi*. Tällöin se voidaan siis kirjoittaa muotoon

$$\begin{cases} x' = f(x, y) \\ y' = g(x, y) \end{cases}$$

Jos tällainen systeemi ratkaistaan, saadaan ratkaisukäyrät $x(t), y(t)$, jotka voidaan normaaliin tapaan piirtää muuttujan t funktiona, voidaan puhua aikariippuvuuskuvasta.

Toinen, usein havainnollisempi tapa on piirtää käyrä $(x(t), y(t))$ xy- tasoon t:n toimiessa käyräparametrina. Jos ratkaisut olisivat $x(t) = \cos t$ ja $y(t) = \sin t$, niin edellisessä kuvassa olisi kosini- ja sinikäyrä ja jälkimmäisessä yksikköympyrä.

Tätä xy-kuvaa sanotaan *faasitasokuvaksi*. Autonomisuudesta seuraa, että piste (x, y) määrää yksikäsitteisesti faasitasokäyrän. Toisin sanoen alkuehto $x(t_0) = x_0, y(t_0) = y_0$ antaa saman faasitasokäyrän kuin $x(t_1) = x_0, y(t_1) = y_0$, vaikka $t_0 \neq t_1$. Ratkaisukäyrän tangentin suunta on $(x'(t), y'(t))$ ja autonomisuuden takia sen määrää paikka (x, y). Siten suuntakenttä voidaan piirtää muodostamalla xy-tason pisteistö ja laskemalla kussakin (x, y)-pisteessä vektori $(f(x, y), g(x, y))$, joka kertoo pisteen (x, y) kautta kulkevan faasitasokäyrän suunnan. Tämä saadaan aikaan jälleen `DEplot`-komennolla.

Huomaa siis, että autonomisen 2×2 systeemin suuntakenttä liittyy faasitasokuvaan eikä aikariippuvuuskuvaan, kuten edellä käsitellyssä yhden ensimmäisen kertaluvun yhtälön tilanteessa.

Otetaan esimerkiksi vaimentamaton heiluri, jolle voidaan johtaa differentiaaliyhtälö

$$\Theta'' + k \sin \Theta = 0,$$

missä Θ on heilahduskulma ja $k = g/L$ (L on heilurin pituus).

Muunnetaan tämä ensimmäisen kertaluvun systeemiksi merkitsemällä $x = \Theta, \quad y = \Theta'$. Tällöin saadaan systeemi

$$\begin{cases} x' = y \\ y' = -k \sin x \end{cases}$$

Piirrämme `DEplot`-komennolla suuntakentän ja useita ratkaisuapproksimaatioita. Alkupisteet kannattaa kirjoittaa selvyyden vuoksi erilliseen muuttujaan, tällöin on myös helppo hyödyntää työarkin editointiominaisuuksia. Aika-arvo alkupisteessä ja t-väli eivät vaikuta itse käyrään muuten kuin siltä osin, miten pitkä käyrän osa piirretään.

```
> with(DEtools):
> sys:=diff(x(t),t)=y(t),diff(y(t),t)=-k*sin(x(t));
> alkuarvot:=[[x(0)=0,y(0)=1/2],[x(0)=0,y(0)=1],[x(0)=0,y(0)=3/2],
  [x(0)=0,y(0)=2],[x(0)=0,y(0)=5/2],[x(0)=0,y(0)=3],[x(0)=0,y(0)=4],
  [x(0)=0,y(0)=-5/2],[x(0)=0,y(0)=-3],[x(0)=0,y(0)=-4],[x(0)=2*Pi,
  y(0)=1/2],[x(0)=2*Pi,y(0)=1],[x(0)=2*Pi,y(0)=2],[x(0)=2*Pi,y(0)=-2]];
> k:=1:
> DEplot([sys],[x(t),y(t)],t=0..20,x=-Pi..3*Pi,y=-4..4,
  alkuarvot,stepsize=0.05);
```

4 Numeeriset ratkaisut, `dsolve(...,numeric)`

Esittelimme ohjelmointia koskevassa luvussa kohdassa **??** alkaen sivulta **??** joitakin numeerisia menetelmiä ja niiden ohjelmointia -proseduureiksi.

file=heiluri.ps, height=5cm, width=10cm

Kuva 5: Heilurin faasitasokuva

Tässä selvittelemme, miten :n `dsolve`-komennolla muodostetaan ja käsitellään numeerisia ratkaisuja. Siihen liittyy joitakin niksejä, jotka on toivottavasti helppo omaksua esimerkkiemme ja käyttösuositustemme avulla. Komennolla `dsolve(...,type=numeric,...)` on suuri määrä lisävalitsimia niin menetelmään ja virhetoleransseihin kuin ratkaisujen esitystapaan ym. liittyen. Emme esittele niitä mitenkään kattavasti, vaan kiinnitämme huomiota seikkoihin, jotka ensikäyttäjälle saattavat aiheuttaa hämmennystä.

Tarkastellaan alkuarvotehtävää $xx' = t^2, \quad x(1) = 2$. Merkitsemme vaihteeksi ratkaistavaa funktiota ("riippuvaa muuttujaa") x:llä.

Ratkaisemme sekä analyyttisesti että numeerisesti, kun yhtälö sattuu olemaan helposti myös analyyttisesti ratkaistavissa.

```
> yht:=x(t)*diff(x(t),t)=t^2;
```

$$yht := x(t)\frac{d}{dt}x(t) = t^2$$

```
> ratk:=dsolve({yht,x(1)=2},x(t));
```

$$ratk := x(t) = 1/3\sqrt{6\,t^3 + 30}$$

Ratkaistaan nyt numeerisesti.

```
> nratk:=dsolve({yht,x(1)=2},x(t),type=numeric,startinit=true);
                        nratk := proc(rkf45_x)  ...  end
```

Tässä numeerinen ratkaisumenetelmä muodosti uuden proseduurin, jonka avulla ratkaisuapproksimaatioita voidaan laskea. Se on siis toteutettu operaattorityylillä, jota kuvataan kohdassa **??** sivulla **??**. Oletusmenetelmänä on *R*unge–Kutta–Fehlberg 4/5. Suosittelemme option `startinit=true` käyttöä, sen pois jättäminen voi johtaa hämmentäviin tilanteisiin.

Voimme nyt laskea arvoja, taulukoida, piirtää.

```
> nratk(1);
                                  [t = 1, x(t) = 2.]
> array([seq(nratk(xx),xx=[seq(i*0.2,i=0..5)])]);
```

$$\begin{bmatrix} t = 0 & x(t) = 1.825741875912851 \\ t = 0.2 & x(t) = 1.827201878629088 \\ t = 0.4 & x(t) = 1.837389455484131 \\ t = 0.6 & x(t) = 1.864760931485637 \\ t = 0.8 & x(t) = 1.916942008775982 \\ t = 1.0 & x(t) = 2.0 \end{bmatrix}$$

Ratkaisu voidaan määritellä funktioksi:

```
> numx:=u->subs(nratk(u),x(t)):
```

Toinen tapa olisi

```
> nx:=u->rhs(op(2,nratk(u)));
```

Nyt voimme laske tavanomaiseen tapaan vaikkapa

```
> seq(numx(t),t=0..3);
```

$$1.825741875912851, 2., 2.943920291623080, 4.618802159707592$$

Ensimmäinen yllätys tulee vastaan, kun haluamme piirtää funktion kuvaajan.

```
> plot(numx(t),t=0..1);
Error, (in dsolve/numeric_solnall_rkf45) cannot evaluate boolean
```

Tämä johtuu siitä, että `plot`-funktiolle annettava lauseke `numx(t)` yritetään evaluoida symbolisesti, mikä ei numeerisen ratkaisijan kysymyksessä ollen ole mahdollista. Niinpä tämä ennenaikainen evaluaatio on estettävä näin:

```
> plot('numx(t)',t=0..1);
```

Toinen mahdollisuus on käyttää `plot`-komennon funktiosyntaksia, jolloin lausekkeeksi evaluoiminen ei voi `plot`:lle tulla mieleenkään. Vrt. s. **??**. Siis näin:

```
> plot(numx,0..1);
```

Jos halutaan pelkästään kuva, ei tarvita funktiomäärittelyä, vaan voidaan käyttää `plots`-pakkauksen `odeplot`- funktiota.

```
> with(plots):
> odeplot(nratk,[t,x(t)],-1..2);
```

Tässä täytyy noudattaa samaa menettelyä kuin edellä. Siten `odeplot`:lle tulee antaa t-alue muodossa `-1..2`, eikä `t=-1..2`. Käytetään siis samanlaista syntaksia kuin `plot`:n funktiomuodossa.

Itse asiassa `plot`-funktio antaa tarkempia kuvia samalla työmäärällä, koska siinä käytetään adaptiivista arvojen laskentaa toisin kuin `odeplot`:ssa.

Lisää esimerkkejä numeerisista menetelmistä ja erityisesti moninaisista funktion `dsolve(..,numeric)` käyttömahdollisuuksista on komennon `?dsolve,numeric` tulostuksen lisäksi mm. kirjoissa [**?**, ss. 66–81] ja [**?**, ss. 536–548] .

Myös `www`-sivullamme kohdassa *differentiaaliyhtälöt* on alakohta *numeerisia ratkaisuja*, jossa on erilaisia esimerkkejä.