Aalto-yliopisto

Matemaattiset ohjelmistot MS-E1999

5-6. luennon tehtävät

Palautukset 7.4. 23.59 mennessä.

Tämä on viimeinen tehtäväsarja, täydennetty luvatuilla lisätehtävillä.

Arvostelu: Tuplapisteet!

Tehtävät

1. Klassinen saalis-saalistaja-ongelma ("ketut ja jänikset", Volterran-Lotkan malli) on seuraavanlainen: Alueella olevien jänisten ja kettujen lukumääriä merkitään $x(t)$:llä ja $y(t)$:llä vastaavasti.

   Ajatellaan, että jänikset syövät ruohoa ja ketut jäniksiä (ja vain niitä). Kasvua rajoittavia tekijöitä ei ole, muuta kuin jäniksille ketut ja ketuille jänisten puute (eli kilpailevat ketut). Tilannetta mallintakoot yhtälöt

   $$\begin{cases} x' = 200\,x - 4\,x\,y \\ y' = -150\,y + 2\,x\,y. \end{cases}$$

   Ratkaise yhtälösystemi Matlabin `ODE45`:llä.

   Määritä systeemin *kriittiset pisteet* ts. pisteet, joissa molemmat derivaatat= 0 äläkä välitä, vaikka piste $(0, 0)$ on "epäfysikaalinen". Piirrä `ODE45`:llä laskemiasi ratkaisukäyriä aikatasoon ja trajektoreita faasitasoon. (Muistathan `subplot:n`, voi olla siisti.)
   Piirrä myös `pplane8`-ohjelmalla suuntakenttä ja lisää ratkaisukäyriä ja trajektoreita. Huomaa, että voit piirtää samaan pplane-ikkunaan myös laskemiasi ratkaisuja.

   Päättele “olennaisen” kriittisen pisteen luonne (stabiili eli “attraktiivinen” vs. epästabiili eli “repulsiivinen”. “Päättelyyn” riittää klikkailu kriittisen pisteen läheisyydessä.

2. Tarkastellaan vaimennettua heiluria, jonka yhtälö on

   $$\Theta'' + c\,\Theta' + k\,\sin(\Theta) = 0.$$

   Määritä kriittiset pisteet.

Hahmottele tavalla tai toisella `pplane`:llä trajektoreita ja selvitä niitä seuraamalla heilurin käytöstä, vertaa vaimentamattoman käytökseen.

`pplane`-ohje: Valitsemalla *Gallery* ja sieltä *pendulum*, voit asettaa vaimennusta enemmän tai vähemmän. Kokeile pienellä vaimennuksella aluksi ainakin.

Huom: Tehtävän saa haluttaessa tehdä Maplella. Tällöin `dsolve(...,"numeric")` Tarvittaessa lisäohjeita.

3. Maple-tehtävä:
Funktion f kiintopiste on x_0, jolle $f(x_0) = x_0$. Iteroimalla: $x_{n+1} = f(x_n)$ pyritään löytämään jokin kiintopiste Yleisesti: Jos Iteraatio suppenee $\to r$ ja f on jatkuva, niin $f(r) = r$.

Kiintopiste r on *stabiili*, jos x_n pysyy "lähellä" r:ää, kun lähtöpiste x_0 on "riittävän lähellä r :ää. Se on lisäksi "attraktiivinen", jos $x_n \to r$, kun x_0 on 'riittävän lähellä r :ää. Se on lisäksi "attraktiivinen", jos $x_n \to r$,

Päinvastaisessa tapauksessa puhutaan epästabiilista/ *repulsiivisesta* kiintopisteestä. (Määritelmät hiukan epätarkkoja, mutta idea!)

Derivoituvan funktion f kiintopiste r on attraktori, jos $|f'(r)| < 1$ ja repellori, jos $|f'(r)| > 1$. (Jos $= 1$, ei voida päätellä) Iteraatiota voidan havainnollistaa porraskuviolla, joka saadaan Maplessa ihaltavan kätevästi määrittelemällä funktio:

```
porras:=x->plot([[x,x],[x,f(x)],[f(x),f(x)]])
```

Suorita tämä määrittely ja kokeile funktion $f(x) = 3.0x - x^2$ iterointia ja visualisointia:

```
with(plots)
f:=x-> ...
fjaxkuva:=plot([x,f(x)],x=0 .. 3,color=[blue,black]):
display(fjaxkuva, porras(.4))
display(fjaxkuva, porras(.4), porras(f(.4)))
display(fjaxkuva, porras(.4), seq(porras((f@@k)(.4)), k = 1 .. 5))
```

Huomaa: `f@f` on yhdistetty funktio $f \circ f$, ja `(f@@n)` on f :n n-kertainen iteraatio: $f \circ \ldots \circ f$.

Tehtävät:
Iteroidaan funktiota $f(x) = a\,x - x^2$.

a.) Osoita, että $a - 1$ on $f : n$ kiintopiste. Millä a :n arvoilla se on attraktori?

b.) Tutki funktion iterointia, kun $a = 3.1$ Suorita iterointi tällä kertaa for-silmukassa:

```
x[0]:=2.05:
for nn from 0 to 39 do
  x[nn+1]:=f(x[nn]) end do:
 fjaxkuva:=plot([x,f(x)],x=1.5.. 2.5):
 display(fjaxkuva,seq(porras(x[k],k=0..40)));
```

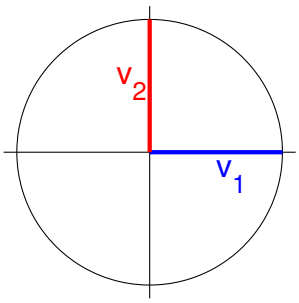
Piirrä myös pisteet $(k, x[k]), k = 0, \ldots, 40$

c.) Tee vastaavat, kun $a = 3.8, x_0 = 1.5$. Mitä havaitset

4. mlG023.tex
Piirrä ympyrä parametrimuodossa, ja tallenna koordinaatit matriisimuodossa `coords = [x;y]` $(x = cos(t); y = sin(t))$. Tämän jälkeen piirrä ympyrä, ja lisää skriptiin seuraavat komennot

```
hold on
% Draw axes
plot([-1.1 1.1],[0 0],'k','linewidth',1)
plot([0 0],[-1.1 1.1],'k','linewidth',1)
% Add flavour: add text and colour.
plot([0,1],[0 0],'b','linewidth',4)
plot([0,0],[0 1],'r','linewidth',4)
p1 = text(.5,-.2,'v_1','fontsize',30);
set(p1,'color',[0 0 1])
p2 = text(-.25,.5,'v_2','fontsize',30);
set(p2,'color',[1 0 0])
% set axis properties
axis([-1.1 1.1 -1.1 1.1])
axis equal
axis off
```

Tuloksen pitäisi olla jotain tällaista:



Jatketaan skriptin editoimista. Määritä matriisi

$$\mathbf{A} = \begin{bmatrix} \frac{7}{6} & -\frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} \end{bmatrix}.$$

a) Laske matriisin $\mathbf{A}$ ominaisarvot ja piirrä ominaisvektorit samaan kuvaan (hold on).

b) Operoi matriisiin `coords` (= ympyrän koordinaattipisteet) matriisilla $\mathbf{A}$ ja piirrä myös transformoidun ympyrän kuva samaan kuvaan.

c) Olkoot ominaisarvot λ_1, λ_2 ja vastaavat ominaisvektorit v_1, v_2. Piirrä vektorit $\lambda_k v_k, k = 1, 2$ ja $-Av_k, \quad k = 1, 2$ samaan kuvaan (Miinus, jotta erottuvat paremmin.)

d) Aja skripti uudelleen niin, että muutat matriisin symmetriseksi, esim näin: `A=rand(2,2); A=A'*A`. Vertaa kuvia, miten symmetrinen eroaa?

5. Oletetaan, että meille on annettu dataa muodossa $(x_k, y_k).k = 1 \dots m$, johon muodustuu kaksi murtopisteen erottamaa lineaarista suuntausta. Esimerkiksi

```
x=-2:0.1:4; y=0.2*sin(3*x);
y(x<1)=y(x<1)+0.5*(x(x<1)-1);
y(x>=1)=y(x>=1)+2*(x(x>=1)-1);
```

muodostaa selvän murtopisteen kohtaan $x = 1$. Intuitiivisesti tuntuu selvältä, että tällaiseen dataan kannattaa sovittaa PNS-suoran sijaan paloittain lineaarinen funktio, ts. "suora murtopisteellä."

Kirjoita ohjelma joka tekee tämän: ohjelman tulee valita murtopiste (s, t) tasosta hiiren klikkauksen perusteella (kts. vihje) ja sovittaa paloittain lineaarisen funktion dataan tätä murtopistettä käyttäen, ts. sovittaa suoran

$$y = k_1 x + b_1, x < s$$

pisteisiin $(x_k, y_k), x_k < s$ ja suoran

$$y = k_2 x + b_2, x > s$$

pisteisiin $(x_k, y_k), x_k > s$.

Vihje: Tehtävän keskeinen osa on murtopisteen valinta ja datapisteiden suodatus.

Murtopisteen valintaan kannattaa käyttää `ginput` funktiota, joka valitsee klikatun pisteen kuvasta tyyliin

```
[x,y,button] = ginput(1);
```

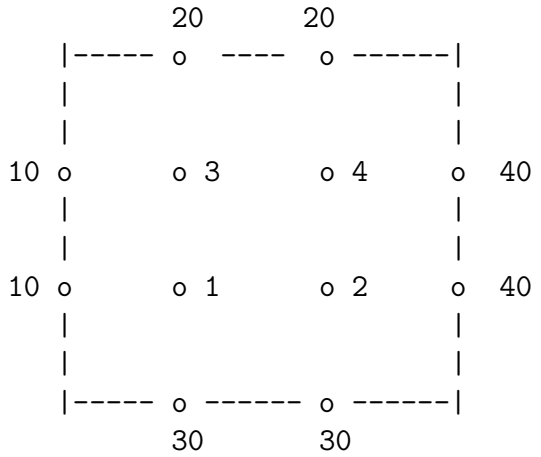
Datan suodatukseen kannattaa käyttää MATLABin loogista indeksointia: esimerkiksi valitaan kaikki vektorin pisteet, jotka ovat pienempiä kuin 5.

```
a = b(b<5);
```

6. **Korjattu versio** (31.3.14)
Tässä tehtävässä tarvitsee vain muodostaa ja ratkaista lineaarinen yhtälöryhmä annettujen ohjeiden mukaan.

Tasapainolämpötilajakauma metallilevyssä.
Kuva esittää metallilevyä, joka on ylä- ja alapinnoiltaan lämpöeristetty ja jonka reunojen lämpötilat on kiinnitetty. (Lämpöä virtaa vain reunojen kautta.) Tasapainolämpötilajakauma saadaan *Laplacen yhtälön* $\Delta u = 0$ ratkaisuna. Numeerinen approksimaatio voidaan laskea ns. differenssimenetelmällä: Jaetaan levy sopivilla hilaviivoilla osiin ja numeroidaan näin muodostuvat solmupisteet. Menetelmä: Kunkin hilasolmun lämpötila on naapurisolmujen lämpötilojen keskiarvo. (Johdetaan kurssin lopulla.)

Muodosta $4 \times 4-$ yhtälösysteemi solmujen $1, 2, 3, 4$ lämpötilojen likiarvoille u_1, u_2, u_3, u_4. Ohje: Aloitetaan solmusta 1: $u_1 = \frac{1}{4}(30 + u_2 + u_3 + 10)$. Vastaavasti muut kolme solmua.



(a) Ratkaise yhtälösysteemi ja sijoita ratkaisulämpötilat ao. hilapisteisiin.

(b) Muodosta $4 \times 4-$ matriisi, jossa on reunalämpötilat ja ratkaisemasi sisäpistelämpötilat sekä nurkissa lähinnä olevien kahden reunasolmun keskiarvot tähän tapaan: `U=[15 20 20 30;10 u3 u4 40; 10 u1 u2 40; 20 30 30 35];` Piirrä ratkaisupinnan approksimaatio: `mesh(U)` tai `surf(U)`.

(b') Ehka hiukan selkeämpää on rakentaa U-matriisi vaiheittain vaikka tähän tapaan:

```
u=u'  % Vaakavektoriksi
U=zeros(4,4)
U(1,:)=[15 20 20 30]
U(2,:)=[10 u(3:4) 40]
...
...
```

7. Tässä tehtävässä tutkitaan kuvien, matriisien ja singulaariarvojen yhteyksiä.

   Matriisin $\mathbf{A} \in \mathbb{R}^{m \times n}$ singulaariarvohajotelma on

   $$\mathbf{A} = \mathbf{USV}^T,$$

   missä matriisi $\mathbf{S}$ on diagonaalimatriisi, ja matriisit $\mathbf{U}$ ja $\mathbf{V}$ ovat ortogonaalisia neliömatriiseja. Matriisin sisältämää informaatiota voidaan tietyssä mielessä kompressoida tiputtamalla osia singulaariarvohajotelmasta pois; on todistettavissa että (MATLABilla ilmaistuna) `U(:,1:k)*S(1:k,1:k)*V(:,1:k)'` on paras mahdollinen rank(k)-approksimaatio matriisille $\mathbf{A}$.

   Kuva voidaan ajatella $m \times n$ matriisina, missä i, j alkio ilmaisee vastaavassa paikassa olevan pikselin väriarvon. Tutkitaan sitten kuinka singulaariarvoja voidaan käyttää hyväksi kuvien pakkaamisessa ja hahmontunnistuksessa.

   Lue haluamasi kuva sisään MATLABin `imread` komennolla. Komento luo (yleensä, mutta hieman kuvasta riippuen), $m \times n \times 3$ matriisin. Tämä vastaa RGB-esitystä: ensimmäisessä kerroksessa on punaisen värin intensiteetit, toisessa vihreän ja kolmannessa sinisen. Muuta tämä matriisi harmaaskaalaan komennoilla `rgb2gray` ja `double`. Tämän jälkeen tee matriisille singulaariarvohajotelma komennolla `[u s v] = svd(P)`, missä `P` on kuvasi matriisiesitys. Tutki sitten millä k:n arvolla komentojono

```
>> M = u(:,1:k)*s(1:k,1:k)*v(:,1:k)';
>> imagesc(M)
```

tuottaa havaittavia tuloksia. Pitäisi myös päteä, että kuvan isommat hahmot alkavat erottua ensin, mikä tekee singulaariarvoista huomattavan tehokkaan työkalun hahmontunnistuksessa.

Vihje: Kuvan ulottuvuuksien ei kannata olla kovin isoja: singulaariarvohajotelma on raskas laskettava. Jos haluat lisähaastetta, erottele kuvan värikerrokset, tee hajotelma niille erikseen, ja kokoa tulokset. Näin saat aikaan värikuvia. MATLABin käyttämä väriskaala saattaa joskus johtaa hilpeisiin kuviin: komennolla `colormap gray` pääset harmaaseen maailmaan.

8. Olkoot c ja z_0 kompleksilukuja. Tällöin iteraation

$$z_n = z_{n-1}^2 + c$$

määräämä dynaaminen systeemi tunnetaan kvadraattisena kuvauksena. Valituille luvuille c ja z_0 ylläoleva rekursio johtaa kompleksiseen lukujonoon $z_1, z_2, z_3 \ldots$. Tätä jonoa kutsutaan z_0:n kiertoradaksi. Riippuen lukujen c ja z_0 valinnasta ratojen muotoja on useita.

Annetulle kiinteälle luvulle c useimmilla z_0 rata lähestyy ääretöntä (eli $|z_n|$ kasvaa rajatta kun $n \to \infty$.) Joillakin c ja z_0 rata kuitenkin suppenee kohti jotain periodista silmukkaa (eli arvot kiertävät z_0 jollain tietyllä etäisyydellä $|z_n|$); joillakin alkuarvoilla rata on kaoottinen. Nämä alkuarvot z_0 ovat kuvauksen Julia-joukko.

Tässä harjoituksessa kirjoitetaan MATLAB-ohjelma, joka laskee ns. täytetyn Julia-joukon, joka koostuu niistä alkioista z_0 joiden radat jollain annetulla arvolla c eivät kasva rajatta – tavallinen Julia-joukko on tämän joukon reuna.

On näytetty, että jos $|z_n|$ kasvaa isommaksi kuin 2 jollain arvolla n, rekursio kasvaa rajatta. Arvoa n jolla tämä tapahtuu, kutsutaan tässä tehtävässä pisteen z_0 "pakonopeudeksi."

Aloita kirjoittamalla funktio `n = escapeVelocity(z0,c,N)`, jossa N on jokin yläraja pakonopeuksille (erityisesti: jos $|z_n| < 2 \; \forall \; n < N$, funktion tulee palauttaa N. Näin vältetään ikuiset silmukat).

Luodaksesi Julia-joukon kirjoita funktio `M=julia(zMax,c,N)`. Argumentti `zMax` määrää kompleksitasosta nelikulmion $|Im(z)| < z_{max}, |Re(z)| < z_{maz}$. c ja N ovat samat argumentit kuin edellä, palautettava matriisi $\mathbf{M}$ koostuu määritetyn hilan pakonopeuksista.

Aloita funktion `julia` kirjoittaminen määrittelemällä 500×500 hila realitasossa, luo sen avulla vastaava hila $\mathbf{Z}$ kompleksitasolle, ja aja funktio `escapeVelocity` jokaiselle matriisin $\mathbf{Z}$ alkiolle.

Vihje: Realiakselin väli $[a, b]$ määritellään MATLABissa komennolla `I = linspace(a,b,n)`, missä n on haluttujen pisteiden määrä, kuten esim. 500. Hila reaalitasolle määritellään komennolla `[x y] = meshgrid(t1,t2)` , missä t1 ja t2 ovat välejä reaaliakselilta. Tästä luodaan kompleksitasoa peittävä hila komennolla `z = x+i*y`.

Kompleksiluvun modulin saa selville itseisarvofunktiolla `abs`.

9. Yksiulotteinen Poisson'n yhtälö on reuna-arvotehtävä:

$$\Delta u(x) = f(x), u(a) = A, u(b) = B,$$

missä Laplace-operaattori on yksinkertaisesti $\Delta = \frac{\partial^2}{\partial x^2}$

Voidaan unohtaa hienot nimittelyt ja tarkastella differentiaaliyhtälön muuntamista diskreetiksi yhtälöryhmäksi jakamalla väli h:n pituisiin osiin ja approksimoimalla 2.derivaattaa jakopisteissä keskidifferenssillä:

$$\Delta_h u(x) = \frac{u(x+h) - 2\,u(x) + u(x-h)}{h^2}.$$

Tehtävänä on ratkaista reuna-arvotehtävä

$$u'' = e^{-x^2}, u(-1) = u(1) = 0.$$

Jos merkitset ratkaisufunktion u approksimaatiota jakopisteessä x_i u_i:llä, $i = 2, \ldots, n+1$ saat yhtälöryhmän, jonka matriisin näet suorittamalla nämä komennot. Tutustu samalla sparse-alkuisiin funktioihin, tässä erityisesti spdiags.

```
n = 5;          % sisasolmujen lkm.
h = 2/(n+1);    % Osavaleja n+1
x = (-1:h:1)'   % Tässä näkyy x-akselin jakopisteet
f = exp(-x.^2); % Yhtälön oikea puoli, "lähdetermi"
e = ones(n,1);
D = spdiags([e -2*e e],[-1 0 1],n,n);  % Kätevä tapa koota diagonaaleittain.
full(D)                          % Katsotaan harvan matriisin esitys täytenä.
A = D/h^2;
```

Reuna-arvotehtävässä tunnettuja arvoja ovat tietysti reunasolmuarvot, ratkaistavana on siis sisäsolmuarvot `sisaind=2:n+1`. Koska reuna-arvot = 0, saat koko u-vektorin alustamalla ensin `u=zeros(n+2,1)` ja "loksauttamalla" sitten sisäsolmuarvot tyyliin `u(sisaind)=A\f(sisaind)` .

Nyt voit piirtää ratkaisufunktion kuvaajan. Suorita tehtävä luokkaa $n = 100$ olevalla arvolla, voit sitten kaksinkertaistaa ja katsoa, miten vaikuttaa.

Kokeile harvalle matriisille myös komentoa `spy`.

Tehtävä on aika valmiiksi neuvottu, mutta aihe voi olla monelle outo ja uusi. Laskennallisesti ainoa merkittävä tapahtuma on lineaarisen yhtälösysteemin ratkaisu.
Tehtävä voi toimia johdatuksena tasossa ja avaruudessa suoritettaviin Laplacen/Poissonin, Lämpö(diffuusio) ja aaltoyhtälöiden ym. lineaaristen osittaisdiffyhtälöiden ratkaisemiseen. Samalla monessa yhteydessä tärkeään harvojen (sparse) matriisien hienosti toteutettuun työkalupakkiin.

10. Ratkaise Maplella reuna-arvotehtävät:

a.)

$$u'' = e^{-x^2}, u(-1) = u(1) = 0.$$

b.)

$$y'' - y^2 + 1 = 0, y(0) = 0, y(1) = 1$$

Edellinen ratkeaa analyyttisesti dsolve:lla. Tarkista ja piirrä.
Jälkimmäinen onnistuu järkevästi vain numeerisesti. Suosittelen dsolven valitsimia:
`numeric,output=listprocedure`

Tällöin ratkaisun jälkikäsittely käy helposti tyyliin:

```
ratk:=dsolve(...,numeric,output=listprocedure);
Y:=eval(y(x),ratk[2]);
Haluttaessa myös:
dY:=eval(diff(y(x),x),ratk[3]));
```

Huomaa, `eval` tekee saman kuin `subs` ja lisäksi evaluoi. Argumenttien järjestys on toinen.

Ohjeita

Diff. yhtälöiden numeriikkaa

Annettu diff. yhtälö(systeemi): $y' = f(t, y), \quad y(t_0) = y_0$

Jos kyseessä on systeemi, niin f ja y ovat vektoriarvoisia.

Eulerin menetelmä $y_{n+1} = y_n + hf(t_n, y_n), \quad y(t_0) = y_0$ GKV $= O(h)$ (GKV=**G**lobaali **k**atkaisu**v**irhe)

Esim:

```
>> F=@(y) [-y(1)+y(2);-y(1)-y(2)]
>> h=0.2;
>> y0=[0;4]
>> y1=y0+h*F(y0)
...
```

Autonomisen DYS:n faasitasokuva

Trajektori faasitasossa tarkoittaa vain sitä, että piirretään $y_1\, y_2-$tasoon pisteitä $(y_1(t_1), y_2(t_1)), (y_1(t_2), y_2(t_2)), \ldots$. Näistä pisteistä $(y_1(t), y_2(t))$, t :n juostessa läpi reaaliarvoja, muodostuu tuo faasitasokäyrä, eli trajektori.
Huomaa, että faasitason suuntakenttä määräytyy autonomiselle systeemille $\mathbf{y}' = f(y_1, y_2)$ pelkästään $(y_1, y_2)-$pisteistä. (Epäautonomiselle $\mathbf{y}' = f(t, y_1, y_2)$ faasitason sijasta pitäisi tarkastella faasiavaruutta, antaapa sen nyt olla.)

Lineaaristen $2 \times 2-$systeemien faasitasoluokitus

Ominaisarvot λ_1, λ_2. Tyyppikuvaus ja stabiilisuus viittaavat kriittiseen pisteeseen, joka lineaarisella homogeenisella on $\vec{0}$. (Jos A on singulaarinen, niin kaikki $N(A)$:n pisteet ovat kriittisiä pisteitä, mutta alla olevassa luokittelussa tämä suljetaan pois (ts. ominaisarvo 0 on suljettu pois).

- Samanmerkkiset $\implies$ *noodi*, jos < 0, niin *nielunoodi*, (vahvasti) stabiili, jos > 0, niin *lähdenoodi*, epästabiili, Trajektorit voivat näyttää potenssifunktiomaisilta erilaisin potenssein (riippuu ominaisarvojen suhteesta), erikoistapauksessa voivat olla jopa sädekimppu. Käytöstä voisi ehkä kutsua yleistetyn "paraabelimaiseksi"

- Erimerkkiset $\implies$ *satula*, epästabiili. Sama potenssikäytös kuin edellä, nyt "hyperbelimäisesti"

- Vain yksi ominaisvektori $\implies$ *degeneroitunut noodi*, jos ominaisarvo $\lambda < 0$, nielu (stabiili) ja jos $\lambda > 0$, niin lähde, epästabiili. Trajektori eivät ulkonäöltään paljon eroa noodityylistä.

- Puhtaasti imaginaariset ominaisarvot ($\pm bi$), *keskus*, heikosti stabiili. Trajektorit ellipsejä.

- Kompleksiset ominaisarvot $\lambda = \alpha \pm i\beta$, *lähdespiraali*, jos $\alpha > 0$, epästabiili, ja *nieluspiraali*, jos $\alpha < 0$, stabiili. Trajektorit elliptisiä spiraaleja.

pplane

Avaa yllä oleva Rice-universityn pplane-viite ja paina `pplane` painiketta. `PPLANE Equation window`:ssa valitse `Gallery` ja sieltä `linear`. Nyt pääset kätevästi määrittelemään 2×2-matriisin alkiot. Paina Graph Phase Plane. Voit valita hiirellä alkuarvopisteen, ja ohjelma piirtää sen kautta kulkevan trajektorin. Kokeile (aluksi vaikka valmiilla oletusmatriisilla) Usein tulet piirtäneeksi tarpeettoman monta trajektoria. Painamalla uudestaan Graph Phase Plane-painiketta, saat taas puhtaan suuntakentän. Piirtoikkunaa on syytä muuttaa tehtävän mukaan.