Matematiikan laitos
Helsingin yliopisto

MATEMATIIKAN MENETELMÄKURSSI

Matti Vuorinen
vuorinen 'at' utu 'dot' fi

Sisältö

Kurssin kotisivu:
https://wiki.helsinki.fi/pages/viewpage.action?pageId=70225790

Johdanto

Matematiikan menetelmäkurssilla esitellään erityisesti luonnontieteiden tarvitsemia matematiikan menetelmiä opetusta tietokoneella havainnollistaen. Kurssin laskuharjoituksetkin vaativat tietokoneiden käyttöä. Esitiedoiksi edellytetään matematiikan approbatur tai vastaavat tiedot (esim. teoksen [LP] mukaisesti) sekä jonkin ohjelmointikielen (esim. C-kieli) käyttötaitoa.

Matematiikan kulmakivi, todistukset, on tällä kurssilla korvattu pienillä ohjelmanpätkillä, jotka generoivat teoriaa valaisevia kuvia tai tulosteita. Näin muodostuu eräänlainen kokeellisen matematiikan kurssi, jossa ilmiöitä tutkitaan esim. graafisesti.

1990-luvun alussa voimakkaasti yleistyneet tieteellisen laskennan kielet kuten Maple, `Mathematica` ja MATLAB lienevät nykyään varteenotettavimpia vaihtoehtoja tällaisen kurssin tarpeisiin. Näille kaikille on ominaista grafiikan helppo käyttö ja monien numeeristen perustoimintojen automatisointi. Maple ja `Mathematica` pystyvät myös ns. symboliseen laskentaan.

Kurssin pääkieleksi valittu MATLAB pohjautuu tunnettuun LINPACK-ohjelmakirjastoon ja on osoittautunut erittäin soveliaaksi tarkoitukseen. MATLAB on vakiinnuttanut asemansa myös numeeriikan ammattilaisten työkaluna, jonka suosio on edelleen kasvussa. Symbolista lähestymistapaa puolestaan kurssilla edustaa Okko Kanervan laatima jakso Maple-kielestä. (Hän on myös muokannut kurssimonisteen tekstiä yms. kauttaaltaan ja korjannut samalla lukuisia virheitä.) Kurssiin sisältyy useita kymmeniä pieniä MATLAB-ohjelmia, joista tärkeimpien Maple-versiot on esitetty lopussa olevassa liitteessä.

Mahdollinen palaute pyritään ottamaan huomioon monisteen tulevissa versioissa.

MATLAB on esimerkki kaupallisesta skripti-kielestä. Viime vuosina on tullut suosituksi ilmainen Python-kieli, jonka laajennukset Numerical Python ja Scientific Python tarjoavat numeerisilta ominaisuuksiltaan lähes MATLABin veroisen ympäristön. Kurssimme harjoitustehtäviin tarjotaan malliratkaisut myös Python-kielellä.

Matti Vuorinen

`vuorinen 'at' users 'dot' csc 'dot' fi`

1 Numeeristen ongelmien luonteesta

Tämän luvun tarkoituksena on luoda yleiskatsaus muutamiin matemaattisiin ongelmiin ja esittää, miten ne voidaan ratkaista MATLAB-ohjelmointikielellä. Näiden esimerkkien yhteydessä ei vielä perustella sitä, miksi ratkaisu tehdään esitetyllä tavalla — se selviää myöhemmissä luvuissa. MATLABin systemaattisen opiskelun aloitamme luvussa 2.

Ennen em. esimerkkejä pari alkuhuomautusta kurssia tukevasta kirjallisuudesta ja ohjelmistoista, joista on luettelo monisteen lopussa. Mikään teos ei sellaisenaan kata koko aihepiiriä, mutta hyviä lähteitä ovat kirjallisuusluettelossa mainitut, [M], [NR], [S1] sekä käsikirja [AS].

Työvälineinämme toimivat ohjelmat MATLAB ja Maple ovat molemmat tulkkeja ja soveltuvat hyvin interaktiiviseen työskentelyyn. Niitä on esitelty esim. teoksissa [MATu] ja [Mpl/L].

Siirrymme nyt tarkastelemaan esimerkkejä matemaattisista ongelmista, joiden ratkaisu voidaan tehokkaasti tehdä MATLAB-kielellä. Ratkaisua edeltävä vaihe on MATLAB-ohjelman käynnistys. Kukin ratkaisu edellyttää yleensä parin tekstitiedoston tekemistä editorilla, joka MATLABin komentotilassa käynnistyy komennolla `!emacs`. Tässä yhteydessä huutomerkki ilmaisee, että kyseessä on käyttöjärjestelmän tuntema komento "`emacs`". Emacsin asemesta voidaan käyttää mitä tahansa muutakin editoria.

1.1. Juuren haku. Etsi yhtälön $e^{-3x} - x = 0$ juuri.
MATLAB-ratkaisu: Tehdään tiedosto f101.m:

```
function y=f101(x)
y=exp(-3*x)-x;
```

Annetaan MATLAB-komento: `x=fzero('f101',0.5)`
Saadaan vastaus: `x=0.3500`.
Tarkistus tehdään kirjoittamalla `check=f101(x)` .
Saadaan vastaus: `check=0`, joten `x` on todella etsitty juuri.

1.2. Lineaarisen yhtälöryhmän $Ax = b$ ratkaisu.
Yli puolet numeerisen matematiikan ongelmista käyttää osanaan lineaarisen yhtälöryhmän ratkaisua. $n \times n$ -matriisin A kääntämiseen tarvitaan luokkaa n^3 oleva määrä peruslaskutoimituksia, joten esim. 400×400 -matriisin kääntämiseen tarvitaan n. $64 \cdot 10^6$ laskutoimitusta. (Vertailun vuoksi todetaan, että yhdessä vuodessa on n. $32 \cdot 10^6$ sekuntia.)

Tehtävä: Muodosta 50×50 -satunnaismatriisi a ja 50×1 -matriisi b. Ratkaise $ax = b$.

MATLAB-ratkaisu: Ratkaisu saadaan antamalla seuraavat syötteet:

```
a=rand(50,50); b=rand(50,1);
x=a\b; check=norm(a*x-b)
```

Muuttujan `check` lukuarvoa voidaan käyttää arviona löydetyn numeerisen ratkaisun virheelle. Toinen, pidemmissä tehtävissä ainoa järkevä menettely, on luoda ratkaisun sisältävä m-tiedosto, esim. e102.m, joka suoritetaan komennolla "`e102`".

```
% FILE  e102.m  begins.
a= rand(50,50);  b = rand(50,1);
x= a\b;
check = norm(a*x-b)
% FILE  e102.m  ends.
```

1.3. Minimin haku. Monet käytännön ongelmat johtavat minimointitehtäviin. Esimerkkinä kustannusten minimointi tuotteen valmistuksessa.

Tehtävä: Minimoi funktio $f_{103}(x) = \sin(\tan(x)) + 1.2\cos(x)$ välillä $(-1.6, 1.6)$.

MATLAB-ratkaisu: Tehdään tiedosto f103.m:

```
function y=f103(x)
y=sin(tan(x))+1.2*cos(x);
```

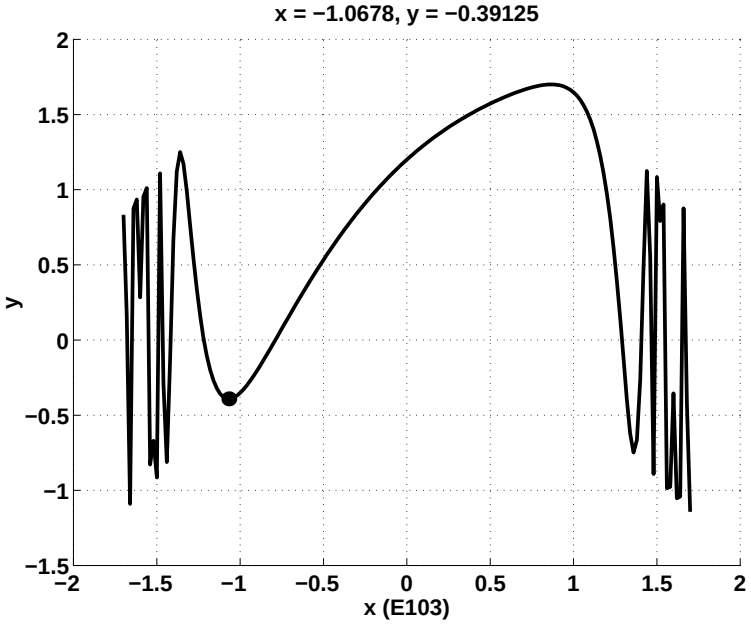
Annetaan komento: `a=fmin('f103',-1.6,1.6)`

Saadaan vastaus: `a=-1.0678`.

Löydetty minimi on lokaali, ei globaali.

Ratkaisu m-tiedoston avulla:

```
% FILE  e103.m  begins.
clf;
x = -1.7: 0.02: 1.7; y = f103(x);
axes('FontSize',[15],'FontWeight','bold');  hold on;
plot(x,y); title('Funktio esim. 1.3'),pause;
a = fmin('f103',-1.6,1.6)
disp(['Funktion arvo = ' num2str(f103(a)) ]);
txt =['x = ' num2str(a) ', y = ' num2str(f103(a)) ]  ;
grid;
plt1=plot(x,y,a,f103(a),'k.','MarkerSize',30);
title(txt,'FontSize',[15]);
xlabel('x (E103)','FontSize',[15] ); ylabel('y','FontSize',[15] );
set(plt1,'LineWidth', 2.5);       % Line width is changed
hold off;
% FILE  e103.m  ends.
```



1.4. Data-analyysi. Pienimmän neliösumman (pns) polynomin sovitus pisteistöön.

Tehtävä: Muodosta vektorit x, y, joille $x(j) = 0.3 * (j-1)$, $y(j) = x(j)^2 + \pi x(j) + 0.5 * \sin(10 * x(j))$, kun $j = 1, \ldots, 17$, ja etsi tähän dataan sovitetun toisen asteen polynomin $c_1 x^2 + c_2 x + c_3$ kertoimet $c = (c_1, c_2, c_3)$. Lähtökohtanamme on, että näin muodostettu data on "synteettistä" dataa, jossa sin-termi kuvaa "mittausvirhettä". Synteettinen data soveltuu hyvin tarkoitukseen, sillä odotamme, että "oikeassa vastauksessa" (c_1, c_2) on lähellä lukuparia $(1, \pi)$.

MATLAB-ratkaisu:

```
x=0:0.3:5;
y=x.^2+pi*x+0.5*sin(10*x);
coef=polyfit(x,y,2)
```

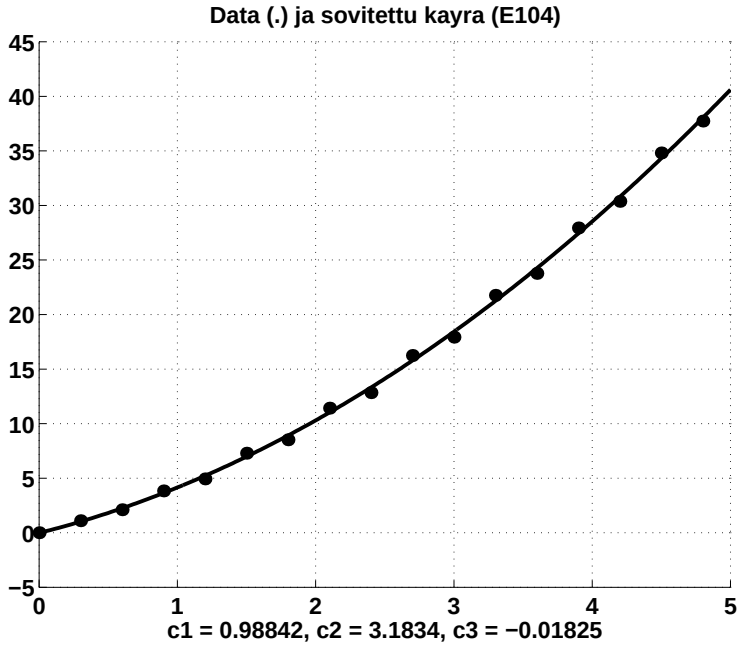
Saadaan ratkaisu

```
coef=0.9884  3.1834  -0.0183
```

Piirretään vielä kuva datasta ja sovitetusta polynomista, ja esitetään toinen ratkaisu m-tiedostona.

```
% FILE  e104.m  begins.
clf;
x= 0: 0.3: 5;
y = x.^2 + pi*x + 0.5*sin(10*x);
```

```
coef = polyfit(x,y,2)
c = coef; txt = ['c1 = ' num2str(c(1)) ', c2 = ' num2str(c(2)) ];
txt = [txt ', c3 = ' num2str(c(3)) ];
axes('FontSize',[15],'FontWeight','bold');  hold on;

xi = 0: 0.02:5;  yi = polyval(coef,xi);
pic1=plot(xi,yi,x,y,'k.','MarkerSize',25);
 title('Data (.) ja sovitettu kayra (E104)','FontSize',[15] );
xlabel(txt,'FontSize',[15]); pause;
grid;
set(pic1,'LineWidth', 2.5);       % Line width is changed
print -deps fig104.eps
% FILE  e104.m  ends.
```



1.5. Tavallisen differentiaaliyhtälön alkuarvotehtävän ratkaisu. Monissa sovelluksissa (esim. mekaniikassa) esiintyy toisen kertaluvun vakiokertoimisen differentiaaliyhtälön alkuarvotehtävä.

Tehtävä: Ratkaise DY: $y'' + ay' + by = 0$ alkuarvoilla $y(0) = 1$, $y'(0) = 0$, kun $a = -0.05$ ja $b = 0.15$. Etsi numeerinen approksimaatio ratkaisulle pisteessä $x = 1.0$.

MATLAB-ratkaisu: Palautetaan 2. kertaluvun DY 1. kertaluvun DY-ryhmäksi seuraavasti. Kirjoitetaan aluksi DY muotoon $y'' = -ay' - by$. Merkitään $y_1 = y$, $y_2 = y'$, jolloin DY $\Leftrightarrow$

$$\begin{cases} y_1' = y_2 \\ y_2' = -ay_2 - by_1 \end{cases}.$$

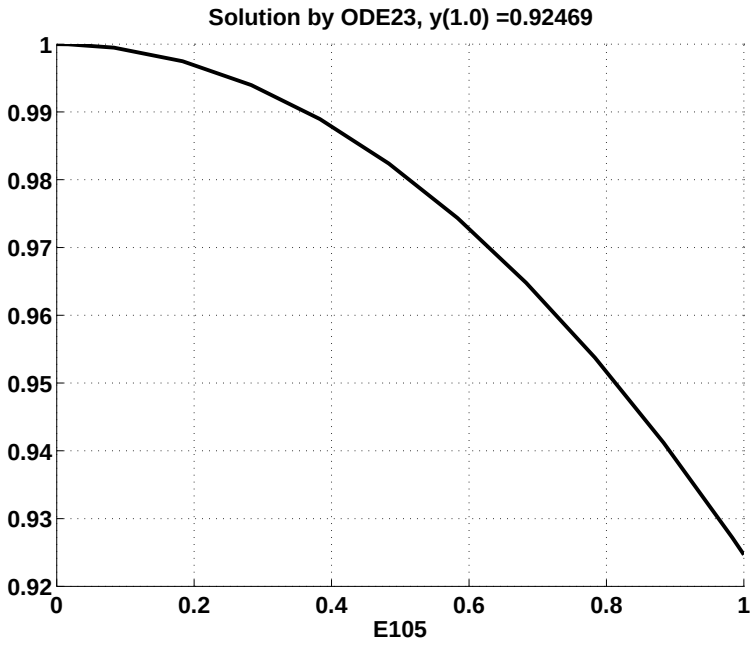
Luodaan seuraavat tiedostot deq105.m ja e105.m, jolloin kirjoittamalla `e105` saadaan oheinen kuva.

```
% FILE  e105.m  begins.  31 Aug. 1999
% USES ode23.m, deq105.m
% Solves y" + a*y' + b*y = 0; y(0) = 1; y'(0) = 0.0
% Want  y(1.0) = ?
% Write as: y" = -a*y' - b*y, Put y2 = y',  y1 = y and get
%           y1' = y2
%           y2' = -a*y2 - b*y1

clear;
global a;
global b;
a =- 0.05; b= 0.15;
fval = 1.0;
dfval = 0.0;
[t,y] = ode23('deq105', [0.0 1.0], [fval dfval]);
clf;
axes('FontSize',[15],'FontWeight','bold');  hold on;
plt1=plot(t,y(:,1));
txt=['Solution by ODE23, y(1.0) ='num2str(y(size(y,1),1)) ];
title(txt,'FontSize',[15]);
xlabel('E105','FontSize',[15]);
set(plt1,'LineWidth', 2.5);       % Line width is changed
grid;
hold off;
pause;
A=[t,y(:,1),y(:,2)]
```

```
% FILE  e105.m  ends.
```

```
         0    1.0000         0
    0.0005    1.0000   -0.0001
    0.0032    1.0000   -0.0005
    0.0165    1.0000   -0.0025
    0.0832    0.9995   -0.0125
       .         .         .
       .         .         .
       .         .         .
    0.8832    0.9412   -0.1328
    0.9832    0.9272   -0.1475
    1.0000    0.9247   -0.1500
```

```
function ydot=deq105(t,y)
global a;
global b;
ydot=[y(2);-a*y(2)-b*y(1)];
```



1.6. Integraalien laskeminen. Kaarenpituuden ja pinta-alojen määritysongelmat johtavat usein "mahdottomiin" integrointeihin.

Tehtävä. Laske integraali

$$\int_0^{\pi} (\sin x)^2 \ \exp(\sqrt{x})\, dx\,.$$

MATLAB-ratkaisu: Luodaan tiedosto fquad.m:

```
function y=fquad(x)
y=(sin(x).^2).*exp(sqrt(x));
```

Kirjoittamalla

```
a=quad('fquad',0,pi)
```

saadaan vastaus `a=5.5256`. Vaihtoehtoinen tapa on käyttää tiedostoa kuten alla.

```
% FILE  e106.m  begins.
% 31. Aug. 1993
x = 0: 0.01: pi; y = fquad(x);
plot(x,y), title('Integrand, e106'),pause;
a= quad('fquad',0, pi)
% FILE  e106.m  ends.
```

1.7. Interpolointiongelma. Kolmannen asteen polynomin arvot pisteissä $-1, 0, 1, 2$ ovat $0, 1, 0.5, 2$. Etsi polynomin arvo pisteessä $x = 1.5$.

MATLAB:

```
x=[-1 0 1 2]; y=[0 1 0.5 2];
n=length(x)-1;
coef=polyfit(x,y,n);
y0=polyval(coef,1.5);
disp(['y0 = ' num2str(y0)]);
```

MATLAB kirjoittaa kuvaruudulle vastauksen:

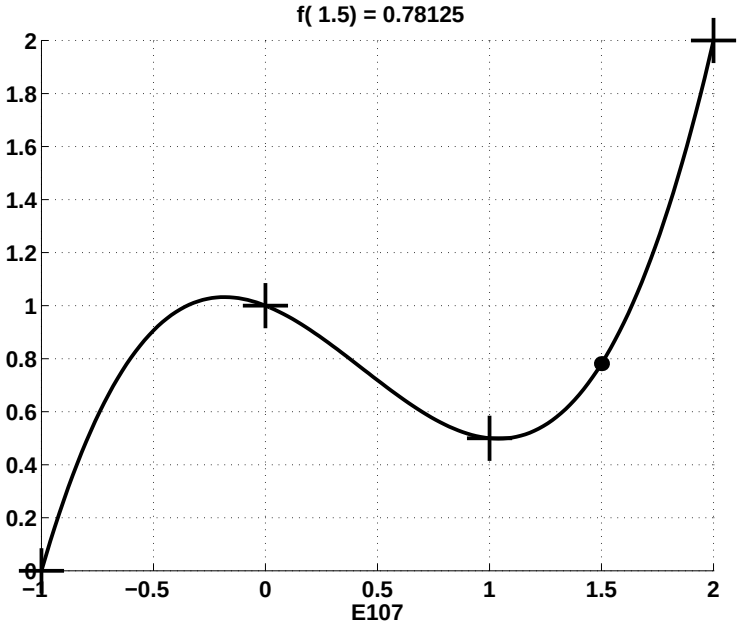
```
y0 = 0.78125
```

Tarkistus:

```
xi=-1:0.02:2; yi=polyval(coef,xi);
plot(xi,yi,1.5,y0,'ko');
```

Esitetään vielä ratkaisu m-tiedostona, joka suoritetaan tavalliseen tapaan komennolla “e107”.

```
% FILE  e107.m  begins.  31 Aug. 1999
clf;
x = [-1 0 1 2]; y = [0 1 0.5 2];
n =length(x)-1;
coef = polyfit(x,y,n);
x0 = 1.5; y0 = polyval(coef,x0)
xi = -1: 0.02: 2; yi = polyval(coef,xi);
txt = [' f( ' num2str(x0) ') = ' num2str(y0)];
axes('FontSize',[15],'FontWeight','bold');  hold on;
plt1=plot(xi,yi,x0,y0,'k.',x,y,'k+','MarkerSize',30);
title(txt,'FontSize',[15]);
set(plt1,'LineWidth', 2.5);       % Line width is changed
hold off;
xlabel('E107','FontSize',[15] );
grid;
pause;
% FILE  e107.m  ends.
```



1.8. Dirichlet'n ongelma. Olkoon $D = \{(x, y) : 0 \leq x \leq 4, 0 \leq y \leq 6\}$ ja $u\colon D \to \mathbb{R}$ funktio s.e.

$$\left.\begin{array}{l} u_{xx} + u_{yy} = 0 \\ u(x,0) = f_1(x) = 0 \\ u(x,6) = f_2(x) = 180 \end{array}\right\} \; 0 \leq x \leq 4$$

$$\left.\begin{array}{l} u(0,y) = f_3(y) = 80 \\ u(4,y) = f_4(y) = 0 \end{array}\right\} \; 0 \leq y \leq 6$$

Tehtävä voidaan ratkaista numeerisin menetelmin.
Ratkaisu MATLABilla esitetään kurssin loppupuolella.

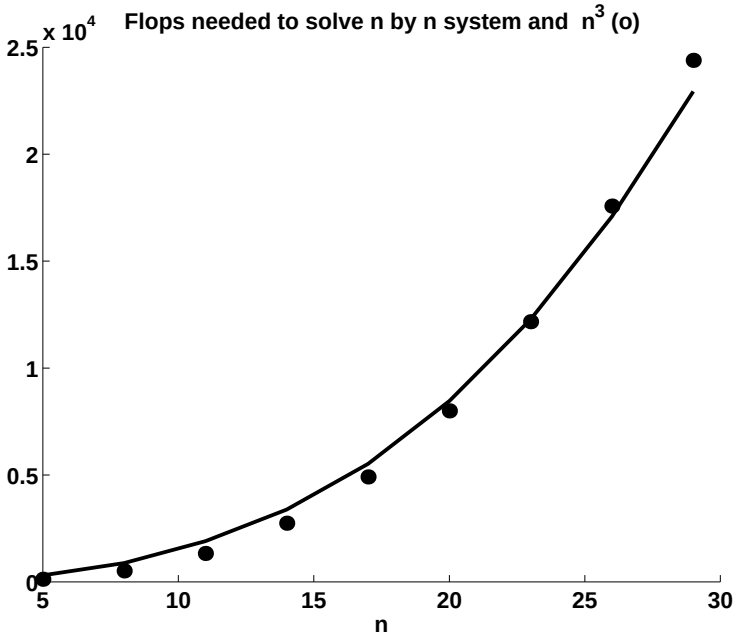
1.9. Matemaattinen kokeilutoiminta. **Tehtävä:** Selvitä kokeilemalla montako flopsia (floating point operations) ts. laskutoimitusta MATLAB käyttää $n \times n$ -yhtälöryhmän ratkaisuun, kun `n=5:3:30`.
MATLAB:

```
size2=[]; ops=[];
for n=5:3:30
   a=rand(n,n); b=rand(n,1);
   f1=flops; x=a\b; f2=flops-f1;
   size2=[size2 n]; ops=[ops f2];
end;
```

Piirretään vielä samaan koordinaatistoon kuva flopsien määrästä n:n funktiona ja vertailufunktiosta n^3.

```
% FILE  e109.m  begins.  31 Aug. 1999
clf;
size2 = []; ops = [];
for n =5:3:30
    a = rand(n,n); b = rand(n,1);
    f1= flops; x= a\b; f2 = flops-f1;
    size2 = [size2 n]; ops = [ops f2];
end;
axes('FontSize',[15],'FontWeight','bold');  hold on;

plt1=plot(size2, ops, size2, size2.^3, 'k.','MarkerSize',30);

xlabel('n','FontSize',[15] );
title(' Flops needed to solve n by n system and  n^3 (o) ', ...
```

```
      'FontSize',[15]);
set(plt1,'LineWidth', 2.5);       % Line width is changed
hold off;
% FILE  e109.m  ends.
```



Kuten yllä on esimerkeissä 1.1 – 1.9 nähty, voidaan MATLABin avulla vaivattomasti ratkaista numeerisia tehtäviä, joiden ratkaisu ilman apuvälineitä olisi työlästä tai mahdotonta. Syy MATLABin käytön vaivattomuuteen on sen sisäänrakennetuissa perusfunktioissa (mm. `fzero`, \, `fmin`, `polyfit`, `ode23`, `quad`, `polyval`), jotka automatisoivat pitkälle tavallisimmat numeeriset tehtävät, ja lisäpiirteenä on mainittava vaivaton tulosten graafinen esittäminen. Huomion arvoinen seikka on sekin, että MATLABin toiminta PC- ja Unix-koneissa on samanlaista.

Vaivattomuutensa lisäksi MATLAB on erittäin tehokas numeerisissa tehtävissä. Tähän on kaksi syytä. Ensinnäkin pohjana olevat algoritmit perustuvat laajalti testattuun ja optimoituun LINPACK-kirjastoon. Toiseksi MATLAB on nopea. Nopeus perustuu siihen, että MATLAB on matriisikieli, jossa peruslaskutoimitukset (`+`, `-`, `^`, `/`) on suoraan määritelty matriiseille (eikä pelkästään kahden reaaliluvun välille). Peukalosäännöksi voidaan sanoa, että numeerisissa tehtävissä MATLAB voi olla jopa sata kertaa nopeampi kuin `Mathematica`.

MATLAB-ohjelmointia opetellessa on syytä kiinnittää huomiota juuri matriisien käsittelyyn; esim. jos `a` on matriisi, on sille aina määritelty alkioittainen

potenssiin korotus `a.^3`, mutta `a^2` on määritelty vain, jos `a` on neliömatriisi. Perustietotyyppi on kompleksinen matriisi.

1.10. Ohjelmointivinkkejä. Kannattaa pyrkiä luomaan systemaattiset ohjelmointitottumukset. Koska tyypillinen istunto käsittää uusien m-tiedostojen luomista, olisi erityisen tärkeää löytää näppärä nimeämiskäytäntö luotaville tiedostoille. Suositeltava on esimerkiksi seuraava:

`h03t02.m`: harj. 3 tehtävän 2 ratkaisu;
`h03t02.dat`: harj. 3 tehtävä 2, tuotettu data;
`f0302.m`: harj. 3 tehtävän 2 tarvitsema funktio.

Olemme edellä jo useasti nähneet, että MATLAB-ohjelmat ovat tavallisia MATLABin syntaksin mukaan tehtyjä tekstitiedostoja ja koostuvat siis ASCII-merkeistä. Yksinkertaisuuden vuoksi tulemme jatkossa tarkastelemaan pelkästään tekstitiedostoja, vaikka MATLAB voi lukea ja kirjoittaa myös binääritiedostoja. Koska katsomme tiedostojen sisältöä tavallisesti editorilla (emacs), on kunkin tekstitiedoston alkuun ja loppuun syytä laittaa tiedoston nimi, jolloin tiedostojen järjestyksen ylläpito on helpompaa. Edellä on tätä ideaa toteutettu m-tiedostojen kirjoittamisessa. Tulemme seuraamaan tätä käytäntöä jatkossakin. Tiedosto näyttää siis seuraavalta:

```
% FILE  h0302.m  begins.
.....................
. tiedoston sisalto .
.....................
% FILE  h0302.m  ends.
```

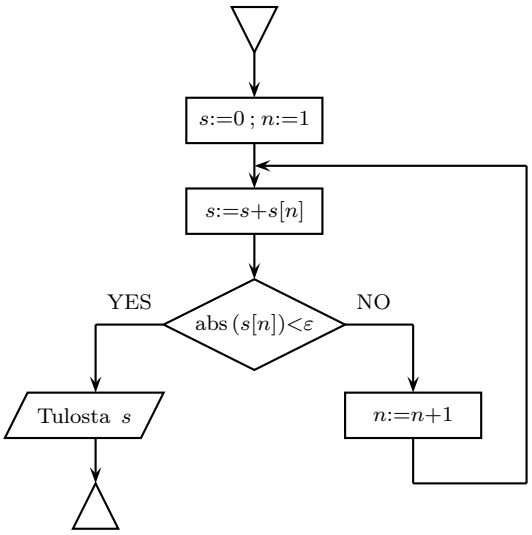
Tiedostojen nimien tulisi MATLABissa olla pidempiä kuin muuttujien nimien. (Voi tulla käsittämättömiä virhetilanteita, jos on tiedosto `x.m` ja jos toisaalta `x`:ää käytetään muuttujana toisessa merkityksessä). Lisää ohjelmointiesimerkkejä on luvussa 2.

1.11. Algoritmin käsite. Numeeriset menetelmät kuvataan tavallisesti *algoritmin* muodossa. Algoritmi on (äärellinen) joukko sääntöjä, joka kullekin (kelvolliselle) syötteelle tuottaa ainakin yhden tulosteen. Lisäksi vaaditaan, että algoritmin toiminta kullekin syötteelle päättyy äärellisen askelmäärän jälkeen.

1.12. Algoritmin esitystapoja. (a) Sanallinen, (b) kaavio (esim. lohkokaavio), (c) ohjelmointikieli.

(a) Lasketaan sarjan $\sum_{n=0}^{\infty} s_n$ termejä yhteen, kunnes tavataan termi s_n s.e. $|s_n| < \varepsilon$

(b)



(c) (Pascal:)

```
ss:=0; n:=1;
repeat
begin ss := ss + s[n]; n := n + 1; end;
until ( abs(s[n])<eps )
writeln(ss);
```

(C-kieli:)

```
S = 0.0; n = 1;
do {S = S + s[n]; n = n + 1;}
while ( fabs(s[n])<eps);
printf("%12.5 E", S);
```

(MATLAB:)

```
S = 0; n = 1;
while not (abs(s(n)) < eps)
   S = S + s(n);
   n = n + 1;
end
S
```

1.13. Esimerkki. Merkitään $(a,n) = a(a+1)\cdots(a+n-1)$ ja $(a,0) = 1$, jolloin $(1,n) = n!$. Määritellään *Gaussin hypergeometrinen funktio* kaavalla

$$ {}_2F_1(a,b;c;x) = F(a,b;c;x) = \sum_{n=0}^{\infty} \frac{(a,n)(b,n)}{(c,n)} \frac{x^n}{(1,n)} \,. $$

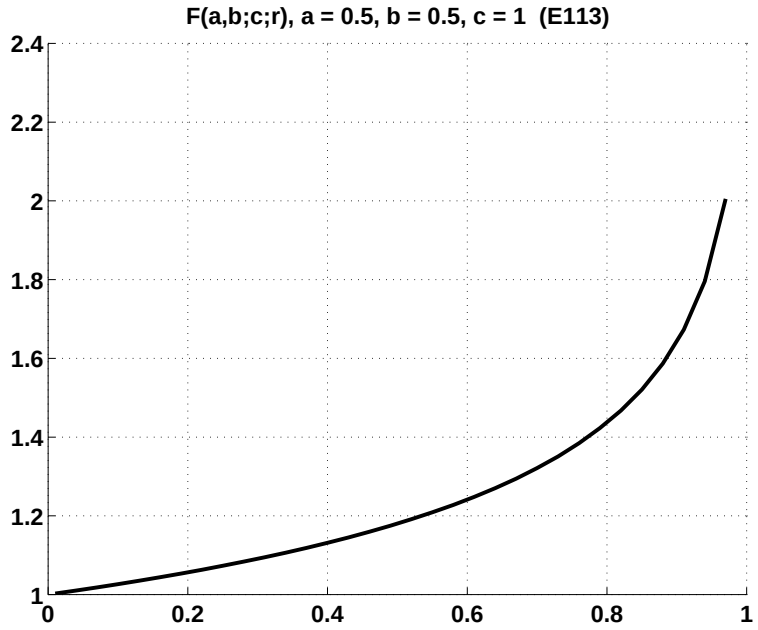
Esitämme nyt MATLAB-algoritmin ${}_2F_1$:n laskemiseksi osasumman $\sum_{p=0}^{200}$ avulla tapauksessa, missä $a,b,c \in \mathbb{R}$ ja x voi olla $m \times n$ -matriisi. Vaihtoehtoisia algoritmeja ${}_2F_1$:lle on esitetty teoksessa [NR] s. 271 (ja myös teoksessa [Mos]).

```
function y = fabc(a,b,c,r)
% Compute 2F1(a,b;c;r)
% The Gauss hypergeometric function F(a,b;c;r)
if ( (size(a) ~= 1) | (size(b) ~= 1) | (size(c) ~= 1) )
   disp('Parameter error in FABC.m');
   end;
[m,n] =size(r);
one =ones(m,n);
s=one;
g= 1.0;

for j=1:100,
        g=g*(a+j-1)*(b+j-1)/(j*(c+j-1));
        s = s+ g*(r.^j);
end;
y = s;
return;
% FILE  fabc.m  ends.
```

Huomaa erityisesti, kuinka muodostetaan $m \times n$ -matriisi `one`, jonka kaikki alkiot ovat 1.

```
% FILE  e113.m  begins.  31 Aug. 1999
clf;
r = 0.01: 0.03: 0.99;  a = 0.5; b = 0.5; c = 1.0;
y = fabc(0.5,0.5,1.0,r);
axes('FontSize',[15],'FontWeight','bold');  hold on;
txt = ['a = ' num2str(a) ', b = ' num2str(b) ', c = ' num2str(c)];
plt1=plot(r,y);
title(['F(a,b;c;r), ' txt  '  (E113)'],'FontSize',[15]);
grid;
```

```
set(plt1,'LineWidth', 2.5);       % Line width is changed
hold off;
% FILE  e113.m  ends.
```



1.14. Algoritmin kompleksisuus. Numeerisessa laskennassa eniten käytetyt algoritmit ovat tavallisesti laskennan kriittinen pullonkaula. Näitä algoritmeja ovat mm. suurten lineaaristen yhtälöryhmien ratkaisu, differentiaaliyhtälöiden ratkaisu sekä erilaiset optimointialgoritmit. Tällaisten algoritmien kompleksisuus on muodostunut tärkeäksi algoritmin hyvyyskriteeriksi. Saman tehtävän ratkaisuun voi olla suoritusajaltaan hyvin erilaisia algoritmeja. Algoritmin tehokkuutta tai kompleksisuutta mitataan tavallisesti polynomi- tai eksponenttifunktion avulla syötteen koosta. Esimerkiksi $n \times n$ -matriisin kääntö on noin kompleksisuutta n^3 oleva operaatio. Funktion arvojen lasku n-avaruuden kuution $[0,1]^n$ kärkipisteissä on kompleksisuutta 2^n oleva toimitus (sillä kärkipisteitä on 2^n kpl).

Algoritmin kompleksisuudella on suuri periaatteellinen merkitys ja käytännöllistä merkitystä, jos laskenta-ajat muodostuvat pitkiksi. Jo kohtuullisilla n:n arvoilla eksponentiaaliset algoritmit ovat liian hitaita (esim. $n = 100$).

Eräät tärkeimmät läpimurrot numeriikan alalla, FFT (Fast Fourier Transformation) ja Hachiyanin algoritmi ns. kauppamatkustajan ongelmaan, liittyvät aikaisempien vastaavien algoritmien kompleksisuuden parantamiseen suurilla n:n arvoilla.

Tämän kurssin puitteissa algoritmien kompleksisuus on esillä tavallisesti vain sivuteemana. (Ks. kohta 1.9 yllä.)

1.15. Liukuluvut. Reaalilukujen esitys tietokoneen muistissa tapahtuu koneesta ja ohjelmasta riippuvalla tarkkuudella. Yleensä reaalilukujen esitykseen tietokoneissa käytetään muuta kuin 10-järjestelmää, esim. binäärilukuja. Eri kantajärjestelmien välinen konversio on periaatteessa suoraviivainen, ja tarkemmin se on selitetty teoksessa [AS], s. 1012. Reaalilukujen esitystä tietokoneessa kutsutaan *liukulukuesitykseksi* (floating point number). Karkeasti ottaen liukulukuesitys on samanlainen kuin ns. *tieteellinen merkintätapa* jossa esim. lukuja 297.2 ja -0.00029 merkitään $2.972 \cdot 10^2$ ja $-2.9 \cdot 10^{-4}$ ja $637000 = 6.370 \cdot 10^5$ (jos 4 merkitsevää numeroa). Tieteellinen merkintätapa koostuu siis seuraavista osista: a) etumerkki, b) mantissa (välillä $[1, 10)$), c) eksponenttiosa, ja tässä mantissan pituus määräytyy merkitsevien numeroiden mukaan. MATLABin funktiot `base2dec` ja `dec2base` tekevät konversion kymmenkantaisten ja binäärilukujen välillä.

Liukulukuesityksen tarkkuus määräytyy mantissan pituudesta, joten siinä on yleensä aina virhettä. Liukulukuaritmetiikassa (ts. yhteen-, vähennys-, jako-, kertolasku) pätevät samankaltaiset laskusäännöt kuin reaaliluvuille. Eroja voi syntyä mm. tilanteissa, joissa laskutoimituksen tulos "menee liukulukualueen ulkopuolelle" eli esiintyy ns. "ylivuoto" tai "alivuoto". Mm. nollalla jako johtaa ylivuotoon. Periaatteelliselta kannalta on kuitenkin tärkeää huomata, että eräin osin syntyy eroja vaikka ylivuotoa/alivuotoa ei esiintyisikään. Esimerkiksi pyöristysvirheet voivat johtaa tällaiseen tilanteeseen, kuten seuraavasta esimerkistä ilmenee.

1.16. Esimerkki. Reaaliluvuille $b, c > 0$ pätee $(b + c)/2 \leq \max\{b, c\}$. Liukuluvuille $C = 0.982$, $B = 0.984 \;\Rightarrow\; B + C = 1.97 \;\Rightarrow\; (B + C)/2 = 0.985$ (pyöristys 3:een desimaaliin). Siis liukulukuaritmetiikassa voi olla $(B + C)/2 > \max\{B, C\}$. Parempi laskutapa: $B + \frac{1}{2}(C - B)$. Ks. [KMN, s. 249].

1.17. Kone-epsilon. Kuten jo yllä kohdassa 1.15 todettiin, liukulukuesityksellä reaaliluku voidaan esittää koneesta riippuvalla tarkkuudella. Ns. *kone-epsilon* mittaa tätä tarkkuutta. Kone-epsilon meps on 2^{-p}, jos $c < c + 2^{-p}$ pätee liukulukuaritmetiikassa mutta $c < c + 2^{-p-1}$ ei päde ja p on positiivinen kokonaisluku sekä $c = 1$ (tavallisesti).

Turbo C:ssä meps on luokkaa 10^{-19} $(= 2^{-63})$. MATLABissa meps on luokkaa 10^{-16}.

```
/* e117.c   1 Sept. 1993 */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "writedat.c"
```

```
char *tabfile = "e117.dat";
void main()
{

        int c =1, p=1;
        FILE *fp;
        fp = fopen(tabfile, "w");    /* create e117.dat */
        fprintf(fp,"\n FILE %15s begins.",tabfile); writedate(fp);
        printf(" \n \n MACHINE EPSILON = \n \n");
        fprintf(fp," \n \n MACHINE EPSILON = \n \n");
        while (c+pow(2.0,-p) > c) p++;
        printf("  %16.6E  = 2^(-%3d) \n",pow(2,1-p), p-1);
        fprintf(fp,"  %16.6E  = 2^(-%3d) \n",pow(2,1-p), p-1);
        fprintf(fp,"\n FILE %15s ends.",tabfile); writedate(fp);
        fclose(fp);
        getch();
}
```

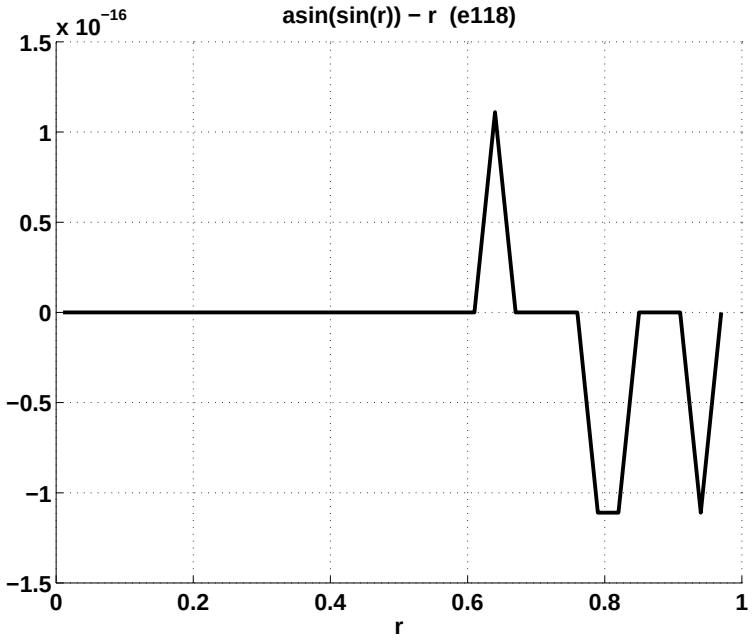
```
 FILE        e117.dat begins. 1. Sept. 1993 19:08:45


 MACHINE EPSILON =

       1.08420E-19  = 2^(- 63)

 FILE        e117.dat ends. 1. Sept. 1993 19:08:45
```

1.18. Ohjelma-epsilon. Kone-epsilonin ohella on syytä kiinnittää huomiota myös siihen, että numeeriset ohjelmat eivät nekään ole absoluuttisen tarkkoja. Viitteitä tarkkuudesta voi saada soveltamalla ohjelmaa tapauksessa, jossa tulokset voidaan laskea myös toisin.

Funktioparin sini–arkussini yhteistarkkuutta voi tutkia piirtämällä funktion $\operatorname{asin}(\sin r) - r$ kuvaajan, kun `r = 0.01:0.013:0.99`. Matemaattisesti funktio on identtisesti nolla, mutta numeerisesti syntyy luokkaa 10^{-16} olevia eroja. Funktiolle $\operatorname{atanh}(\tanh(r)) - r$ välillä `r=0:0.5:8` poikkeama on jo suurempi, luokkaa 10^{-11}.

```
% FILE  e118.m  begins.
clf;
```

```
r = 0.01: 0.03: 0.99;
y =asin(sin(r))-r;
axes('FontSize',[15],'FontWeight','bold');  hold on;
plt1=plot(r,y);
title(['asin(sin(r)) - r  (e118)']);
xlabel('r','FontSize',15);
set(plt1,'LineWidth', 2.5);
grid;
pause;
hold off;
% FILE  e118.m  ends.
```



1.19. Matemaattinen mallintaminen. Matemaattisella mallintamisella tarkoitetaan reaalimaailman ilmiön kuvaamista sellaisessa matemaattisessa muodossa, josta tarkasteltavat suureet voidaan ratkaista joko eksaktisti tai likimääräisesti. Usein kysymyksessä on monivaiheinen prosessi, jossa osina on mm. numeerisen mallin muodostaminen ja ratkaiseminen. Tavallisesti malli kuvaa ko. ilmiötä puutteellisesti ja sisältää yksinkertaistuksia. Numeerista mallia muodostettaessa tehdään lähes aina diskretointi. Tällöin "ääretön korvataan äärellisellä" ja jatkuva muuttuja diskreetillä.

(1) $\sum_{n=1}^{\infty} a_n = \lim_{p\to\infty} \sum_{n=1}^{p} a_n$; $\sum_{n=1}^{p}$ voidaan laskea

(2) $\int_a^b f(x)dx = \lim_{n\to\infty} \sum_{j=1}^{n} f(a + j\frac{b-a}{n}) \cdot \frac{b-a}{n}$; $\sum_1^n$ voidaan laskea

(3) $\prod_{j-1}^{\infty} b_j = \lim_{n\to\infty} \prod_{j=1}^{n} b_j$; $\prod_1^n$ voidaan laskea

Esimerkiksi differentiaaliyhtälöllä voidaan mallintaa monia fysiikan ilmiöitä heittoliikkeestä radioaktiiviseen hajoamiseen (matemaattinen malli). Saatu differentiaaliyhtälö muunnetaan diskreettiin muotoon esim. differenssimenetelmällä (numeerinen malli). Etsitty numeerinen ratkaisu saadaan ratkaisemalla numeerinen malli.

1.20. Virhelähteistä. Numeerisen laskennan virheet johtuvat usean erilaisen virhetekijän yhteisvaikutuksesta, joita nyt tarkastelemme.

1. *Mallivirheet:* Syy: yksinkertaistavat oletukset.
2. *Menetelmävirheet:* Syy: numeerinen malli ratkaistaan äärellisellä määrällä laskutoimituksia:
   - *katkaisuvirhe:* korvataan ääretön summa äärellisellä:
   $$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = \sum_{n=0}^{k} \frac{x^n}{n!} + R_{n+1}(x) \approx \sum_{n=0}^{k} \frac{x^n}{n!},$$
   kun $|R_{n+1}(x)|$ on pieni
   - *diskretointivirhe:* esim. $\int_0^1 = \sum f(x_i)\Delta x$
3. *Lähtöarvovirheet:* Esimerkiksi analysoitava data (kokeelliset mittaukset) on virheellinen.
4. *Pyöristysvirheet:* Numeerisen mallin ratkaisu voi sisältää suuren määrän laskutoimituksia, joista kukin voi tuottaa (koneesta riippuvan) virheen $\Rightarrow$ virheiden kumuloituminen laskennan aikana.
5. *Epästabiilit algoritmit:* Menetelmävirhe voi aiheutua siitä, että käytetään *epästabiilia* algoritmia.
6. *Inhimilliset virheet:* Ohjelmointivirheet (esim.).
7. *Jako 0:lla* voi johtaa arvaamattomiin seurauksiin.

1.21. Suhteellinen ja absoluuttinen virhe. Vektorin $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ approksimaatiota merkitään $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_n) \in \mathbb{R}^n$. Approksimaation virhe $v = x - \tilde{x}$ (tai $\tilde{x} - x$).
Virheen itseisarvo l. *normi* $\|v\|$:

$$\|v\|_\infty = \max\{\, |v_i| \,:\, 1 \le i \le n \,\}$$
$$\|v\|_1 = \sum\nolimits_{i=1}^{n} |v_i|$$
$$\|v\|_2 = (\sum\nolimits_{i=1}^{n} |v_i|^2)^{1/2}$$

Virhearvio: $\|x - \tilde{x}\| \le u$, missä u on virhearvio. Jos $x \neq 0$, niin *suhteellinen virhe* on $\frac{\|x-\tilde{x}\|}{\|x\|}$.
Tulos on ilmaistu tarkkuudella ε, jos $\|x - \tilde{x}\| \le \varepsilon$.
Tulos on ilmaistu p%:n tarkkuudella, jos suhteellinen virhe $< \frac{p}{100}$.

Yksinkertaisissa tapauksissa virheen vaikutusta lopputulokseen voidaan arvioida differentiaalilaskennan avulla.

1.22. Virheen arviointi derivaatan avulla. Arvioimme funktion $f(x) = x^p$ muutosta pisteessä $x = 1$, kun x:n muutos on Δx. Kun $y \in (x, x + \Delta x)$, on väliarvolauseen perusteella $f(y) - f(x) = f'(\xi)(y - x)$ jollekin $\xi \in (x, y)$. Näin ollen

$$|\Delta f| \le \sup\{\, |f'(z)| \,:\, z \in (x, x + \Delta x) \,\} \cdot |\Delta x| \,.$$

Koska $f'(z) = pz^{p-1}$, saamme arvion $|\Delta f| \le |2p|\,|\Delta x|$, kun $|\Delta x|$ on pieni.

1.23. Taylor-sarjan osasummat. Yksinkertainen ajatus funktion approksimoimiseksi on käyttää Taylor-sarjan osasummia. Yleensä nämä ovat varsin huonoja approksimaatioita. Tarkastellaan nyt esimerkkiä $f(x) = \sum_{p=0}^{\infty} \frac{x^p}{p!} = e^x$.

Taulukoimalla osasummien arvoja kun $x = -7.0$, todetaan esim. että osasummien arvot, kun $n = 2, \ldots, 16$, eivät sisällä yhtään oikeaa desimaalia! Approksimointiteoriassa johdetaan erilaisia parempia approksimointeja, mutta ei tällä kurssilla. Ks. myös [AS], [NR].

```
/* e123.C        2. Aug. 1993
   Study EXPONENTIAL function approximation by Taylor-series */
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "writedat.c"
char *tabfile = "e123.dat";
double TAYLORS1(x,n)
```

```
double x;
int n;
{
      double s= 1.0, t=1.0; int i;
      for(i=1;i<n+1;i++)
      {
        t=t*x/i;
        s=s+t;
      }
      return(s  );
}
void main(void)
{
      int i;
      double d,x=-7.0;
      double v1,v2;
      FILE *fp;
      fp = fopen(tabfile, "w");    /* create e128.DAT */
      fprintf(fp,"\n FILE %15s begins.",tabfile); writedate(fp);
      fprintf(fp,"\n Terms         Partial sum       Error ",
              tabfile);
      for(i=1;i<20;i++)
      {
        v1= TAYLORS1(x,2*i)  ;
        v2 = v1 - exp(x);
        printf("\n %3d    %16.6E     % 12.4E ",5*i,v1,v2);
        fprintf(fp,"\n %3d    %16.6E     % 12.4E ",2*i,v1,v2);
      }
      printf("\n The answer is = %16.6E ",exp(x));
      fprintf(fp,"\n The answer is = %16.6E ",exp(x));
      fprintf(fp,"\n FILE %15s ends.",tabfile); writedate(fp);
      fclose(fp);
}
```

```
 FILE        e123.dat begins 1. Sept. 1993 19:21:57

 Terms         Partial sum       Error
   2         1.85000E+01        1.850E+01
   4         6.13750E+01        6.137E+01
```

```
 6          8.47181E+01          8.472E+01
 8          6.42929E+01          6.429E+01
10          3.09318E+01          3.093E+01
12          1.02917E+01          1.029E+01
14          2.51196E+00          2.511E+00
16          4.69778E-01          4.689E-01
18          7.00927E-02          6.918E-02
20          9.18367E-03          8.272E-03
22          1.72977E-03          8.179E-04
24          9.79884E-04          6.800E-05
26          9.16703E-04          4.821E-06
28          9.12177E-04          2.949E-07
30          9.11898E-04          1.573E-08
32          9.11883E-04          7.377E-10
34          9.11882E-04          3.067E-11
36          9.11882E-04          1.133E-12
38          9.11882E-04          3.282E-14
The answer is =      9.11882E-04
FILE         e123.dat ends 1. Sept. 1993 19:21:58
```

1.24. Eulerin–Mascheronin vakio. Merkitään harmonisen sarjan osasummaa $H_n = \sum_{k=1}^{n} \frac{1}{k}$. Silloin Eulerin–Mascheronin vakio γ on

$$\gamma = \lim_{n\to\infty} (H_n - \log n)\,.$$

Konvergenssi on tässä piinallisen hidasta. Numeerisesti tehokkaampi laskutapa on

$$\gamma = \lim_{n\to\infty} (H_n - \log(n + \frac{1}{2}))\,,$$

kuten voidaan havaita yksinkertaisin testein. Todetaan, että pieni muutos algoritmissa parantaa oleellisesti algoritmin konvergenssia.

1.25. Merkitsevien numeroiden kumoutuminen. Samanmerkkisten lukujen vähennyslaskussa voi tapahtua merkitsevien numeroiden kumoutumista, joka olennaisesti huonontaa tarkkuutta. Seuraavassa muutamia esimerkkejä, joissa matemaattisella identiteetillä voidaan torjua tarkkuuden huononemista.

(1) $\sqrt{x^2+1} - 1 = \frac{x^2}{1+\sqrt{1+x^2}}$. Kun $x \sim 0$, on jälkimmäinen muoto parempi.

(2) $\tan(x) - \sin(x) = \frac{1}{2}x^3 + \frac{1}{8}x^5 + \frac{13}{240}x^7 + \cdots$. Jälkimmäinen muoto parempi kun $x \sim 0$.

(3) $ax^2 + bx + c = 0 \quad (a \neq 0,\ b^2 - 4ac \geq 0,\ b \neq 0)$

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\,.$$

Tarkkuutta huonontavaa kumoutumista voi tapahtua, jos $4ac << b^2$. Oikea laskutapa: $q = -\frac{1}{2}(b + \ \mathrm{sgn}(b)\sqrt{b^2 - 4ac})$, $x_1 = \frac{q}{a}$, $x_2 = \frac{c}{q}$.

(4) Positiivisten lukujen $0 < x_1 < \ldots < x_n$ yhteenlasku on edullista tehdä suuruusjärjestyksessä

```
s=0;
for (j=1;j<=n;j+1){s+=x[j];}
```

Numeerisen laskennan käytäntö on osoittanut, että kohtien (1)–(4) tyyppisillä keinoilla on tärkeä merkitys. Yhteenveto kuvion muodossa:

MATEMAATTINEN IDENTITEETTI	$\neq$	NUMEERINEN IDENTITEETTI

1.26. Numeerisen algoritmin stabiilius. Laskutoimitus tai algoritmi on *epästabiili* l. *häiriöaltis* l. *pahanlaatuinen* (unstable, sensitive, ill-conditioned), jos pienet virheet lähtötiedoissa vaikuttavat merkittävästi lopputulokseen. Muulloin laskutoimitus tai algoritmi on *stabiili* l. *hyvänlaatuinen* (stable).

Perusesimerkki epästabiilista ongelmasta on yhtälöryhmän $Ax = b$, missä A on $n \times n$ -matriisi, ratkaiseminen kun A on "huono". Yksikäsitteinen ratkaisu ei yleensä ole mahdollista, jos $\det(A) = 0$. Siis tällaiset (singulaariset) matriisit ovat tässä mielessä huonoja.

On todettu käteväksi mitata $n \times n$ -matriisin A "kuntoisuutta" reaaliluvulla $\mathrm{cond}(A)$. Tällöin tapauksessa $\det(A) = 0$ on $\mathrm{cond}(A) = \infty$. Tapauksessa $\det(A) \neq 0$ $\mathrm{cond}(A) \in [1, \infty)$, ja kuntoisuusluku määritellään A:n ns. *singulaariarvojen* avulla. Kuntoisuusluvun vaikutusta lineaarisen yhtälöryhmän ratkaisuun tutkitaan luvussa 4.

1.27. Esimerkki. Koodirivit

```
a=rand(20,20);
c=cond(a)
```

antavat satunnaismatriisin `a` kuntoisuusluvun `c`.

1.28. Esimerkki. Suorien $\begin{cases} y = x \\ y = (1+\delta)x - 1 \end{cases}$, missä $\delta = 10^{-p}$, kun $p = 2, 3, \ldots$, leikkauspiste on $(\frac{1}{\delta}, \frac{1}{\delta})$. Siis tehtävä on epästabiili.

1.29. Numeerisen matematiikan tavoitteet. Numeerisen matematiikan tavoitteena on tuottaa tehokkaita ja stabiileja algoritmeja, jotka mahdollisimman vähillä laskutoimituksilla antavat mahdollisimman paljon informaatiota. Yleensä on mahdollista löytää erikoistilanteita, joissa algoritmi ei toimi oikein. Numeerisessa matematiikassa joudutaan rajoittumaan tilanteisiin, jossa "kaikki käy hyvin": funktiot ovat riittävän säännöllisiä jne. Sovelluksissa tällaiset oletukset tehdään usein vaieten, implisiittisesti. Esimerkiksi funktion numeerisessa integroinnissa ja minimoinnissa täytyy olettaa, ettei funktio heilahtele liian nopeasti.

Toisaalta on muistettava, että liukulukuaritmetiikka tuo virheitä mukaan ja että esim. funktion nollakohta ei ole välttämättä määritelty meps:n tarkkuudella, vaan tilanne nollakohdan lähellä mepsin mittakaavassa tarkasteltuna voi näyttää hyvin rajusti oskilloivalta. Ks. kohta 5.16. Näin ollen 0-kohtia etsittäessä ei tarkkuusvaatimuksia saa asettaa liian tiukoiksi.

1.30. Mitä ohjelmilta voidaan vaatia?

- virhetilanteiden kontrolli ja virheilmoitus tarvittaessa
- helppokäyttöisyys, "luettavuus" (kommentit), dokumentointi
- stabiliteetti, tehokkuus, virheettömyys
- siirrettävyys toisiin käyttöjärjestelmiin, kieliin
- muunneltavuus (source code)

  Huom.: Käytännössä on lähes mahdotonta testein osoittaa ohjelmaa virheettömäksi (mutta voidaan ehkä löytää virheitä).

2 MATLABin perusteet

Luvussa 1 olemme jo nähneet muutamia esimerkkejä MATLAB-kielen käytöstä. Tarkoituksemme on aloittaa nyt systemaattinen perehtyminen kielen peruspiirteisiin. Edellisessä luvussa nopeasti esitetyt seikat tulevat nyt kerrattaviksi helpommin ymmärrettävässä muodossa.

MATLAB on C. Molerin [Mo] kehittämä numeriikan ja numeerisen matriisilaskennan kieli, joka on saavuttanut vakiintuneen aseman tieteellisen laskennan alueella. MATLABin perusteista on olemassa joukko hyviä oppaita, esim. [Sig] (tai [K]). Sen sijaan ohjelman mukana tuleva käsikirja ei ole tarkoitettu oppikirjaksi. MATLABin demo-ohjelmaa läpikäymällä pääsee näkemään MATLABin piirteet, ja sen lähdekoodi on kätevä opas ohjelmointiin. Kustakin MATLAB-käskystä saa myös kuvauksen help-komennolla, esim. `help plot`.

Katsomme alla muutamia esimerkkejä tyypillisistä MATLAB-istunnoista. Ohjelman toiminnan oppiminen tapahtuu tehokkaimmin, kun lukija samalla kirjoittaa po. istuntojen komennot päätteellä ja toteaa ohjelman tulosteen. Oletamme, että lukijalla on käytettävissään jokin MATLAB-ohjelman perusopas, esim. [Sig].

Mainitsemme kahdesta ominaisuudesta, jotka tulevat toistuvasti esille, nimittäin että (a) MATLAB on tulkki ja (b) MATLAB on matriisikieli.

Kohdan (a) mukaisesti interaktiivisen MATLAB-istunnon tapahtumat kirjautuvat ns. "työtilaan" (workspace), jolla tarkoitetaan MATLAB-tulkissa määriteltyjä muuttujia.

Kohdasta (a) seuraa mm., että muuttujien arvot "pysyvät muistissa" ts. ne ovat käytettävissä työtilassa ja että niiden nykyiset arvot saadaan esille kirjoittamalla muuttujan nimi. Työtilan muuttujat saadaan esiin komennolla `who`. Virheiden välttämiseksi on syytä aika ajoin pyyhkiä tarpeettomat muuttujat muistista komennolla `clear`. Kohta (b) on tehokkaan käytön kulmakivi: aikaa vievät ohjelmointisilmukat voidaan tavallisesti korvata yksinkertaisilla vektori- tai matriisioperaatioilla. Erityisesti sisäkkäisiä silmukoita pitäisi välttää. MATLABin matemaattiset funktiot (esim. sin, cos, log, exp) hyväksyvät myös matriisiargumentin. Käyttäjän on omia funktioita määritellessään pyrittävä samaan.

MATLABin käynnistys tapahtuu komennolla "`matlab`", lopetus komennolla "`quit`", käyttöjärjestelmän komennon toteutus "!huutomerkkitekniikalla", esim. "`!dir`", "`!emacs`", sekä istunnon talletus tekstitiedostoon "`diary istunto.doc`" (annettava alussa). Palautamme luvusta 1 mieleen, että m-tiedoston fname.m lukeminen työtilaan ja siellä olevien käskyjen toteutus tapahtuu komennolla `fname`. Oletamme alla, että MATLAB on käynnistetty.

2.1. Keskiarvo. Olkoon $f(x) = \sin(x) + \cos(x^2)$. Laske $\frac{1}{100}\sum_{j=1}^{100} f(j)$.
Ratkaisu. Generoidaan funktio f:

```
!emacs f201.m
```

Kirjoitamme tiedostoon f201.m seuraavat 2 riviä:

```
function y=f201(x)
y=sin(x)+cos(x.^2);  <— puolipiste estää tulostuksen,
                        . on alkioittainen operaatio
```

(Vanhempia MATLABin versioita käyttäessämme laittaisimme tähän tiedostoon vielä kolmannen rivin "`end;`".) Tiedoston talletuksen jälkeen muodostetaan haluttu keskiarvo seuraavasti:

```
x=1:1:100; y=f201(x); m = length(x);
aver=(1/m)*sum(y)  <— ei puolipistettä, siis tulostuu
```

Huomaa, että `sum` on tässä MATLABin sisäinen vektorioperaatio. Perehdy komennon `help sum` avulla sen toimintaan matriisiargumentin tapauksessa.

Toinen tapa keskiarvon laskuun perustuu MATLABin sisäisen funktion `mean` käyttöön.

2.2. Funktion arvojen taulukointi. Talleta Γ-funktion arvot $\Gamma(0.5*j)$, $j = 1, 2, \ldots, 10$, tiedostoon muodossa

$$\begin{array}{l} 0.5000\ \Gamma(0.5) \\ 1.0000\ \Gamma(1.0) \\ 1.5000\ \Gamma(1.5) \\ \quad \ldots\ \ldots \end{array} .$$

Ratkaisu. Muodostetaan `x=0.5:0.5:5` ja `y=gamma(x)`.

Allaolevassa m-tiedostossa esitetään kolme eri ratkaisutapaa.

```
% FILE  e202.m  begins.
x = 0.5:0.5:5;  y = gamma(x);
data=[x' y'];
delete e202a.dat
save  e202a.dat data  -ascii   % TAPA 1
delete e202b.dat
diary e202b.dat
disp(data)
diary off                      % TAPA 2
delete e202c.dat
fid = fopen('e202c.dat','w')
fprintf(fid,'%8.4f    %12.6e \n', data');
fclose(fid);                   % TAPA 3
% FILE  e202.m  ends.
```

2.3. Polynomiyhtälön juuret. Etsi polynomiyhtälön $\sum_{j=1}^{p} c_j \cdot x^{p-j} = 0$ juuret.

Ratkaisu: Muodostetaan ensin jokin satunnaisvektori, esim. `c=rand(1,10)`. Polynomiyhtälön juuret saadaan komennolla `z=roots(c)`.

2.4. Matriisiargumentti. Edellä olemme jo tutustuneet MATLABin tyypillisiin alkioittaisiin operaatioihin, joista tärkeimmät muodostavat joukon {`+`, `-`, `.*`, `./`, `.^`}. Tavallisimmat alkeisfunktiot, kuten trigonometriset ja hyperboliset funktiot, on toteutettu matriisiargumenteille. Näin voimme helposti todeta esim. tutun identiteetin voimassaolon:

```
x= rand(3,3);
z= sin(x).^2 + cos(x).^2 -1;
disp(z)
  0     0     0
  0     0     0
  0     0     0
```

Huomaa erityisesti, että yllä `1` on reaaliluku, kun taas `z` ja `x` ovat matriiseja. Tässä tilanteessa MATLAB osaa automaattisesti "laajentaa skalaarin" `1` oikeankokoiseksi matriisiksi. Keskenään erikokoisille matriiseille ei peruslaskutoimituksia ole määritelty.

Versiota 4.0 edeltävissä MATLABin versioissa oli tässä kohden eroja. Joskus voi olla syytä selvyyden vuoksi tuoda esille matriisin koko, vaikka MATLABin syntaksi ei sitä nykyään vaatisikaan. Tämä voidaan tehdä `size`-funktion avulla seuraavasti:

```
x= rand(2,2);
one= ones(size(x));
one./x

ans =

    1.1303    2.2914
    3.6669    1.3046
```

Matriisi voidaan muuttaa vektoriksi ja takaisin matriisiksi näin:

```
   x= rand(3,4);
   xx= x(:)'

xx =

  Columns 1 through 7

    0.2190    0.0470    0.6789    0.6793    0.9347    0.3835    0.5194

  Columns 8 through 12

    0.8310    0.0346    0.0535    0.5297    0.6711

   x2 = reshape(xx,2,6)

x2 =

    0.2190    0.6789    0.9347    0.5194    0.0346    0.5297
    0.0470    0.6793    0.3835    0.8310    0.0535    0.6711
```

2.5. Satunnaiskulku. Satunnaisesti liikkuvan partikkelin sijainnin määräävät koordinaatit x ja y noudattavat normaalijakaumaa. Piirrä kuva partikkelin liikkeestä 200 askeleen aikana.

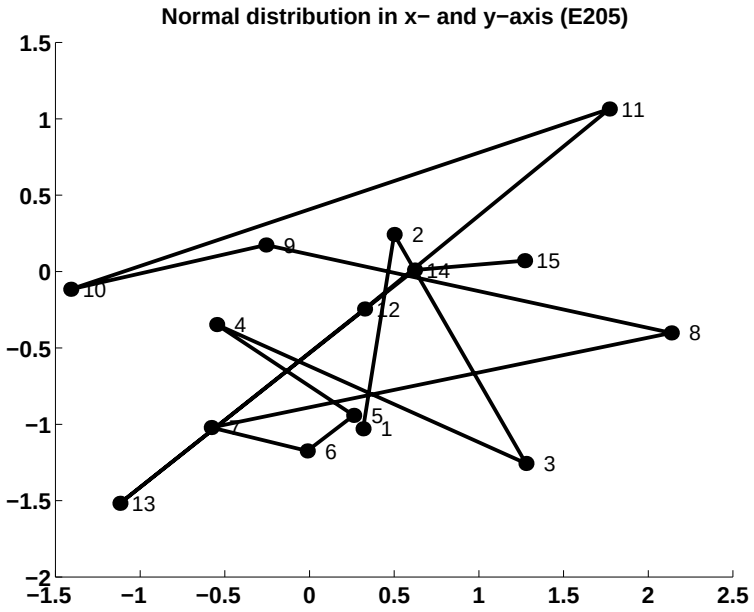
Ratkaisu.

```
x=randn(1,200); y=randn(1,200);
plot(x,y);
```

Pisteiden nimikointi järjestysnumerolla onnistuu seuraavasti:

```
% FILE  e205.m  begins.
clf;
n = 15;
ypt =randn(1,n);
xpt = randn(1,n);
mer = [];
for i =1:n
     nro = num2str(i);
     if (i<10)
       nro =[blanks(1) num2str(i)];
```

```
    end;
    mer = [mer; nro ];
end;
axes('FontSize',[15],'FontWeight','bold');  hold on;
info =['Normal distribution in x- and y-axis'  ];
xpt1 =xpt +0.02*(max(xpt)-min(xpt))*ones(1,n);
% xpt1 ohjaa pisteiden numerot (text alla) hieman oikealle niin,
% etteivat numerot osu pisteiden paalle
plt1=plot(xpt,ypt,'k.',xpt, ypt,'MarkerSize',30);
text(xpt1, ypt, mer,'FontSize',[14]);
title([info ' (E205)'],'FontSize',[15]);
set(plt1,'LineWidth', 2.5);       % Line width is changed
hold off;
% FILE  e205.m  ends.
```



2.6. Vektorin pienin alkio. Etsi vektorin x "pienin komponentti", ts. jokin sellainen x:n komponentti $x(j)$ että $x(j) \leq x(k)$ kaikilla k, ja tätä komponenttia vastaava indeksi.

Ratkaisu.

```
x=rand(1,1000);
[xmin,imin]=min(x);
```

Tällöin ko. pienin arvo on `xmin=x(imin)`. Komento `min(y)` on määritelty myös matriiseille `y`, ks. `help min`.

Vektorin lajittelu suuruusjärjestykseen voidaan tehdä seuraavasti:

```
x=sin(1:0.5:30);
y=sort(x);
```

2.7. Esimerkki. "Salaperäinen matriisi". Oletamme että olemme konstruoineet 5×5 -matriisin a, mutta emme voi kirjoittaa sitä suoraan näkyviin. Tehtävänä on rekonstruoida matriisi a käyttäen apuna pelkästään tuloja $a * x$ missä x on vektori.

Ratkaisu.

```
a=magic(5);
b=ones(5,5);
for j=1:5
    e=zeros(5,1); e(j,1)=1;   % yksikkovektori  e_j
    b(:,j)=a*e;
end;
```

Matriisien a ja b samuus voidaan todeta, paljastamatta a:n alkioita, tulostamalla

```
norm(a-b)
```

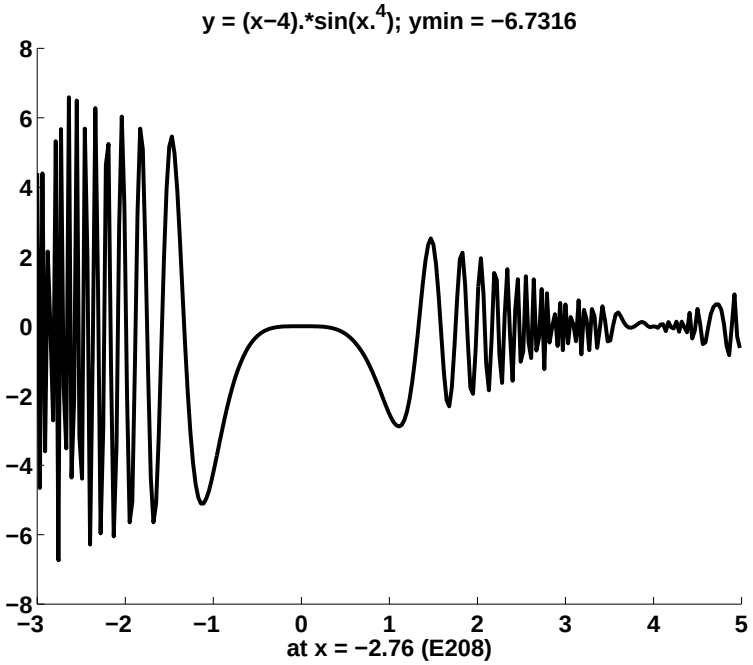
jolloin saadaan vastaus 0.

2.8. Funktion minimikohta. Etsi kunkin funktion $f_j(x) = (x - j) * \sin(x^j)$ minimikohta välillä $[-3, 5]$ ja piirrä kuva funktiosta f_j $(j = 1, \ldots, 5)$.

Ratkaisu: Alla olevassa ratkaisussa esiintyy useita hyödyllisiä seikkoja, kuten ohjelmointisilmukka, laskutulosten esittäminen kuvan otsikkotekstissä (`num2str` ja `title`) sekä `pause`-käsky.

```
% FILE  e208.m  begins.
for j=1:5
    clf;
    x=-3: 0.03: 5;
    y = (x-j).*sin(x.^j);
    [ymin, imin] = min(y);
    txt = ['y = (x-' num2str(j) ').*sin(x.^' num2str(j) ];
    txt = [txt  '); ymin = ' num2str(ymin) ];
    axes('FontSize',[15],'FontWeight','bold');  hold on;
    plt1=plot(x,y,x(imin),ymin,'k.');
```

```
    title(txt,'FontSize',[15]);
    xlabel(['at x = ' num2str(x(imin)) ' (E208)'], ...
            'FontSize',[15] );
    set(plt1,'LineWidth', 2.5);       % Line width is changed
    hold off;
    pause;
end;
% FILE  e208.m  ends.
```



2.9. Muuttujien oletusarvot. MATLABissa on mahdollista antaa funktioiden argumenteille oletusarvoja. Tarkastellaan hypergeometrisen sarjan osasummien laskemista siten, että käyttäjä voi ilmoittaa mukaan otettavien termien määrän, vrt. esimerkki 1.13. Käskyllä

```
y=fabcn(a,b,c,r,nmax)
```

lasketaan `nmax` ensimmäisen termin summa. Tämä voidaan tehdä "systeemimuuttujan" `nargin` avulla, joka ilmoittaa funktiokutsussa olevien argumenttien lukumäärän.

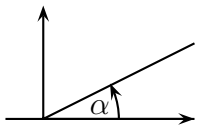
```
function y = fabcn(a,b,c,r,nmax)
% Compute 2F1(a,b;c;r)
```

```
% The Gauss hypergeometric function F(a,b;c;r)
if ( (size(a) ~= 1) | (size(b) ~= 1) | (size(c) ~= 1) )
   error('Parameter error in FABC.m');
end;
[m,n] =size(r);
one =ones(m,n);
s=one;
g= 1.0; p = 100;
if (nargin == 5)
    p= nmax;
end;
for j=1:p,
        g=g*(a+j-1)*(b+j-1)/(j*(c+j-1));
        s = s+ g*(r.^j);
end;
y = s;
return;
% FILE  fabcn.m  ends.
```

2.10. Kompleksiluvut. Kompleksilukujen joukon $\mathbb{C}$ alkioita merkitään $z = x + iy$, missä $i = \sqrt{-1}$ ja $x, y \in \mathbb{R}$. Kompleksiluvuille on [LP]:ssä määritelty summa, erotus, tulo ja osamäärä, joille pätevät laskusäännöt on myös esitelty siellä. Määritellään liittoluku $\overline{z} = x - iy$, jolloin

$$z\overline{z} = x^2 + y^2\,, \quad \operatorname{Re} z = \frac{1}{2}(z + \overline{z})\,, \quad \operatorname{Im} z = \frac{1}{2i}(z - \overline{z})\,.$$

Esitys napakoordinaateissa:



$$z = r(\cos\varphi + i\sin\varphi) = |z|(\cos(\arg z) + i\sin(\arg z))\,, \quad |z| = \sqrt{z\overline{z}}\,.$$

Palautetaan mieliin perusrelaatiot:

$$\begin{aligned} \arg(z_1 z_2) &= \arg z_1 + \arg z_2 \\ \arg(z_1/z_2) &= \arg z_1 - \arg z_2 \end{aligned}$$

$$z^n = r^n(\cos n\varphi + i \sin n\varphi) \quad (\textit{de Moivre}\text{'n kaava 1667-1754})$$

Kompleksilukujen geometrista esitystä xy-tasossa sanotaan *Argandin* esitykseksi (1768-1822).

2.11. Potenssisarjat kompleksialueella. Teoksessa [LP] funktiot ovat yleensä määriteltyjä $\mathbb{R}$:n osaväleillä. Jos funktiolla on esim. Taylor-sarja

$$f(x) = \sum_{n=0}^{\infty} a_n (x - x_0)^n \,, \tag{2.12}$$

niin voidaan määritellä ainakin formaalisti (tarkoittaa: varmistumatta siitä, onko kaava mielekäs ja suppeneeko sarja) kompleksiarvoille z

$$f(z) = \sum_{n=0}^{\infty} a_n (z - x_0)^n \,. \tag{2.13}$$

Monien funktioiden tapauksessa voidaan esittää perustelu sarjan (2.13) suppenemiselle myös kompleksitason jossakin osa-alueessa; tällaisen funktion määrittelyalue voidaan siis laajentaa reaaliakselin ulkopuolelle. Esimerkkejä ovat mm. trigonometriset, hyperboliset, eksponentti- ja logaritmifunktiot sekä monet muut muotoa (2.12) olevat funktiot. Emme käy näitä laajemmin esittelemään, toteamme vain että

$$\log z = \log |z| + i \arg z \,, \tag{2.14}$$

kun $z \in \mathbb{C}$, ja että

$$e^{x+iy} = e^x(e^{iy}) = e^x(\cos y + i \sin y) \quad (\textit{Euler}, \text{ 1707-1783}), \tag{2.15}$$

kun $x, y \in \mathbb{R}$.

MATLAB-merkinnät:
$$\begin{cases} \arg z \to \texttt{angle(z)}; & \mathrm{Re}(z) \to \texttt{real(z)} \\ |z| \to \texttt{abs(z)}; & \mathrm{Im}(z) \to \texttt{imag(z)} \\ \overline{z} \to \texttt{conj(z)}; & (\texttt{angle} \in (-\texttt{pi}, \texttt{pi})). \end{cases}$$

2.16. Kompleksiaritmetiikkaa MATLABissa. MATLABin perustietotyyppi on kompleksinen matriisi. Tästä seuraa mm., että useimmat alkeisfunktiot (esim. `sin`, `asin`, `cosh`) hyväksyvät argumentikseen kompleksisen matriisin. Versiossa 3.5 ennen kompleksilukujen käyttöä on määriteltävä `i=sqrt(-1)`. Nykyversiossa sekä `i` että `j` ovat käytössä tähän tarkoitukseen ilman tällaista määrittelyäkin.

2.17. Esimerkki. Tutki satunnaisilla 10×10 -matriiseilla, päteekö MATLABissa identiteetti

$$z = \exp(\log z) \quad (z \in \mathbb{C}) .$$

Ratkaisu:

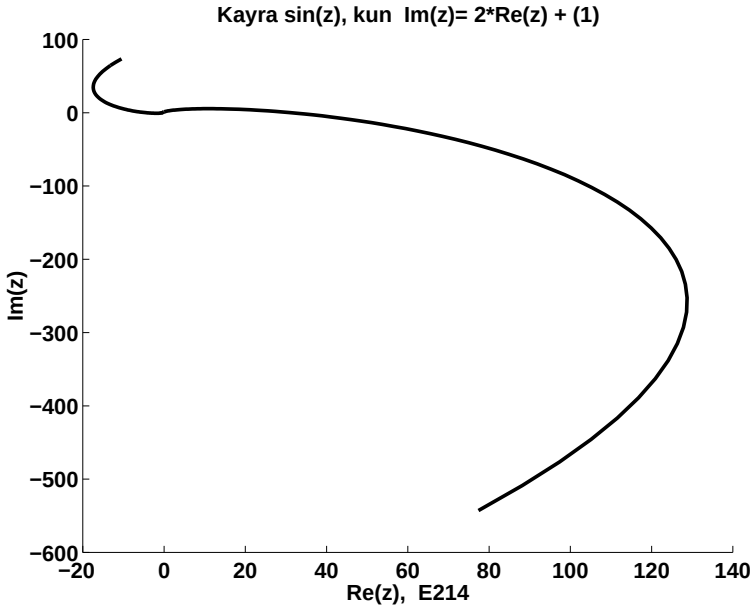
```
while (1==1)
   a=rand(10,10); b=rand(10,10); i=sqrt(-1);
   z=2*a+i*b;
   w=z-exp(log z);
   check=norm(w), pause;
end;
```

MATLABissa on useita erilaisia matriisinormeja, ks. komennolla `help norm`. Edellä `norm(w)` on matriisin `w` suurin singulaariarvo (vrt. luku 4).

2.18. Input. Suoran kuvautuminen $\sin z$:ssa. Kompleksitason sinifunktio kuvaa alueen alueelle (tämä vaatii perustelun; ks. funktioteorian kurssi) ja suoran jollekin käyrälle. MATLABissa voimme selvittää asiaa seuraavasti.

```
% FILE  e214.m  begins.
i = sqrt(-1);
while(1==1)
        clf;
        a = input('Enter a real number a for a*x + b : ');
        b = input('Enter a real number b for a*x + b : ');
        x = -3: 0.03: 3;  y = a*x+b;
        z = x+ i*y;
        w = sin(z);
        plot(x,y);
        title(['Suora ' num2str(a) '*x + ' num2str(b) ]), pause;
        clf;
        x1 = real(w); y1 =imag(w);
        axes('FontSize',[15],'FontWeight','bold');  hold on;
        plt1=plot(x1,y1);
        title(['Kayra sin(z), kun  Im(z)= ' num2str(a) ...
               '*Re(z) + (' num2str(b) ')'],'FontSize',[15]);
        xlabel('Re(z),  E214','FontSize',[15]);
        ylabel('Im(z)','FontSize',[15]);
        set(plt1,'LineWidth', 2.5);       % Line width is changed
        hold off;
```

```
        %print -deps e214
        pause;
end;
% FILE  e214.m  ends.
```

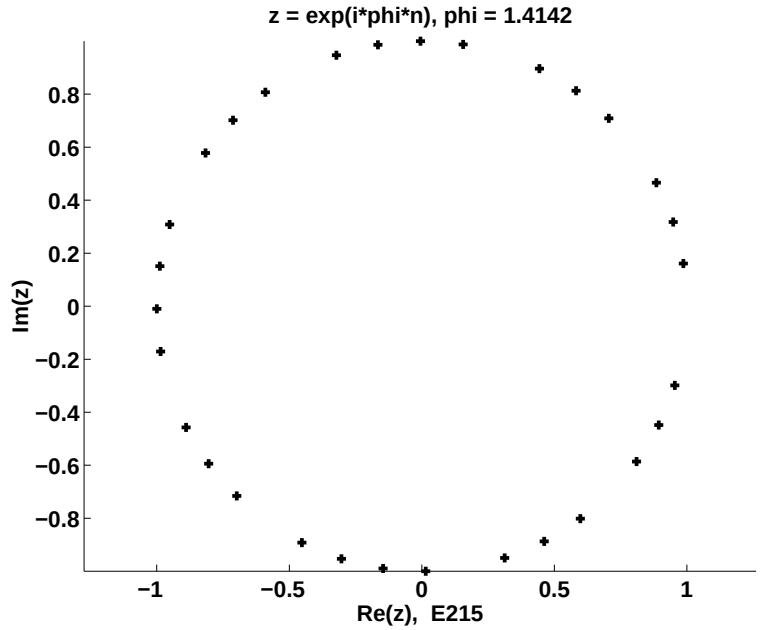


2.19. Esimerkki. Piirretään pisteet $e^{in\varphi}$ tasoon, kun `n=1:30`.

Ratkaisu. Huomaa alla olevassa koodissa esiintyvä `input`-käsky syötteen saamiseksi käyttäjältä.

```
% FILE  e215.m  begins.
i = sqrt(-1); x =[]; y =[];
phi = input('Enter phi in (0, 2*pi): ');
for n=1:30
    z = exp(i*n*phi);
    x = [x, real(z)]; y = [y, imag(z)];
end;
clf;
axes('FontSize',[15],'FontWeight','bold'), axis equal, hold on;
plt1=plot(x,y,'k+');
title(['z = exp(i*phi*n), phi = ' num2str(phi)],'FontSize',[15]);
xlabel('Re(z),  E215','FontSize',[15]);
ylabel('Im(z)','FontSize',[15]);
set(plt1,'LineWidth', 2.5);       % Line width is changed
hold off;
```

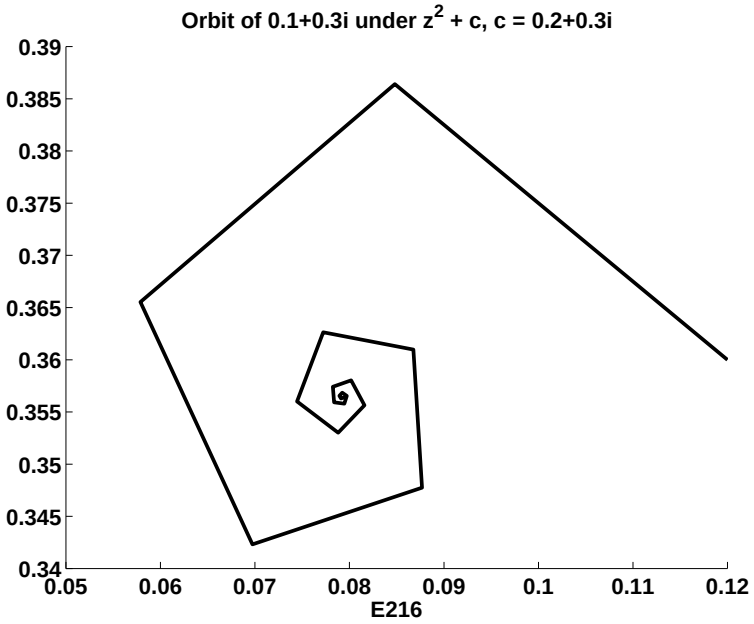
```
pause;

% FILE  e215.m  ends.
```



2.20. Kvadraattisen polynomin iterointi. Olkoon $f(z) = z^2 + c$ ja $z_0 \in \mathbb{C}$, $z_{p+1} = f(z_p)$. Joukkoa $\{z_k\}$ sanotaan pisteen z_0 *radaksi*. Radan lasku käsin on työlästä. MATLABilla saadaan nopeasti kuva tilanteesta.

```
% FILE  e216.m  begins.
i = sqrt(-1); orb =[];
z = input('Enter a complex number z: '); z0 =z;
c = input('Enter a complex number c: ');
n = input('Enter a positive integer n: ');
for k=1:n
    z = z.^2 +c;
    orb= [orb z];
end;
orb, pause;
```

```
clf;
axes('FontSize',[15],'FontWeight','bold');  hold on;
txt = ['Orbit of ' num2str(z0)];
txt = [txt ' under z^2 + c, c = ' num2str(c)];

plt1=plot(real(orb), imag(orb));
title(txt,'FontSize',[15]);
xlabel('E216','FontSize',[15]);
set(plt1,'LineWidth', 2.5);       % Line width is changed
hold off;

pause;
% FILE  e216.m  ends.
```



2.21. Julian joukon piirto. Kun $c \in \mathbb{C}$ on annettu, piirrämme niiden pisteiden z joukon, joiden rata iteroinnissa $z \mapsto z^2 + c$ pysyy ympyrän $|z| < 2$ sisällä. Tällaisten pisteiden joukkoa sanotaan polynomin $z^2 + c$ *Julian joukoksi*.

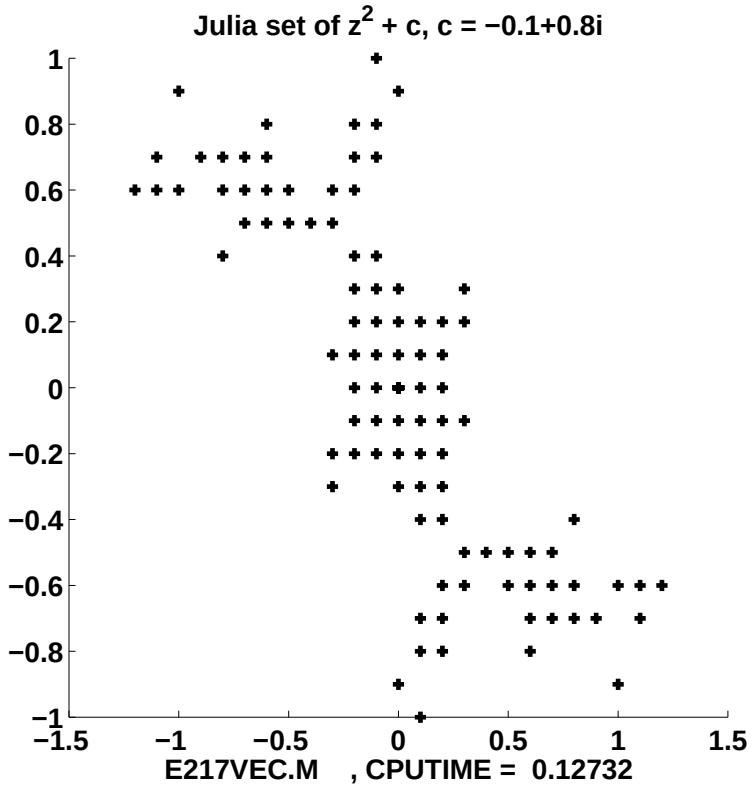
Ratkaisu: Huomaa alla olevassa ratkaisussa muuttujan `jul` alustus `jul=[]`.

```
% FILE  e217.m  begins.
% Draw Julia set . From Marcus: Matrices and Matlab p. 74
% A vectorized 10 times faster version is in e217vec.m
% The slowness is caused by the nested for loops
% which can be avoided, if vectorization is used
% Try c = -1 and c = -0.1+ 0.8*i
  c = input ('enter a complex number c = c1 + i * c2  ');
                tic;
                jul = [];
                for x = -2:.05:2
                        for y = -2:.05:2
                                z = x + i * y;
                                b = z;
                                n = 1;
                                while ((n <= 20) & (abs(b) <= 2))
                                        b = b^2+ c;
                                        n = n + 1;
                                end
                                if n == 21
                                        jul = [jul z];
                                end
                        end
                end
  axis('square')
  tempus=toc;
  txt =[num2str(c)];
  if isempty(jul)==1
        error('     Your c is not suitable!');
  else
        plot(jul, '+');
  end;
  title(['Julia set of z^2 + c, c = ' txt]),
  xlabel(['E217   , CPUTIME = ' num2str(tempus) ]),pause;
% FILE  e217.m  ends.
```

Yleisesti voidaan todeta, että erityisesti kaksinkertaisia ohjelmasilmukoita pitää välttää MATLABissa, koska ne ovat hitaita. Silmukat voidaan usein korvata käyttämällä matriisioperaatioita viittaamatta matriisiin alkioihin. Seuraavassa on nopeampi matriisioperaatioihin pohjautuva versio edellisestä.

```
% FILE  e217vec.m  begins.
```

```
% Try c = -1 and c = -0.1+ 0.8*i
  clear;
  c = input ('Enter a complex number c = c1 + i * c2:  ');
  tic;
  x0 = 0;y0 = 0;
  x1 =x0-2.0;x2 =x0+2.0; dx =(x2-x1)/40;  x= x1:dx: x2;
  y1 =y0-2.0;y2 =y0+2.0; dy =(y2-y1)/40;  y= y1:dy: y2;
  m = length(y);
  n = length(x);
  onex = ones(1,n);
  one = ones(m,n); i =sqrt(-1);
  z=[];
  for p=1:m
      z =[z; x+i*y(m+1-p)*onex];
  end;
  zsta=z;
  for q =1:21
      z=z.^2+c*one;
      test = (abs(z) <2*one);
      zsta = test.*zsta;
  end;
  w = zsta(:);   % Tama muuntaa matriisin vektoriksi !
  clf;
  axes('FontSize',[15],'FontWeight','bold');  hold on;
  axis('square')
  txt =[num2str(c)];
  plt1=plot(w, '+');
  title(['Julia set of z^2 + c, c = ' txt],'FontSize',[15]);
  xlabel(['E217VEC.M    , CPUTIME =  ' num2str(toc) ], ...
          'FontSize',[15]);
  set(plt1,'LineWidth', 2.5);       % Line width is changed
  hold off;
  pause;
% FILE  e217vec.m  ends.
```



2.22. Valikko. Tee yksinkertainen tekstipohjainen valikko, jolla voit toteuttaa jonkin annetuista m-tiedostoista (esim. laskuharjoitustehtävän).

Ratkaisu. Otamme mallia MATLABin omasta demo-ohjelmasta, jonka valikko koostuu seuraavista osista: (a) käynnistystiedosto, esim. h1.m; (b) valintojen luettelon sisältävä tiedosto, esim. h1lst.m; (c) eri valintoja vastaavat m-tiedostot, esim. h0101.m, h0102.m, h0103.m.

Alla esimerkkejä tiedostoista h1.m, h1lst.m, h010*.m.

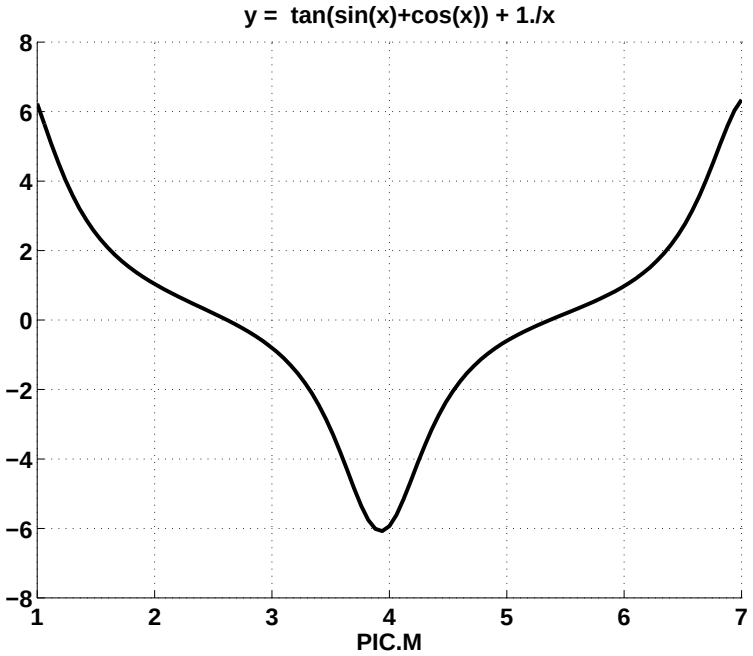
```
% FILE  h1.m  begins.
% Simple menu for Matlab
% Based on MATLAB's demo.m, demolist.m
help h1
% shows "help lines" of h1.m
pause
    help    h1lst
    disp('Choose from these')
    n = input(' Enter number 0 .. 6: ');
```

```
    demos= ['h0101 '
            'h0102 '
            'h0103 '
            'h0104 '
            'h0105 '
            'h0106 '];
    if ((n<= 0) | (n >6) ) break
    end;
    demos = demos(n,:);
    eval(demos) ;
    clear
% FILE  h1.m  ends.

% FILE  h1lst.m  begins.
%
% Mallina MATLABin oma demo: Tiedostot demo.m ja demolist.m
%
%        ------- h1   -------
%
%        1) Teht.1
%        2) Teht.2
%        3) Teht.3
%        4) Teht.4
%        5) Teht.5
%        6) Teht.6
%
%        0) Lopeta
%
%  FILE  h1lst.m  ends.

%FILE  h0101.m  begins.
x = 1:10;  y =sin(x);
plot(x,y), pause;
delete h0101.dat
diary h0101.dat
disp([x' y']);
diary off;
%FILE  h0101.m  ends.
```

2.23. Esimerkki. Kohdassa 2.22 esitetty valikko perustui mm. `eval`-käskyn käyttöön. Tämän kätevän käskyn toisena sovelluksena on seuraava yk-

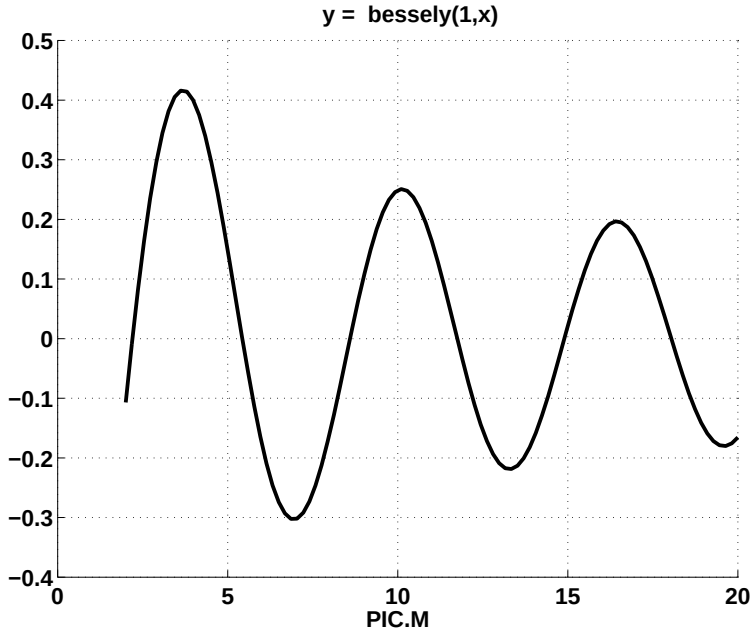
sinkertainen funktion kuvaajan piirto-ohjelma. Komennoilla `pic('tan(sin(x)+ cos(x)) + 1./x '), pic('bessely(1,x)')` piirretään funktioiden $\tan(\sin(x)+\cos(x)) + \frac{1}{x}$ ja bessely $(1, x)$ kuvaajat.

```
function y = pic(fexpr)
% PIC plots the graphs of functions: MATLAB 5.3 (R11)
% USAGE: pic(fexpr) where fexpr is a MATLAB function expression
% Example: pic('sin(tan(x)) ')
% N.B. The argument name must be x.
% SAVES data in pic.dat
clf;
disp([' Enter now argument bounds x1, x2 for ' fexpr]);
t ='Please enter'; x1=input([ t ' x1: ']);
x2=input([ t ' x2: ']); x =x1: (x2-x1)/100: x2;
% N.B. x is a global variable name "sin(x)"
eval(['f = ' fexpr ';']);
axes('FontSize',[15],'FontWeight','bold'); hold on;
fig=plot(x,f); grid
title(['    y =  ' fexpr],'FontSize',[15],'FontWeight','bold');
xlabel('PIC.M','FontSize',[15],'FontWeight','bold');
delete pic.dat;            diary pic.dat;
disp(['f = ' fexpr]);
disp( ['     x       f     '] );
disp(['--------------------']);
disp([x' , f']);   diary off;
set(fig,'LineWidth',2.5);
y ='pic.m completed and data saved in pic.dat';
% FILE  pic.m  ends.
```



Näyte ohjelman `pic` luoman tiedoston sisällöstä:

```
f = tan(sin(x)+cos(x)) + 1./x
     x       f
--------------------
    1.0000    6.2272
    1.0600    5.6450
    1.1200    5.0693
    1.1800    4.5284
       .         .
       .         .
       .         .
    6.8200    5.0784
    6.8800    5.5896
    6.9400    6.0312
    7.0000    6.3431
```



2.24. Välinpuolitusmenetelmä. Yhden muuttujan jatkuvan funktion nollakohta voidaan etsiä tunnetulla välinpuolitusmenetelmällä. Alla yksi `eval`-käskyyn pohjautuva toteutus. Kohdassa 5.17 esitetään virhearvio välinpuolitusmenetelmälle.

```
function y = bisect(fname,a,b,times)
%BISECT a and b must be a bracketing pair of zero
%       (that is, f(a) f(b) < 0 )
%       of real numbers or equally long vectors
%       fname is a string such that eval(['y =' fname ])
%       gives for y the value at x.
[m,n] = size(a);   one = ones(m,n);
if nargin ==4
    count = times;
end;
if nargin == 3
    count=10;
end;
for i=1:count
    c = 0.5*(a+b);
    x =c; eval(['fc = ' fname ';']);
    x =a; eval(['fa = ' fname ';']);
    x =b; eval(['fb = ' fname ';']);
```

```
    if any( fa.*fb >0)
       disp('Bracketing error in bisect ');
       disp(['norm = ' num2str(norm(a-b))]);
       break;
    end;
    booleb = (fa.*fc > 0);
    boolea = one - booleb;
    a = boolea.*a + booleb.*c;
    b = booleb.*b + boolea.*c;
    a-b;
    norm(a-b);
    y= a;
end;
% FILE  bisect.m  ends.
```

Toiminta:

```
>> bisect(' cosh(x)-4',0, 3,45)

ans =
    2.0634

>> cosh(ans)-4

ans =
  -1.4033e-13
```

2.25. Parametrin sovitus. Lähtökohtana on mittausaineisto (x_j, y_j) $(j = 1, \ldots, m)$, johon halutaan sovittaa muotoa $y = f(x, \lambda)$ oleva "malli", missä $\lambda = (\lambda_1, \ldots, \lambda_p)$. Tavallisesti p on pieni lukuun m verrattuna. Haluamme siis löytää vektorin $\bar{\lambda}$. jolle luvut $f(x_j, \bar{\lambda})$ liittyvät mahdollisimman hyvin lukuihin y_j pns-mielessä. Muodostamme kohdefunktion

$$s(\lambda) = \sum_{j=1}^{m} (y_j - f(x_j, \lambda))^2 \,,$$

jolle halutaan löytää mahdollisimman pieni arvo. Tehtävä on epätriviaali. Tavallisimmat minimointimenetelmät löytävät kohdefunktiolle vain lokaalin minimin

globaalin minimin asemesta. Jos minimiarvoja on olemassa, niiden ei tarvitse olla yksikäsitteisiä.

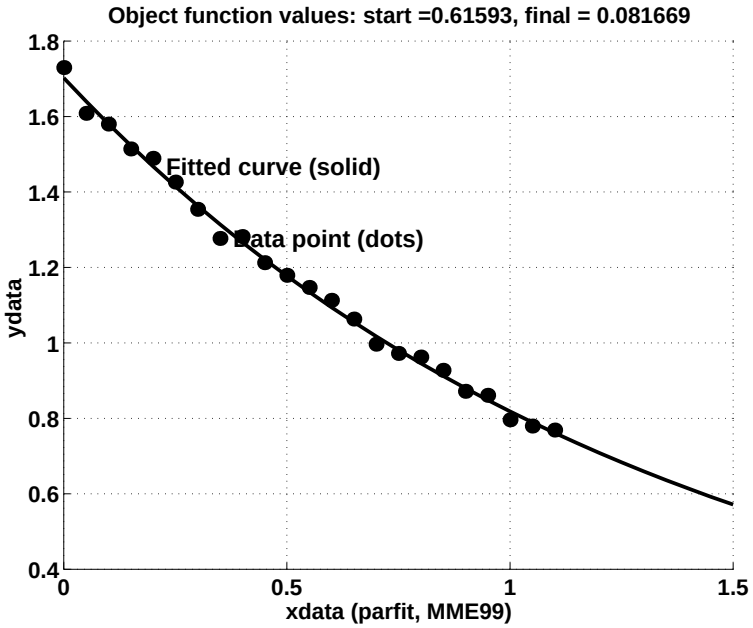
MATLAB-testausta varten tuotamme aluksi "synteettistä" dataa. Tällä hyvin monikäyttöisellä idealla tarkoitamme seuraavaa. Valitsemme ensin jonkin funktion, esim.

$$f(x,\lambda) = \lambda_1 e^{-x} + \lambda_2 e^{-\lambda_3 x}\,; \quad \lambda = (\lambda_1, \lambda_2, \lambda_3)\,,$$

ja generoimme datapisteet valitsemalla ensin λ_0:n ja asettamalla sitten $x_j = 0.05 \cdot j$, $y_j = f(x_j, \lambda_0)(1 + 0.1 \cdot \texttt{rand})$. Minimointiohjelmalle `fminsearch` annetaan argumenteiksi kohdefunktio ja alkuarvaus $\bar{\lambda}$ parametrivektorille.

```
% FILE  parfit.m  begins.
% USES: fmodel, fobj
% MME99
global xdata;
global ydata;
xdata= 0:0.05:1.1;
lam1=[0.2 1.5 0.7];
y=fmodel(lam1,xdata); % Generate syntetic data
                      % with parameters lam1
ydata= y.*(0.97+0.05*rand(1,length(xdata)));
                      % Add some "errors"
%fprintf('%8.4f',xdata); fprintf('\n');
%fprintf('%8.4f',ydata);
lam0=[1 1 1];         % Initial guess for lambda
y0=fobj(lam0);        % Initial value of object function
% OLD MATLAB:lam=fmins('fobj',lam0);
lam=fminsearch('fobj',lam0);
                      % lam is the fitted value for
                      % the parameter vector
x=0:0.05:1.5;
yfit=fmodel(lam, x);
yfinal=fobj(lam);     % Final value of the object function
clf;
axes('FontSize',[15],'FontWeight','bold'); hold on;
title(['Object function values: start =' num2str(y0) ', final = ' ...
       num2str(yfinal)])
plt=plot(x,yfit,xdata,ydata,'k.','MarkerSize',30);  grid;
txt1=' {\bf Fitted curve (solid)}';
text(x(5),yfit(5),txt1,'FontWeight','bold','FontSize',[16]);
txt2=' {\bf Data point (dots)}';
text(xdata(8),ydata(8),txt2,'FontWeight','bold','FontSize',[16]);
```

```
ylabel('ydata '); xlabel('xdata (parfit, MME99)')
set(plt,'LineWidth',2.5);
% FILE  parfit.m  ends.


function y=fmodel(lam,x)
y= lam(1)*exp(-x)+lam(2)*exp(-lam(3)*x);
% FILE  fmodel.m  ends.


function y=fobj(lambda)
global xdata;
global ydata;
y=norm(fmodel(lambda,xdata)-ydata);
% FILE  fobj.m  ends.
```



2.26. MATLABin uusia piirteitä. Uudempiin versioihin on otettu mukaan C-kielen rakenteista tietotyyppiä (`struct`) vastaava käsite, jolle joskus käytetään nimitystä tietue. Se voi sisältää osanaan useita erityyppisiä kenttiä. Kukin kenttä on jotakin MATLAB-perustietotyyppiä (luku, matriiseja tai merkkijono). Tietueen kenttien sisältöä voidaan tutkia ns. pistenotaation avulla kuten alempana ilmenee. Opastusta tietueiden käyttöön saa helpdeskin avulla (käynnistä Netscape ja anna MATLAB-komento `helpdesk`). Seuraava on poimittu sieltä.

Create structure array

Syntax

```
s = struct('field1',values1,'field2',values2,...)
```

Description

s = struct('field1',values1,'field2',values2,...)
creates a structure array with the specified
fields and values. The value arrays
values1, values2, etc. must be cell arrays of the same size or
scalar cells. Corresponding elements of the value arrays are
placed into corresponding structure array elements. The size of
the resulting structure is the same size as the value cell arrays
or 1-by-1 if none of the values is a cell.

Examples

The command

```
s = struct('type',{'big','little'},'color',{'red'},'x',{3 4})
```

produces a structure array s:

```
s =
1x2 struct array with fields:
    type
    color
    x
```

The value arrays have been distributed among the fields of s:

```
s(1)
ans =
      type: 'big'
    color: 'red'
        x: 3
s(2)
```

```
ans =
      type: 'little'
     color: 'red'
         x: 4
```

Annamme nyt vähän laajemman esimerkin tietueen käytöstä henkilörekisterin luomiseksi.

```
% FILE  adrlst.m  begins.
% Last modified 01-Sept-1999, Ver. 5.3 (R11)
% Creates an artificial database of persons
clear all;
first0='ensi';            flst=perms(first0);
last0='alku';             llst=perms(last0);
id0='1234';               idlst=perms(id0);
street0='yliopistonk 5';  slst=perms(street0(1:4));
city0='helsinki';         clst=perms(city0(1:4));
n=factorial(length(first0));
prs=struct('First',first0,'Last',last0,...
   'Id',id0,'Street',street0,...
   'City',city0,'Data',fix(100*rand(1,3)))
txt='First    Last     Id      Street     City ';
txt=[txt '    Data(1:3)   Aver/data'];
disp(txt);
line='------------------------------';
line=[line line];
disp(line);
s1=0.0;
for j=1:n
   prs(j)=struct('First',flst(j,:),'Last',llst(j,:),...
      'Id',idlst(j,:),'Street',slst(j,:),...
       'City',clst(j,:),'Data',fix(100*rand(1,3)));
   txt=[prs(j).First];
   txt=[txt blanks(5) prs(j).Last];
   txt=[txt blanks(5) prs(j).Id];
   txt=[txt blanks(5) prs(j).Street];
   txt=[txt blanks(5) prs(j).City];
   txt=[txt blanks(5) mat2str(prs(j).Data)];
   s1=s1+prs(j).Data(1);
   fprintf(['%s  %d\n' line '\n'],...
               txt,fix(sum(prs(j).Data)/3));
```

```
end
disp(['Sum of data(1) = ' num2str(s1)])
% FILE  adrlst.m  ends.
```

Toiminta:

```
>> adrlst

prs =

     First: 'ensi'
      Last: 'alku'
        Id: '1234'
    Street: 'yliopistonk 5'
      City: 'helsinki'
      Data: [40 15 6]

First    Last     Id      Street     City     Data(1:3)   Aver/data
-------------------------------------------------------------
isne     ukla     4321     oily     sleh     [12 2 86]  33
-------------------------------------------------------------
sine     kula     3421     ioly     lseh     [41 79 81]  67
-------------------------------------------------------------
inse     ulka     4231     oliy     selh     [39 82 33]  51
-------------------------------------------------------------
nise     luka     2431     loiy     eslh     [19 51 20]  30
-------------------------------------------------------------
 .        .        .        .        .           .       .
 .        .        .        .        .           .       .
 .        .        .        .        .           .       .
-------------------------------------------------------------
seni     kalu     3124     iylo     lhes     [75 90 42]  69
-------------------------------------------------------------
esni     aklu     1324     yilo     hles     [65 0 98]  54
-------------------------------------------------------------
nesi     laku     2134     lyio     ehls     [41 34 66]  47
-------------------------------------------------------------
ensi     alku     1234     ylio     hels     [44 83 52]  59
-------------------------------------------------------------
Sum of data(1) = 1141
```

2.27. Matriisien kirjoittaminen tiedostoon. Yksi data-analyysin perusasioista on tietojen lukeminen tiedostosta /kirjoittaminen tiedostoon. MATLABin käskyt `load` ja `save` tekevät juuri näitä asioita. Huomaa erityisesti niiden option `-ascii` käyttömahdollisuus.

Alla esitämme ohjelmat `putmat`, `getmat` ja `showmat` matriisien kirjoittamiseksi tiedostoon, lukemiseksi tiedostosta sekä kirjoittamiseksi kuvaruudulle.

```
function fname=putmat(a,filen,format)
% PUTMAT    write a real matrix to a file with
%           the given format
% Last modified 01-Sept-1999
% Example: putmat(a,'a1.dat',' %12.6e')
% Warning: There must be at least 2 characters before
%          ".dat"
[d1,d2]=size(a);
j=1;
fname=filen;
while ~(exist(fname) == 0)
 j=j+1;
 fname=strcat(fname(1:2),  num2str(j,'%02d'),'.dat');
end;
fid=fopen(fname,'w');
f=['\n' format];
for j=1:d2
  f=[f format];
end;
f=[f '\n'];
fprintf(fid,' %3d   %3d',d1,d2);
for j=1:d1
fprintf(fid,f,a(j,:));
end
fprintf('\n')
fclose(fid);
% FILE  putmat.m  ends.
```

`putmat` kirjoittaa matriisin tiedostoon formaatissa

```
   4     2
  0.82140  0.92180
  0.44470  0.73820
  0.61540  0.17620
  0.79190  0.40570
```

getmat puolestaan lukee yo. tavalla tiedostoon kirjoitetun matriisin.

```
function a=getmat(fname)
% GETMAT reads a real matrix from a file
% Last modified 24-Sept-1999
% Example: a=getmat('a1.dat')
fid=fopen(fname,'r');
d=fscanf(fid,'%d',2);  d1=d(1); d2=d(2);
a=zeros(d1,d2);
a=fscanf(fid,'%lf');
fclose(fid);
b=reshape(a,d2,d1);
a=b';
% FILE getmat.m ends.
```

showmat kirjoittaa matriisin kuvaruudulle halutun formaatin mukaisesti.

```
function a=showmat(a,format)
% SHOWMAT displays a real matrix
% Last modified 01-Sept-1999
% Example: showmat(rand(5,2),' %7.4f')
[d1,d2]=size(a);
f=['\n' format];
for j=1:d2
  f=[f format];
end;
f=[f '\n'];
for j=1:d1
fprintf(f,a(j,:));
end
fprintf('\n');
% FILE  showmat.m  ends.
```

```
>> a=rand(4,4)

a =

    0.0455    0.0354    0.8031    0.2381
    0.0857    0.9796    0.5167    0.9431
    0.5131    0.4974    0.5027    0.0886
    0.7154    0.7007    0.7547    0.5430
```

```
>> showmat(a,' %6.4f');

 0.0455 0.0354 0.8031 0.2381
 0.0857 0.9796 0.5167 0.9431
 0.5131 0.4974 0.5027 0.0886
 0.7154 0.7007 0.7547 0.5430
```

Kuten esimerkistä ilmenee, oletusarvoisesti MATLAB tulostaa matriisit melko väljällä formaatilla. `showmat` on yksi mahdollisuus kompaktimpaan tulostukseen. Ks. myös `format`-käskyn optiota.

```
% FILE  mattst.m  begins.
% Last modified 01-Sept-1999
% USES: showmat, getmat, putmat
% Test matrix utilities showmat,getmat,putmat
m=input('Enter pos. integers m in [1,7]: ');
n=input('Enter pos. integers m in [1,7]: ');
a1=fix(10000*rand(m,n))/10000;
                % Digits 5,6,... will be zero
showmat(a1,' %8.5f');
fname=putmat(a1,'a1.dat', ' %8.5f')
a2=getmat(fname);
%showmat(a1-a2, '  %g');
a1-a2
if (norm(a1-a2)==0)
   disp('Test error = 0 ')
end;
%  FILE  mattst.m  ends.
```

Ohjelma `mattst.m` havainnollistaa yllä olevien ohjelmien yhteiskäyttöä.

```
>> mattst
Enter pos. integers m in [1,7]: 4
Enter pos. integers m in [1,7]: 3

  0.90340  0.37900  0.53510
  0.86130  0.33730  0.70190
  0.63140  0.31450  0.87500
  0.26230  0.94710  0.98620
```

```
fname =
a1.dat
ans =
     0     0     0
     0     0     0
     0     0     0
     0     0     0
Test error = 0

>> type a1.dat

   4     3
  0.90340  0.37900  0.53510
  0.86130  0.33730  0.70190
  0.63140  0.31450  0.87500
  0.26230  0.94710  0.98620
```

2.28. Tavallisia MATLAB-virheitä. Ohjelmoinnin oppaana on syytä käyttää helpiä ja tutustua MATLABin omien demojen koodeihin. Esimerkiksi komento `help fzero` antaa opastusta funktion `fzero` käytöstä, ja komennolla `type polyfit` nähdään, miten `polyfit` on toteutettu. Sama soveltuu useimpiin MATLAB-funktioihin (poikkeuksena ns. built-in funktiot, esim. `svd`).

- nimien sekaantuminen: `x` on vektori, mutta toisaalta on tehty m-tiedosto x.m, jossa on ristiriitainen määrittely. Voi johtaa käsittämättömiin virhetilanteisiin. Älä myöskään käytä tiedostonnimiä i.m, j.m, k.m tai *.m, missä * on MATLABin funktio

- nimien sekaantuminen voi aiheutua myös siitä, että työtilassa olevat muuttujat häiritsevät. Esim. on määritelty

  ```
  x= 1:100;
  x(1:20)=rand(1,20);
  y = sin(x(1:20));
  ```

  Tällöin komento `plot(x,y)` ei toimi, sen sijaan voidaan esim. piirtää komennolla `plot(x(1:20),y(1:20))`

- operaatioiden `x^2` ja `x.^2` ero, kun `x=1:10`. Jälkimmäinen on alkioittainen potenssiin korotus
- `a*b`:n ja `a.*b`:n välinen ero, kun `a` ja `b` matriiseja
- vaakavektori `x=[1:10]` $\neq$ pystyvektori `x'`
- dimensiovirhe matriisioperaatioissa
- dimensiovirhe MATLAB-funktioiden argumenteissa (esim. `plot`, `polyfit`, `spline` jne.)
- hyvät ohjelmointitottumukset (kommentointi, dokumentointi, tiedostojen systemaattinen nimeäminen jne.) estävät virheitä
- versiossa 3.5 on määriteltävä `i=sqrt(-1)` ennen käyttöä
- indeksin `i` ja $\sqrt{-1}$:n sotkeentuminen
- matriisin A konjugaatti-transpoosi on `A'` ja A^T on `conj(A)'`

3 Interpolaatio

Tässä luvussa perehdymme MATLABin avulla polynomien käsittelyyn ja interpolointiin.

3.1. Polynomit. MATLAB tarjoaa polynomien käsittelyyn useita valmiita funktioita. Tavallisimmat näistä ovat `roots`, `poly`, `polyval`, `polyfit`.

Polynomeja esitetään muodossa

$$p_n(z) = \sum_{k=1}^{n+1} a(k) z^{n-k+1} \tag{3.2}$$

ja niitä tutkitaan MATLABissa lähinnä juurien tai kertoimien $a(k)$ avulla.

Esimerkki. (1) Wilkinsonin polynomi: `a=poly(1:20)` antaa vektorin `a(1)` ... `a(21)`, jota vastaavan polynomin juuret ovat luvut $1, 2, \ldots, 20$.

(2) `roots(a)` antaa polynomin (1) juuret.

(3) `polyval(coef,z)` antaa kerroinvektoria `coef` vastaavan polynomin arvon `z`:ssa.

3.3. Kertoimien vaikutus juuriin. Wilkinson totesi, että pieni muutos polynomin

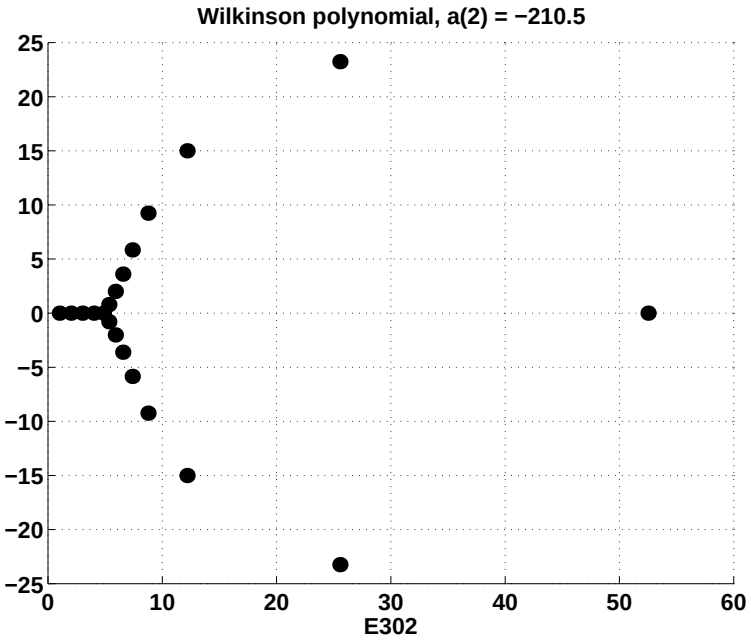
$$w_{20}(z) = \prod_{k=1}^{20}(z-k) = \sum_{p=1}^{21} a(p) z^{21-p}$$

kertoimiin vaikuttaa paljon juuriin. Tässä $a(1) = 1$, $a(2) = -210$, $a(21) = 2.4329 \cdot 10^{18}$.

Varioidaan kerrointa $a(2)$ ja piirretään varioitujen polynomien juuret kompleksitasoon.

```
% FILE  e302.m  begins.
% Study Wilkinson polynomial
for j=1:10
    a=poly(1:20);
    a(2)= a(2)-0.1*(j-5);
    z =roots(a);  x1= real(z);  y1= imag(z);
    clf;
    axes('FontSize',[15],'FontWeight','bold');  hold on;
    txt=['Wilkinson polynomial, a(2) = ' num2str(a(2)) ];
    plot(x1,y1,'k.','MarkerSize',30);
```

```
    title(txt,'FontSize',[15]);
    xlabel('E302','FontSize',[15]);
    grid;
    hold off;
    pause;
end;
% FILE  e302.m  ends.
```



Jos esim. $a(2) = -210.1$, niin vastaavalla polynomiyhtälöllä on juuria z, joille $\mathrm{Re}\, z > 20$, $\mathrm{Im}\, z > 15$; ks. kuva. Kuten kuva osoittaa, pieni kertoimen $a(2)$ variointi muuttaa paljon juurien sijaintia kompleksitasossa. Näin ollen juuret riippuvat kertoimista epästabiililla tavalla.

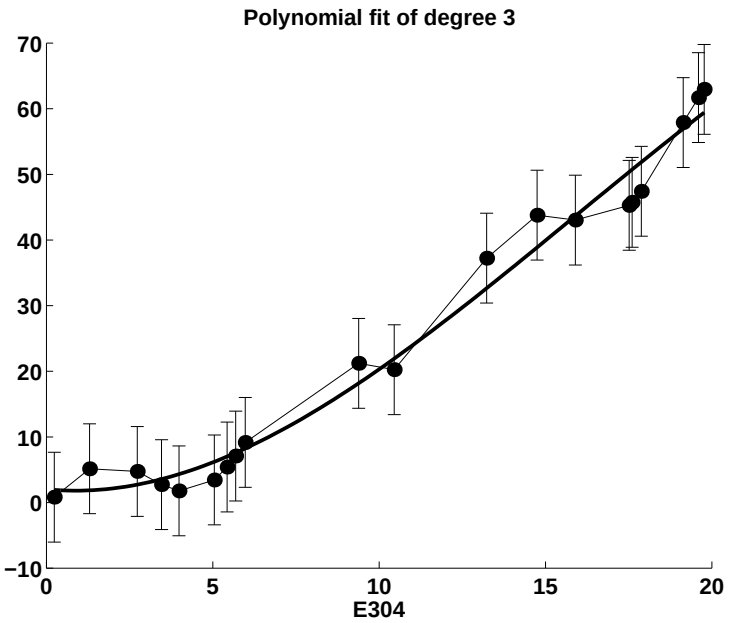
3.4. Hornerin kaava. Polynomin arvojen laskeminen voidaan tehdä suoraan laskemalla termit yhteen. Tämä suoraviivainen menettely ei ole tehokas, jos asteluku on suuri. *Hornerin kaava* antaa tehokkaamman tavan:

```
p=a(1);
for j=1:20
    p=p*z+a(j+1);
end;
```

MATLABin `p= polyval(a,z)` perustuu samaan ideaan, kuten voidaan todeta ottamalla MATLAB-tiedosto polyval.m editoriin tai antamalla MATLABissa komento `type polyval` .

3.5. Polynomisovitus. Datapisteistöön $(x(i), y(i))$ $(i = 1, \ldots, 10)$ halutaan sovittaa pns-menetelmällä astetta n oleva polynomi. Yleensä on parasta valita n pieneksi, esim. $n \leq 4$. Polynomisovitus tapahtuu seuraavasti.

```
% FILE  e304.m  begins.
% Study polynomial fit
n =3;  %degree of polynomial
x = 20*sort(abs(rand(1,20))); y = x.*log(x) + 5*sin(x);
a = polyfit(x,y,n);
xmi = min(x); xma= max(x); dx =0.01*(xma-xmi);
xi =xmi:dx:xma; yi = polyval(a,xi);
clf;
axes('FontSize',[15],'FontWeight','bold');  hold on;
txt =['Polynomial fit of degree '  num2str(n)];
plt1=plot(x,y,'k.',xi,yi,'MarkerSize',30);
title(txt,'FontSize',[15]);
xlabel('E304','FontSize',[15]);
e=0.3*std(y)*ones(size(x));
errorbar(x,y,e);
set(plt1,'LineWidth', 2.5);       % Line width is changed
hold off;
pause;
% FILE  e304.m  ends.
```



Monet MATLABin perusfunktiot ovat käyttäjän nähtävissä m-tiedostoissa. Edellä käytetyt funktiot `polyfit` ja `polyval` ovat tiedostoissa polyfit.m ja polyval.m. Niiden osana on mm. lineaarisen yhtälöryhmän ratkaisu. Palautetaan vielä lyhyesti mieleen em. pns-sovituksen matemaattiset lähtökohdat. Olkoot annettu data (x_j, y_j) $(j = 1, \ldots, m)$, johon sovitetaan polynomi $p_n(x) = \sum_{k=1}^{n+1} a_k x^{n+1-k}$, missä $n + 1 < m$. Muodostetaan summa

$$s(a_1, \ldots, a_{n+1}) = \sum_{j=1}^{m} (p_n(x_j) - y_j)^2 \,,$$

joka halutaan minimoida. Tapauksessa $n + 1 = m$ saattaa löytyä vektori $a = (a_1, \ldots, a_{n+1})$, jolle $s(a) = 0$, mutta tarkasteltavassa tapauksessa $n + 1 < m$ näin ei yleensä ole asian laita. Välttämätön ehto vektorin a minimaalisuudelle saadaan *normaaliyhtälöistä* $\frac{\partial s}{\partial a_k} = 0$ $(k = 1, \ldots, n + 1)$:

$$2 \sum_{j=1}^{m} \frac{\partial p_n(x_j)}{\partial a_k} (p_n(x_j) - y_j) = 0$$

$$\Leftrightarrow \sum_{j=1}^{m} x_j^{n+1-k} \left(\sum_{k=1}^{n+1} a_k x_j^{n+1-k} - y_j \right) = 0 \,.$$

Edellä oleva MATLAB-koodi ratkaisi tämän yhtälöryhmän pns-mielessä a:n suhteen funktion `polyfit` avulla.

3.6. Interpolaatiotehtävä. Funktion arvot $y(i)$ tunnetaan pisteissä $x(i)$ $(i = 1, \ldots, n)$, missä $x(1) < x(2) < \cdots < x(n)$. *Interpolaatiotehtäväksi* kutsutaan tehtävää määrittää approksimaatio funktion arvolle pisteessä z, $x(1) < z < x(n)$. Jos $z < x(1)$ tai $z > x(n)$, on kysymyksessä *ekstrapolaatiotehtävä*. Vrt. kohta 1.7.

3.7. Esimerkki. (1) Interpolaatiotehtävään joudutaan mm. logaritmi- tai trigonometristen funktioiden taulukoita käytettäessä. Erään funktion arvot tunnetaan pisteissä $x_1 = 7.9$ ja $x_2 = 8.0$.

i	x_i	$f(x_i)$
1	7.9	0.89717 4302
2	8.0	0.89823 7113

Funktion arvo pisteessä $x = 7.9527$ saadaan lineaarisella interpolaatiolla kaavalla

$$f(x) = (1 - p) f(x_1) + p f(x_2) \,; \quad p = (x - x_1)/(x_2 - x_1) \,.$$

Nyt $p = 0.527$ ja

$$f(x) = (1 - 0.527) \cdot 0.89717\ 4302 + 0.527 \cdot 0.89823\ 7113 = 0.89773\ 4403$$

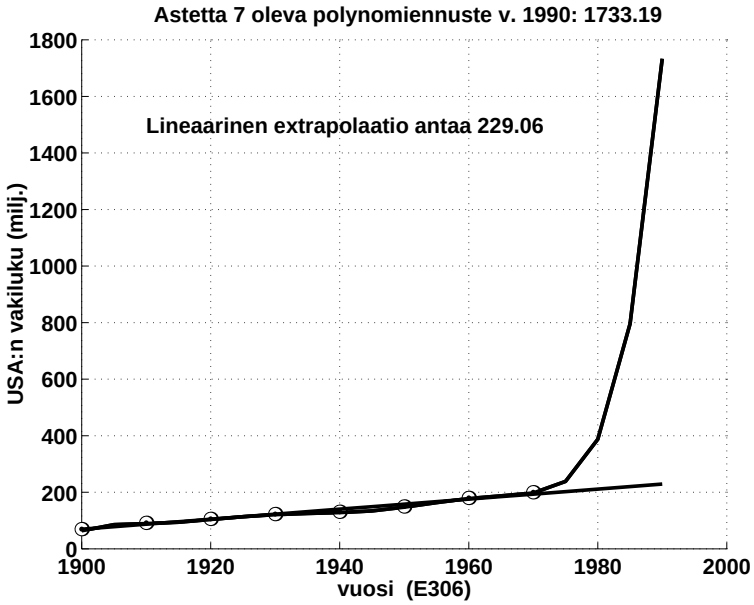
Tämä ja muita interpolaatiomenetelmiä on esitetty teoksessa [AS, s. X].

(2) USA:n väkiluvun ekstrapolaatio. USA:n väkiluvusta on seuraavat tiedot

vuosi	1900	1910	1920	1930	1940	1950	1960	1970
milj.	70	92	106	123	131	150	180	200

Tehtävänä on polynomiekstrapolaation avulla arvioida väkiluku v. 1990.

```
% FILE  e306.m  begins.
% Polynomial extrapolation bases on census data
yr= 1900:10:1970;
pop =[ 70, 92 , 106, 123, 131, 150, 180, 200];
coef= polyfit(yr,pop,7);
yri = 1900:5: 1990;
popi= polyval(coef,yri);
clf;
axes('FontSize',[15],'FontWeight','bold');  hold on;
plt1=plot(yr,pop,'k.', yri,popi,'MarkerSize',[30]);
c = polyfit(yr, pop,1);
poplin =polyval(c,yri);
pop90=polyval(coef,1990);
est = polyval(c,1990);
txt2= ['Lineaarinen extrapolaatio antaa ' num2str(est,'%6.2f')];
txt =[' Astetta 7 oleva polynomiennuste v. 1990: ' ...
      num2str(pop90,'%6.2f')];
plt2=plot(yr, pop,'wo',yri, poplin, yri, popi);
title(txt,'FontSize',[15]);
text(1910,1500,txt2,'FontSize',[15],'FontWeight','bold');
xlabel('vuosi  (E306)','FontSize',[15]);
ylabel('USA:n vakiluku (milj.)','FontSize',[15]);
grid;
set(plt1,'LineWidth', 2.5);       % Line width is changed
set(plt2,'LineWidth', 2.5);       % Line width is changed
hold off;
% FILE  e306.m  ends.
```



Saadaan järjetön vastaus (1733 miljoonaa). Opetus: polynomiekstrapolaatiota kannattaa käyttää varoen. (Huom. MATLABin demossa tämä on esitetty varoittavaksi esimerkiksi.)

(3) Interpolaatiota käytetään myös differentiaaliyhtälöiden numeerisessa integroinnissa.

(4) Ekstrapolaatiota käytetään numeerisessa integroinnissa ns. Rombergin menetelmän yhteydessä.

3.8. Interpolaatiomenetelmiä. Interpolaatiomenetelmillä on viimeisten 30 vuoden aikana ollut paljon sovelluksia mm. tietokoneavusteisessa muotoilussa. Tunnettuja interpolaatiomenetelmiä ovat mm.

(a) polynomi-interpolaatio, Lagrangen kaava;
(b) paloittainen polynomi-interpolaatio, erityisesti ns. splini-interpolointi;
(c) Hermiten interpolaatio;
(d) rationaali-interpolaatio (ns. Padé-approksimointi);
(e) Bézier-käyrät.

3.9. Lagrangen interpolaatiokaava. Interpolaatiotehtävälle 3.6 yksi ratkaisu on astetta $n-1$ olevan polynomin sovittaminen pisteistöön. Ratkaisu voidaan periaatteessa löytää esim. määräämättömien kertoimien menetelmän avulla. J. L. Lagrange (1736–1813) on antanut seuraavan eksplisiittisen kaavan

tälle enintään astetta $n-1$ olevalle polynomille

$$p_n(x) = \sum_{j=1}^{n} l_j(x) y(j)\,; \qquad l_j(x) = \prod_{\substack{i=1 \\ i \neq j}}^{n} \frac{x - x(i)}{x(j) - x(i)}\ .$$

USA:n väkilukuesimerkin 3.7 (2) yhteydessä näimme jo, miten n pisteen dataan $(x(j), y(j))\ \ (j = 1, \ldots, n)$ sovitetaan astetta $n-1$ oleva polynomi MATLAB-kielessä ja miten polynomin arvot lasketaan pisteissä $xi(k)$ $(k = 1, \ldots, m)$:

```
coef = polyfit(x,y,n-1);
yi = polyval(coef,xi);
```

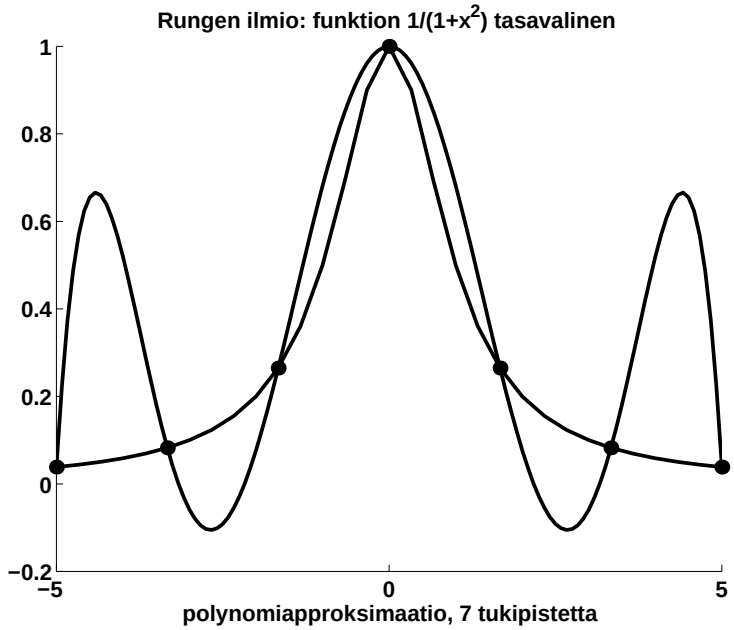
3.10. Splini-interpolaatio. Edellä on jo viitattu splini-interpolaatiomenetelmiin, joita on useita. Yhteistä näille kaikille on "paloittain polynominen luonne", ts. annettuun dataan sovitetaan kiinteää astelukua (esim. 3) olevia polynomeja siten, että polynomin kertoimet ovat samoja esim. väleillä $[x(2k-1), x(2k+1)]\ \ (k = 1, \ldots, p)$ ja derivaatta on jatkuva liitospisteissä $x(2k-1)$. MATLABin funktio `spline` tekee erään splinisovituksen käskyllä

```
yi = spline(x,y,xi);
```

Yleisesti voidaan todeta, että splini-interpolaatio on joustavampaa ja luotettavampaa kuin Lagrangen kaavaan perustuva interpolaatio. Tämä ilmenee myös alla olevasta varoittavasta Rungen esimerkistä.

3.11. Rungen (1856–1927) ilmiö. Tarkastellaan funktion $f(x) = 1/(1+x^2)$ tasavälistä polynomiapproksimointia välillä $[-5, 5]$. Kun tukipisteiden $x(i) = -5 + (i-1) * 10/(n-1)\ \ (i = 1, \ldots, n)$ lukumäärä n kasvaa, alkaa interpoloiva polynomi heilahdella rajusti lähellä välin päätepisteitä. Näin ollen interpoloiva polynomi tuottaa kelvottomia tuloksia.

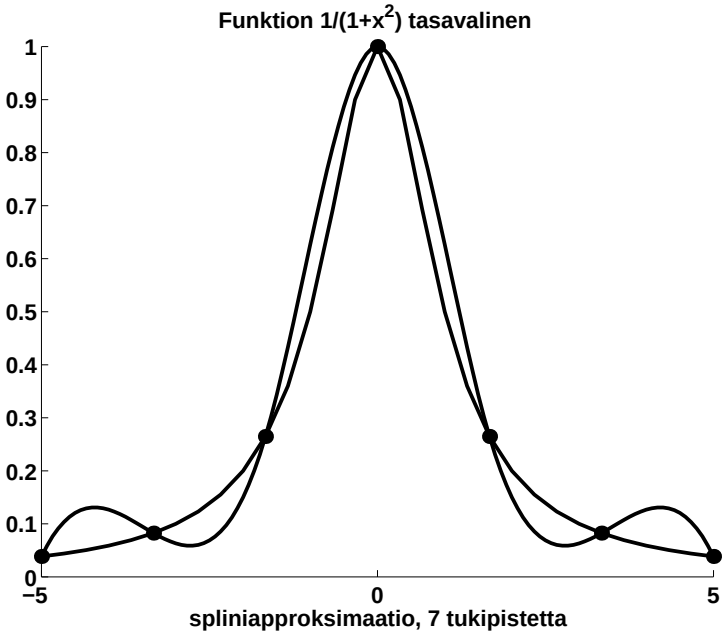
Opetus: Lagrangen interpolaatiota ei pidä käyttää suurilla n:n arvoilla ($n \geq 6$). Jos datapisteitä on paljon, voidaan esim. käyttää paloittain kvadraattista tai kuutiollista interpolaatiota.



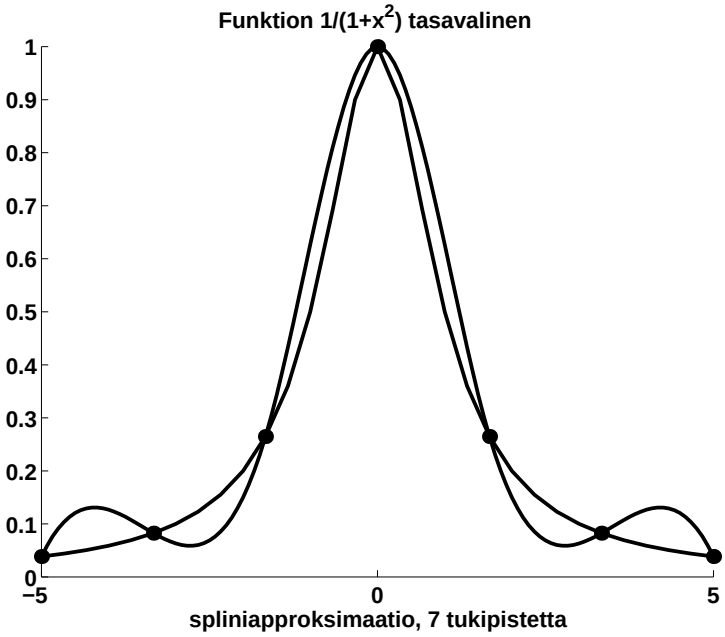
MATLABissa Rungen ilmiötä voidaan tutkia seuraavasti.

```
% FILE  e310.m  begins.
% Rungen ilmio: Funktion y = 1/(1+x^2) tasavalinen
% polynomiapproksimaatio kun n kasvaa
x1= -5; x2= 5;
u= x1 : (x2-x1)/30: x2;
[m,n] = size(u); one = ones(m,n);
v= one./( one +  u.^2);
for n=3:14,
   dx=(x2-x1)/n;
   x= x1: dx: x2;
   one = ones(size(x));
   y =  one./(one + x.^2);
   coeff = polyfit(x,y,n);
   xi= x1: dx/20 : x2;
   yi= polyval(coeff,xi);
   yis = spline(x,y,xi);
   clf;
   axes('FontSize',[15],'FontWeight','bold');  hold on;
   plt1=plot(x,y,'k.',u,v,xi,yi,'k-','MarkerSize',[30]);
   tit =['Rungen ilmio: funktion 1/(1+x^2) '];
   tit =[ tit 'tasavalinen '];
   title(tit),
```

```
   set(plt1,'LineWidth', 2.5);       % Line width is changed
   xlabel(['polynomiapproksimaatio, ' num2str(n+1) ...
           ' tukipistetta'], 'FontSize',[15]);
   pause;
   clf;
   axes('FontSize',[15],'FontWeight','bold');  hold on;
   plt2=plot(x,y,'k.',u,v,xi,yis,'k-','MarkerSize',[30]);
   tit =['Funktion 1/(1+x^2) '];
   tit =[ tit 'tasavalinen '];
   title(tit,'FontSize',[15]);
   xlabel(['spliniapproksimaatio, ' num2str(n+1) ...
           ' tukipistetta'], 'FontSize',[15]);
   set(plt2,'LineWidth', 2.5);       % Line width is changed
   hold off;

pause;
end;
% FILE  e310.m  ends.
```

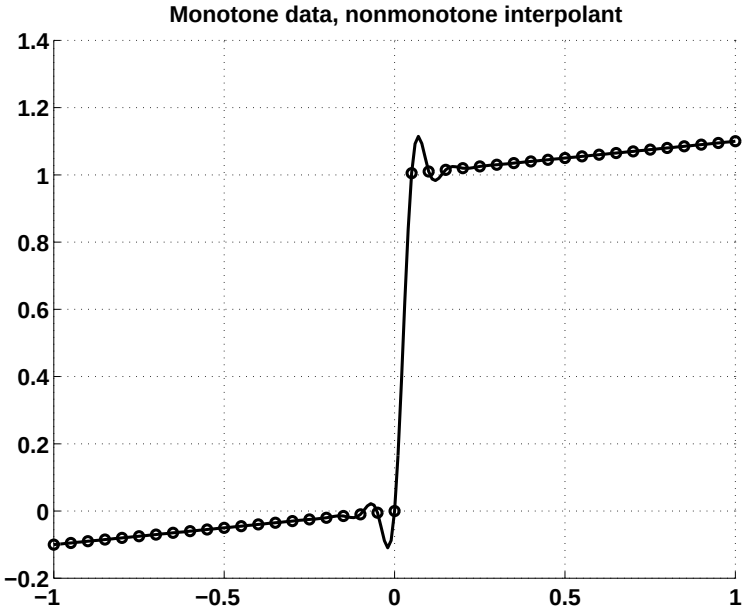


Jos käytetään polynomiapproksimaation sijasta spliniapproksimaatiota, saadaan dataan paremmin sopiva käyrä.



3.12. Monotoninen data. Jos interpolaatio on tehtävä monotoniseen dataan (x_i, y_i) $(i = 1, \ldots, n)$ ts. $x_i < x_{i+1}$ ja $y_i < y_{i+1}$, on luonnollinen kysymys, onko interpoloiva funktio monotoninen esim. välillä $[x_1, x_n]$. Näin ei yleensä tarvitse olla, vaan esim. kuvan ilmiö voi tapahtua splini-interpoloinnissa.

```
% FILE  e311.m  begins.
clf;
axes('FontSize',[15],'FontWeight','bold');  hold on;
x=-1:0.05:1;
y=0.1*x.*(x<=0)+ 0.1*((x+10).*(x>0));
xi=-1:0.01:1;
yi=spline(x,y,xi);
pic=plot(x,y,'ko',xi,yi);
set(pic,'LineWidth', 2);
title('Monotone data, nonmonotone interpolant','FontSize',[15])
grid
% FILE  e311.m  ends.
```



On kuitenkin kehitetty interpolaatiomenetelmiä, jotka tuottavat monotoniselle datalle monotonisen interpoloivan funktion. Ks. [KMN, s. 112].

3.13. MATLABin interpolaatiofunktioita. Edellä on jo esitelty funktioiden `polyval`, `polyfit` ja `spline` toimintaa. Muita interpolaatioon liittyviä MATLAB-funktioita ovat mm. `interp1`, `ppval`, `mkpp`, `unmkpp`, joista saa kuvauksen helpin avulla (ts. `help interp1` jne.).

3.14. Esimerkki. Lämpötila kolmen ajankohdan välillä on kvadraattinen funktio. Lämpötila illalla klo 21 oli 3°C, aamulla klo 6 2°C ja klo 10 8°C. Oliko yöllä pakkasta?

Ratkaisu. Skaalataan $21, 6, 10 \to -3, 6, 10$. Olkoon lämpötila $L(t) = at^2 + bt + c$. Mittaukset antavat

$$\begin{cases} L(-3) = 9a - 3b + c = 3 \\ L(6) = 36a + 6b + c = 2 \\ L(10) = 100a + 10b + c = 8 \end{cases} \quad \Rightarrow \quad \begin{cases} a = 0.1239 \\ b = -0.4829 \\ c = 0.4359 \end{cases} .$$

Minimi hetkellä, jona $L'(t) = 0 \Leftrightarrow t = -\dfrac{b}{2a}$:

$$L(-\frac{b}{2a}) = a(-\frac{b}{2a})^2 - \frac{b^2}{2a} + c = -0.0345\,.$$

Vastaus: Minimilämpötila oli -0.0345°C, siis oli pakkasta.

3.15. Numeerinen derivointi. Funktion $f\colon \mathbb{R} \to \mathbb{R}$ numeerinen derivointi voidaan perustaa kaavaan

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}, \quad \text{jos } h \approx 0. \tag{3.16}$$

Tässä parametri h on valittava sopivasti. Kirjallisuudessa esiintyy mm. peukalosääntö $h = (\text{meps})^d$, missä $d = 0.5$ ja meps on kone-epsilon. Liian pieni $|h|$:n arvo (3.16):ssä voi johtaa tarkkuuden tuhoavaan merkitsevien numeroiden kumoutumiseen. Toisaalta kaavan (3.16) sokea käyttö voi johtaa virheisiin myös siinä tapauksessa, että f "heiluu paljon" välillä $(x-h, x+h)$.

Kaava (3.16) perustuu funktion f approksimointiin lineaarisella funktiolla. Yhtä hyvin f:ää voitaisiin approksimoida polynomilla tukeutuen pisteisiin $x_i = x + ih \quad (i = -n, -n+1, \ldots, n-1, n)$, esim. $n = 2$, käyttäen Lagrangen interpolaatiokaavaa. Jos nyt p_{2n} on tällainen approksimaatio, jolloin $p_{2n}(x_i) = f(x_i)$, kun $i = -n, \ldots, n$, niin (3.16):n asemasta voidaan käyttää approksimaatiota

$$f'(x_0) \approx p'_{2n}(x_0). \tag{3.17}$$

Kaavojen (3.16) ja (3.17) tyyppisiä approksimaatioita ja niiden virhearvioita on tutkittu paljon numeriikan teoksissa, ks. esim. [Na, Ch. V], [AS, Ch. 25]. Tapauksessa $n = 2$ pätee [AS, 25.3.6]:n nojalla

$$f'(x_p) \approx \frac{1}{h}(a(p)f(x_{-2}) - b(p)f(x_{-1}) + c(p)f(x_0) \tag{3.18}$$

$$-d(p)f(x_1) + e(p)f(x_2)); \quad x_p = x_0 + ph\,,$$

missä

$$12a(p) = 2p^3 - 3p^2 - p + 1\,, \quad 6b(p) = 4p^3 - 3p^2 - 8p + 4\,,$$
$$2c(p) = 2p^3 - 5p\,, \quad 6d(p) = 4p^3 + 3p^2 - 8p - 4\,,$$
$$12e(p) = 2p^3 + 3p^2 - p - 1\,.$$

Kaavassa (3.18) muuttuja p saa arvot $-2, -1, 0, 1, 2$. Erityisesti kun $p = 0$, saadaan (3.18):sta

$$f'(x_0) \approx \frac{1}{h}\Big(\frac{1}{12}f(x_{-2}) - \frac{2}{3}f(x_{-1}) + \frac{2}{3}f(x_1) - \frac{1}{12}f(x_2)\Big)\,. \tag{3.19}$$

Kaavojen (3.16)–(3.19) lisäksi muitakin ideoita numeerisen derivoinnin tekemiseksi on esitetty. Mm. [NR, s. 188] on *Ridderin algoritmi*, jossa käytetään seuraavan kaltaista ideaa: Lasketaan arvoilla $h_i = 2^{-i} \quad (i = 1, \ldots, n-1)$ approksimaatio kaavasta (3.16). Saatujen derivaatan approksimaatioiden, olkoot ne df_i,

kautta sovitetaan n-asteinen polynomi dataan $(2^{-i}, df_i)$. Ekstrapolaatio tapaukseen $h = 0$ antaa approksimaation derivaatalle.

Yo. kaavan (3.17) MATLAB-toteutus lienee selvä; joka tapauksessa se on erikoistapaus myöhemmin esitettävästä vektoriarvoisen funktion numeerisen Jacobin matriisin laskemisesta. Alla yksi MATLAB-implementointi numeeriselle derivoinnille.

3.20. Numeerinen derivointi MATLABissa. Tarkastelemme yhden muuttujan funktion tapausta. Funktio on esim. tiedostossa f315.m:

```
function y = f315(x)
y = sin(x) + cos(10*x);
```

Esitämme vertailun vuoksi numeerisen derivoinnin toteutuksen sekä kaavan (3.16) että (3.19) mukaisesti. Kaavan (3.16) mukainen numeerinen derivointi sijoitetaan tiedostoon numdif.m:

```
function df = numdif(f,x,h)      % x vector
[m,n] = size(x);  one = ones(m,n);
df = (feval(f,x+h*one)-feval(f,x-h*one))/(2*h);
```

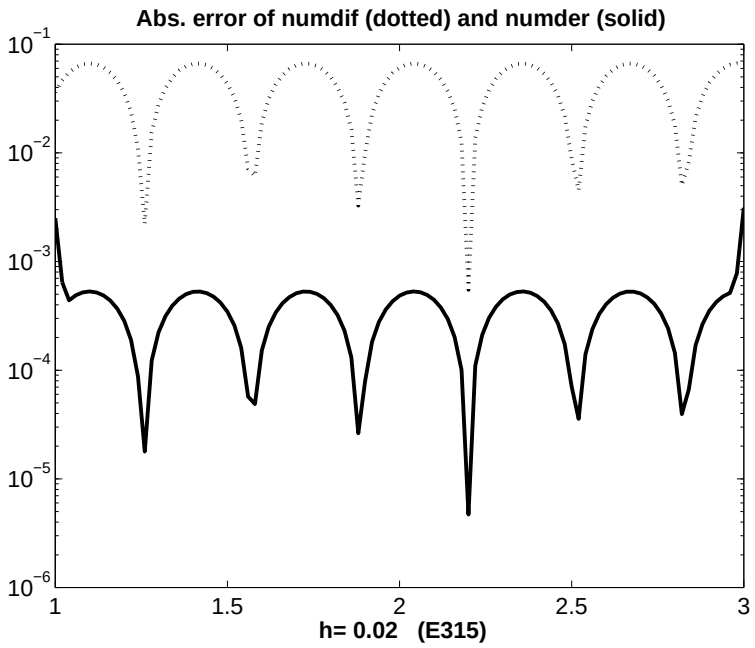
Kaavan (3.19) mukainen numeerinen derivointi sijoitetaan tiedostoon numder.m:

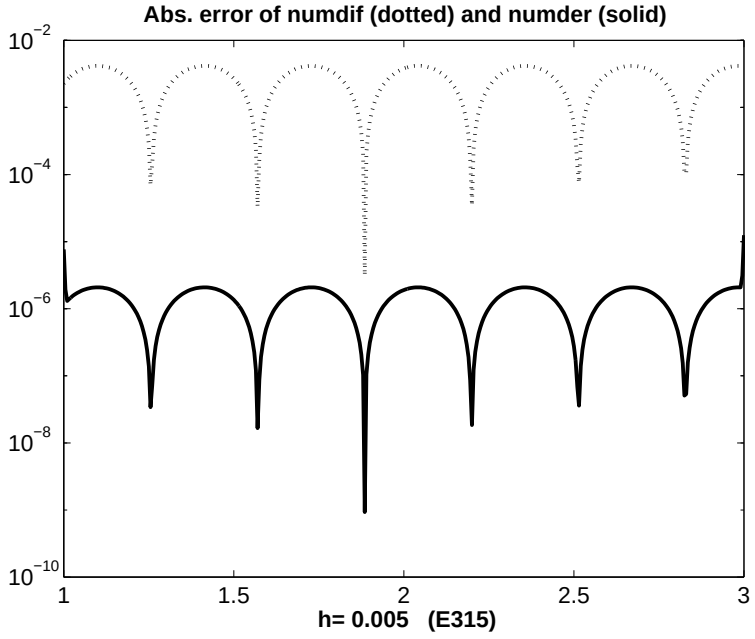
```
function dy = numder(y,h)
%NUMDER gives numerical approximation of derivative
%       of equally spaced data y = y(x+j*h), j=1,...,m, m >=5.
%       The 5-point rule numerical differentiation coefficients,
%       from Abramowitz-Stegun 25.3.6 are used.  To compute
%       dy0=f'(x0) at a single point x0 set e.g. h =0.001
%       x=x0-2*h:h:x0+2*h; y = f(x); dy=numder(y,h); dy0=dy(3);
coe=[];
for p =-2:2
   a= (2*p^3-3*p^2-p+1)/12;     b= (4*p^3-3*p^2-8*p+4)/6;
   c= (2*p^3-5*p)/2;
   d= (4*p^3+3*p^2-8*p-4)/6;    e= (2*p^3+3*p^2-p-1)/12;
   coe=[coe; [a -b c -d e]];
end;
[d1,d2]=size(y);
if ((min(d1,d2)>1) | (max(d1,d2) <5)) % y isn't 1*n or n*1, n>4
  error('Argument error in numder');
end;
```

```
dy =y;
dy(1)=(1/h)*sum(coe(1,:).*y(1:5));
dy(2)=(1/h)*sum(coe(2,:).*y(1:5));
for p=3:d2-2
   dy(p)=(1/h)*sum(coe(3,:).*y(p-2:p+2));
end;
dy(d2-1)=(1/h)*sum(coe(4,:).*y(d2-4:d2));
dy(d2)=(1/h)*sum(coe(5,:).*y(d2-4:d2));
%end;
% FILE  numder.m  ends.
```

Testaustiedostoa e315.m käyttävä testaus on suoritettu kahdella eri h:n arvolla, $h = 0.02$ ja $h = 0.005$. Alla olevista testauksen tuottamista kuvista nähdään, että molemmissa tapauksissa `numder` on vähän tarkempi.

```
% FILE  e315.m  begins.
% USES numder.m, numdif.m
clf;
h = input('Enter h (e.g. = 0.02) : ');
x=1:h:3;
z1 = numdif('f315',x,h)-cos(x)+10*sin(10*x);
axes('FontSize',[15],'FontWeight','bold');
plt1=semilogy(x,abs(z1),'k:');
set(plt1,'LineWidth', 2.5);       % Line width is changed
txt=(['Abs. error of numdif']);
title(txt,'FontSize',[15]);
pause
clf;
dy = numder(f315(x),h);
z2= dy-cos(x)+10*sin(10*x);
axes('FontSize',[15],'FontWeight','bold');  %hold on;
txt=(['Abs. error of numdif (dotted) and numder (solid)']);

plt2=semilogy(x,abs(z2),'k-',x,abs(z1),'k:'); hold on;
title(txt,'FontSize',[15],'FontWeight','bold');
xlabel([' h= ' num2str(h) '   (E315)'],'FontSize',[15],...
        'FontWeight','bold');
set(plt2,'LineWidth', 2.5);       % Line width is changed
hold off;
% FILE  e315.m  ends.
```

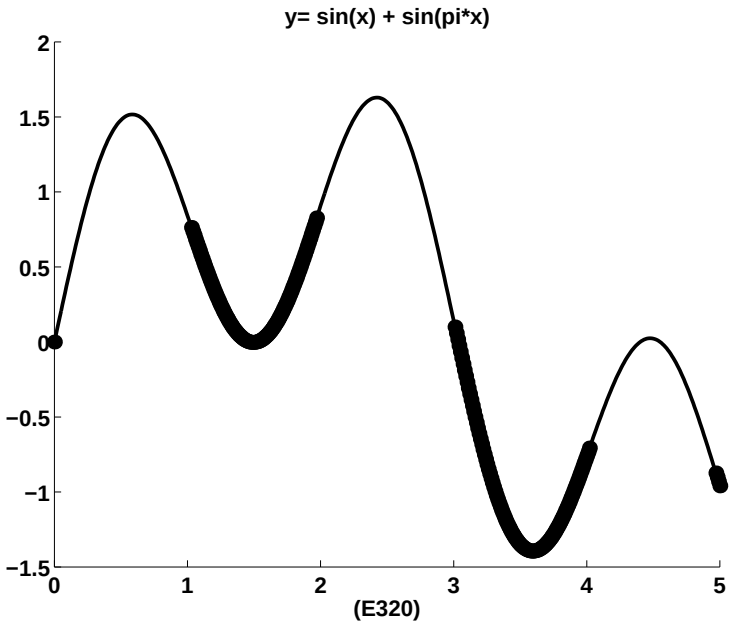




Sovellamme nyt `numder`-algoritmia merkitsemällä käyrään $y = \sin(x)+\sin(\pi x)$ ne pisteet, joissa $y'' > 0$.

```
% FILE  e320.m  begins.
x= 0.0: 0.01: 5;
```

```
y= sin(x) + sin(pi*x);
clf;
axes('FontSize',[15],'FontWeight','bold');  hold on;
plt1=plot(x,y);
pause
txt=['y= sin(x) + sin(pi*x)'];
title(txt,'FontSize',[15]);
set(plt1,'LineWidth', 2.5);
dy = numder(y, 0.01);
ddy = numder(dy, 0.01);
plot(x,y, x(ddy>0), y(ddy>0),'k.','MarkerSize',30)
xlabel(['(E320)']  );
hold off;
% FILE  e320.m  ends.
```



4 Numeerinen lineaarialgebra

Tässä luvussa kertaamme lineaarialgebraa oppikirjan [LP] mukaisesti (Gaussin eliminointi). Samalla opimme lisää lineaarisen yhtälöryhmän numeriikkaa MATLABin avulla. Hyviä perusteoksia: [S2], [CV], [KMN].

4.1. Matriisilaskentaa (kertausta [LP]:stä). Yhtälöryhmä

$$\text{(4.2)} \qquad \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ \vdots \qquad\quad \vdots \qquad\qquad\quad \vdots \quad \vdots & \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{cases}$$

voidaan kirjoittaa helpommin matriisimuodossa

$$\text{(4.3)} \quad AX = B\,; \quad A = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & & \vdots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad B = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix},$$

missä a_{ij}, x_i, $b_i \in \mathbb{R}$ (tai $\in \mathbb{C}$). $m \times n$ -matriisi A voidaan samastaa lineaarikuvauksen $L\colon \mathbb{R}^n \to \mathbb{R}^m$ kanssa. $m \times n$ -matriisien joukossa $M_{m,n}$ voidaan määritellä yhteenlasku $+$ ja $(M_{m,n}\,, +)$ on Abelin ryhmä. Jos $A \in M_{m,n}$ ja $B \in M_{n,p}$, voidaan määritellä tulo $C \in M_{n,p}$ kaavalla

$$c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj} \quad (i = 1, \dots, m\,, j = 1, \dots, p)\,.$$

Matriisitulo ei ole vaihdannainen, ts. $AB \neq BA$ eräillä A, B.

4.4. Esimerkki. Matriisikertolasku (Falk-kaavion avulla). Olkoon

$$A = \begin{pmatrix} 1 & 2 & -1 \\ 0 & 3 & 5 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 3 & 5 & 0 \\ 0 & -1 & 2 & 1 \\ 2 & 1 & 1 & 0 \end{pmatrix}$$

				1	3	5	0	
				0	−1	2	1	B
				2	1	1	0	
A	1	2	−1	−1	0	8	2	AB
	0	3	5	10	2	11	3	

4.5. Käänteismatriisi. Olkoon $A, B \in M_{n,n}$. Sanotaan, että B on A:n käänteismatriisi A^{-1}, jos

$$AB = BA = I\,,$$

missä $I \in M_{n,n}$ on yksikkömatriisi.

Huom. Kaikilla matriiseilla $M_{n,n}$:ssä ei ole käänteismatriisia, joten $(M_{n,n}\,, \cdot)$ ei ole ryhmä. Jos on olemassa A^{-1}, sanotaan, että A on *säännöllinen.*

4.6. Lineaarisen yhtälöryhmän ratkaisu. Palaamme tarkastelemaan lineaarista yhtälöryhmää (4.3) tapauksessa $m = n$. [LP]:ssä todistetaan

4.7. Lause. Lineaarisella yhtälöryhmällä (4.3), $AX = B$, on tapauksessa $m = n$ yksi ja vain yksi ratkaisu $X = A^{-1}B$ täsmälleen silloin, kun A on säännöllinen.

4.8. Lause. Jos $A \in M_{n,n}$, niin seuraavat ehdot ovat yhtäpitävät:

(1) A on säännöllinen;

(2) $\det A \neq 0$ (det:n määritelmä on esitetty [LP]:ssä);

(3) yhtälöllä $AX = 0$ on ainoana ratkaisuna $X = 0$;

(4) yhtälöllä $AX = B$ on yksikäsitteinen ratkaisu;

(5) A:n pystyvektorit ovat lineaarisesti riippumattomia;

(6) A:n vaakavektorit ovat lineaarisesti riippumattomia.

4.9. Gaussin eliminointi. Jos $A \in M_{n,n}$ ja $\det(A) \neq 0$, niin lauseen 4.8 nojalla tiedetään yhtälöryhmän $AX = B$ ratkaisun olemassaolo ja yksikäsitteisyys. Ratkaisun löytämiseksi voidaan käyttää Gaussin eliminointimenetelmää (1777–1855).

Yhtälöryhmän

$$\left\{\begin{array}{rcl} a_{11}x_1 + \cdots + a_{1n}x_n &=& b_1 \\ a_{21}x_1 + \cdots + a_{2n}x_n &=& b_2 \\ \vdots \qquad\qquad \vdots &\vdots& \vdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n &=& b_n \end{array}\right. \tag{4.10}$$

ratkaisemiseksi toteutetaan ensin eliminointi ja sen jälkeen takaisinsijoitus (back substitution). Oletamme, että $a_{11} \neq 0$.

Eliminoinnin 1. askel: Riveistä j ($j \in \{2, \ldots, n\}$) vähennetään 1. rivi kerrottuna a_{j1}/a_{11}:llä.

Eliminoinnin k. askel: Riveistä j ($j \in \{k+1, \ldots, n\}$) vähennetään rivi k kerrottuna vakiolla $\tilde{a}_{jk}/\tilde{a}_{kk}$ (missä $\tilde{a}_{jk}$:t ovat matriisin edellisissä vaiheissa muuttuneita kertoimia). Jos $\tilde{a}_{kk} = 0$, lopetetaan.

Eliminoinnin jälkeen matriisi on saatu muotoon

$$\left\{\begin{array}{rcl} \tilde{a}_{11}x_1 + \tilde{a}_{12}x_2 + \cdots \cdots \cdots \cdots \cdots + \tilde{a}_{1n}x_n &=& \tilde{b}_1 \\ \tilde{a}_{22}x_2 + \cdots \cdots \cdots \cdots \cdots + \tilde{a}_{2n}x_n &=& \tilde{b}_2 \\ \ddots \qquad\qquad \vdots \quad \vdots \quad \vdots &&\\ \tilde{a}_{n-1,n-1}x_{n-1} + \tilde{a}_{n-1,n}x_n &=& \tilde{b}_{n-1} \\ \tilde{a}_{nn}x_n &=& \tilde{b}_n \end{array}\right. , \tag{4.11}$$

ts. (4.10) on palautettu muotoon (4.11), $\tilde{A}X = \tilde{B}$, missä $\tilde{a}_{ij} \neq 0$ vain, jos $j \geq i$.

Takaisinsijoitus: Ratkaistaan (4.11):n viimeisestä yhtälöstä x_n, sijoitetaan x_n viimeistä edelliseen, jolloin voidaan ratkaista x_{n-1} jne. Jatkamalla näin saadaan x_n, x_{n-1}, ... ja lopulta x_1.

4.12. Esimerkki. Ratkaise Gaussin eliminoinnilla yhtälöryhmä

$$\left\{\begin{array}{rcl} 3x_1 - x_2 + x_3 &=& 2 \\ -x_1 + 3x_2 - 2x_3 &=& 1 \\ 2x_1 + 2x_2 + x_3 &=& -3 \end{array}\right. . \tag{4.13}$$

1. *eliminointivaihe:* 2. rivistä vähennetään 1. rivi $-\frac{1}{3}$:lla kerrottuna ja 3. rivistä vähennetään 1. rivi $\frac{2}{3}$:lla kerrottuna: saadaan

$$\left\{\begin{array}{rcl} 3x_1 - x_2 + x_3 &=& 2 \\ 8x_2 - 5x_3 &=& 5 \\ 8x_2 + x_3 &=& -13 \end{array}\right. , \tag{4.14}$$

missä 2. ja 3. yhtälö on vielä sievennetty kertomalla 3:lla.

2. *eliminointivaihe:* 3. rivistä vähennetään 2. rivi luvulla $8/8 = 1$ kerrottuna; saadaan

$$\left\{\begin{array}{rcl} 3x_1 - x_2 + x_3 &=& 2 \\ 8x_2 - 5x_3 &=& 5 \\ 6x_3 &=& -18 \end{array}\right. . \tag{4.15}$$

Takaisinsijoitusvaiheessa ratkaistaan ensin $x_3 = -3$ 3. yhtälöstä, sitten 2. yhtälöstä $x_2 = -5/4$ ja lopuksi $x_1 = 5/4$ 1. yhtälöstä.

4.16. Gaussin eliminointi MATLABissa. Yhtälöryhmän $AX = B$ ratkaisu MATLABissa tapahtuu käskyllä `X=A\B`. Katsomme nyt, miten sama tehdään Gaussin eliminoinnilla (joka todennäköisesti on numeerisesti huonompi kuin `A\B` -ratkaisu). Pivotointia ei tässä tehdä.

$n \times n$ -yhtälöryhmää ratkaistaessa on hyvä laajentaa matriisi A $n \times (n+1)$ -matriisiksi C, jonka viimeinen sarake on B, alla `b`. Eliminoinnin aikana tapahtuvat kertoimien päivitykset kohdistuvat siis C:n alkioihin.

```
% FILE  e409.m  begins.
% Gauss elimination without pivoting
clear
n = input('Enter an integer n >= 2 ');
A = rand(n,n);  b = rand(n,1);
disp(' Extended matrix');
C = [A, b ]
for j=1:n-1,
    for i=j+1:n,
       if ( abs(C(j,j)) < 1E-10)
          disp('Bad matrix, cond(A) = ')
          cond(A)
          break
       end;
       C(i,:) = C(i,:) - (C(i,j)/C(j,j))*C(j,:)
       pause;
    end;
end;
sol(n) = C(n,n+1)/C(n,n) ;
for i=n-1: -1:1,
   s=0.0;
   for j=i+1:n,
       s=s+C(i,j)*sol(j);
   end;
   sol(i) = (C(i,n+1) -s)/C(i,i);
end;
disp(' Solution = ')
sol
disp(' A x solution = ')
bb=(A*(sol'))'
disp(' Should be = RHS = b ')
b'
```

```
disp(['Residual = ' num2str(norm(b'-bb))])
disp('Difference from Matlab-solution =')
norm(A\b - sol')
% FILE  e409.m  ends.
```

Alla on ohjelman esimerkkiajo.

```
>> e409
Enter an integer n >= 2 3
 Extended matrix

C =

    0.0077    0.4175    0.9304    0.0920
    0.3834    0.6868    0.8462    0.6539
    0.0668    0.5890    0.5269    0.4160


C =

    0.0077    0.4175    0.9304    0.0920
         0  -20.1065  -45.4951   -3.9265
    0.0668    0.5890    0.5269    0.4160


C =

    0.0077    0.4175    0.9304    0.0920
         0  -20.1065  -45.4951   -3.9265
         0   -3.0360   -7.5519   -0.3825


C =

    0.0077    0.4175    0.9304    0.0920
         0  -20.1065  -45.4951   -3.9265
         0         0   -0.6824    0.2104

 Solution =
```

```
sol =

    0.7866    0.8928   -0.3083

 A x solution =

bb =

    0.0920    0.6539    0.4160

 Should be = RHS = b

ans =

    0.0920    0.6539    0.4160

Residual = 5.701e-15
Difference from Matlab-solution =

ans =

   1.5688e-14
```

Huom. `C(i,:)` on matriisin `C` `i`:s vaakarivi.

4.17. Gaussin eliminoinnin epästabiilisuus. Kohdassa 4.16 esitetty Gaussin eliminointimenetelmä toimii huonosti, jos matriisi on lähes singulaarinen. Hilbertin matriisi $H_{ij} = 1/(i+j-1)$ on tunnettu esimerkki lähes singulaarisesta matriisista. Oikeaoppinen MATLAB-ratkaisu: `sol=a\b`. Alla esitettävä pivotointi pyrkii torjumaan haitallisen epästabiilisuuden.

4.18. Gaussin eliminointi ja pivotointi. Tarkka paikka Gaussin eliminoinnissa tulee, jos jossakin laskennan vaiheessa jakaja $\tilde{a}_{kk}$ on itseisarvoltaan pieni. Tällöin eliminointi voi olla numeerisesti epästabiili ja pyöristys- ym. virheet voimistua tavalla, joka näkyy suurena virheenä lopputuloksessa. Tällaista epästabiiliutta voidaan yrittää torjua niin, että jakajaksi $\tilde{a}_{kk}$:n paikalle valitaankin jokin muu luvuista $\tilde{a}_{kj}$ $(j \geq k)$. Sopivan jakajan valintamenetelmiä on erilaisia, mutta yhteisenä piirteenä on pyrkimys pienentää laskentavirheitä.

Jos nyt on valittu $\tilde{a}_{kj_k}$ $(j_k \geq k)$, niin suoritetaan pystyrivien k ja j_k vaihto. Tämä merkitsee myös oikealla puolella B:n komponenttien permutointia. Eliminointivaiheen jälkeen saadaan ratkaisu takaisinsijoituksella, kun vielä suoritetaan em. permutointien käänteispermutoinnit ratkaisuvektoriin. Em. rivien vaihtoja ratkaisumenetelmän numeeristen ominaisuuksien parantamiseksi kutsutaan *pivotoinniksi.* Muitakin pivotointimenettelyjä (mm. sarakkeiden vaihtoja) käytetään.

Ks. algoritmi GAUSSJ, [NR, s. 39].

Pivotointi parantaa huomattavasti Gaussin eliminoinnin numeerisia ominaisuuksia.

4.19. Matriisihajoitelmista. Numeerisessa lineaarialgebrassa on tapana käyttää matriisihajoitelmia, joita on useita mm. matriisin rakenteen mukaan. Matriisihajoitelmat tuovat usein esille matriisin keskeisiä numeerisia piirteitä, jotka eivät muuten tulisi näkyviin. Toisaalta matriisihajoitelmat antavat matriisille moniin jatkokehittelyihin sopivan kompaktin esitysmuodon.

Strang [S2] luettelee yli 10 numeriikassa esiintyvää matriisihajoitelmaa.

MATLABissa on valmiina monia matriisien käsittelyyn tarvittavia toimintoja, mm.

- matriisin käänteismatriisi, determinantti, ominaisarvot, cond(A);
- matriisihajoitelmat: LU, QR, SVD, CHOL;
- versiossa 4.0: harvojen matriisien käsittely;
- helppous matriisien muodostamisessa ja laskutoimituksissa (yleensä ei tarvita alkioittaisia operaatioita), valmiit "standardimatriisit", esimerkiksi `zeros(4,5)`, `ones(4,5)`, `hilb(10)`, `invhilb(3)` jne.

Massiivisessa numeerisessa laskennassa yhtälöryhmät voivat sisältää esim. 500 000 muuttujaa. Näin laajojen lineaaristen yhtälöryhmien ratkaisu vaatii "täsmälaskentaa", joka ottaa huomioon yhtälöryhmän rakenteen (symmetria, lohkomatriisi, tridiagonaalimatriisi) ja erityisesti sen, onko matriisi harva. (Matriisi on harva, jos suurin osa sen alkioista on nollia.)

4.20. Singulaariarvohajoitelma SVD. Esittelemme nyt — ilman todistuksia — matriisihajoitelman, jonka avulla saadaan monipuolista tietoa matriisin numeerisista ominaisuuksista. Tämä matriisihajoitelma, SVD, on tietokoneiden yleistyessä noussut keskeiseen asemaan numeerisessa lineaarialgebrassa, mutta sitä ei välttämättä esitetä lineaarialgebran peruskurssilla. Käsin laskien SVD:n käyttö on liian hidasta jo 4×4 -matriiseille, ja sujuva käyttö edellyttää tietokoneita.

Aluksi palautamme [LP]:stä mieliin, että $m \times n$ -matriisin $A = (a_{ij})$ *transpoosi* A^T on $n \times m$ -matriisi $(A^T)_{ij} = a_{ji}$. Neliömatriisi B on *ortogonaalinen*, jos se on säännöllinen (ts. $\det(B) \neq 0$) ja jos $B^T = B^{-1}$.

4.21. Lause (SVD, ei tod.)**.** Reaaliselle $m \times n$ -matriisille A on olemassa ortogonaalinen $m \times m$ -matriisi U ja ortogonaalinen $n \times n$ -matriisi V s.e.

$$A = USV\,,$$

missä S on sellainen $m \times n$ -diagonaalimatriisi, että jos merk. $S = \operatorname{diag}(s_1, \ldots, s_p)$ (missä $p = \min\{m, n\}$), niin $s_1 \geq s_2 \geq \ldots \geq s_p \geq 0$. Jos

$$s_1 \geq s_2 \geq \ldots \geq s_r > s_{r+1} = \ldots = s_p = 0$$

ja jos A:lla on käänteismatriisi, niin määritellään

$$\operatorname{cond}(A) = s_1/s_r\,. \tag{4.22}$$

4.23. Kuntoisuusluku $\operatorname{cond}(A)$**.** Kaavassa (4.22) määritelty $\operatorname{cond}(A)$ on nimeltään *kuntoisuusluku*. Aina $\operatorname{cond}(A) \in [1, \infty]$; $\operatorname{cond}(A) = 1$ yksikkömatriisille ja $\operatorname{cond}(A) = \infty$ singulaariselle matriisille, so. matriisille, jonka determinantti on 0. Tapauksessa $\det(A) \neq 0$ on $\operatorname{cond}(A)$ kvantitatiivinen mitta sille, "kuinka paha" A on numeerisilta ominaisuuksiltaan. (Välihuomautus: $\det(A)$ ei ole hyvä mitta tähän tarkoitukseen, sillä A ja λA ovat numeerisilta ominaisuuksiltaan samanveroisia, kun taas $\det(\lambda A) = \lambda^n \det(A)$, jos A on $n \times n$ -matriisi.)

MATLABin `svd` palauttaa kutsulla

```
[u,s,v] = svd(a);
```

`a`:n SVD:n ja kutsulla

```
b = svd(a)
```

pelkästään singulaariarvot.

Seuraavalla pienellä koodinpätkällä voimme todeta, että MATLAB tosiaan käyttää kaavaa (4.22) $\operatorname{cond}(A)$:n laskemiseen.

```
% FILE  e416.m  begins.
clear;
info =[];
for n = 3:30
```

```
    a= rand(n,n);  b = svd(a);   c= cond(a);
    d= max(b)/min(b)-c;
    info = [info d];
end;
m1 = max(abs(info)); m2 = min(info);
plot(3:30, info),
title(['Max is ' num2str(m1) ', min is ' num2str(m2)]), pause;
xlabel('E416'),pause;
% FILE  e416.m  ends.
```

4.24. Esimerkki. Yhtälöryhmän

$$\begin{cases} 2.000x + 0.6667y &= 2.000; \\ 1.000x + 0.3333y &= 1.000 \end{cases}$$

kerroinmatriisin a kuntoisuusluku on $\text{cond}(a) = 5.5556 \cdot 10^4$ ja yhtälöryhmällä on yksikäsitteinen ratkaisu $x = 1.000$, $y = 0.000$. Sen sijaan yhtälöryhmällä

$$\begin{cases} 2.000x + 0.6666y &= 2.000 \\ \frac{1}{2}(2.000x + 0.6666y) &= \frac{1}{2} \cdot 2.000 \end{cases}$$

on äärettömän monta ratkaisua

$$\begin{cases} x &= 1.000 - 0.3333k \\ y &= k, \quad k \in \mathbb{R}^1 \end{cases} .$$

Huomaa, että $\text{cond}(a)$ on suuri ja siis tehtävä epästabiili.

4.25. Kuntoisuusluvun muuttaminen. Konstruoi jokin 5×5 -matriisi, jolla on ennalta annettu kuntoisuusluku.

Ratkaisu. Ajatuksena on (1) generoida satunnaismatriisin a SVD, $a = usv$, ja (2) muuttaa singulaariarvoja s suotuisasti, tuloksena s_1, niin että (3) muokatulla matriisilla $b = us_1v$ on haluttu $\text{cond}(b)$.

MATLAB-toteutus:

```
% FILE  e418.m  begins.
clear;
n =5;
a= rand(n,n); c = input('Please enter a number c > 1: ');
[u, s, v] = svd(a);
```

```
for j=1:n
        ss(j,j) = s(1,1)- (j-1)*(s(1,1) -s(1,1)/c)/(n-1);
end;
aa = u*ss*v';
disp('The matrix ')
aa
disp([' has condition number = ' num2str(cond(aa)) ]), pause;
% FILE  e418.m  ends.
```

4.26. cond(A)**:n vaikutus tarkkuuteen.** Numeerisessa lineaarialgebrassa johdetaan virhearvioita lineaarisen yhtälöryhmän ratkaisun tarkkuudelle kuntoisuusluvun avulla. Haluamme nyt selvittää kokeellisesti, miten tarkkuus riippuu kuntoisuusluvusta. Toteutusidea:

(1) generoidaan kuten 4.18:ssa 10×10 -matriisi a, jolle $\mathrm{cond}(a) = 10^j$;

(2) merkitään $b = a[1, 1, \ldots, 1]^T$, jolloin yhtälöryhmän $ax = b$ ratkaisu on

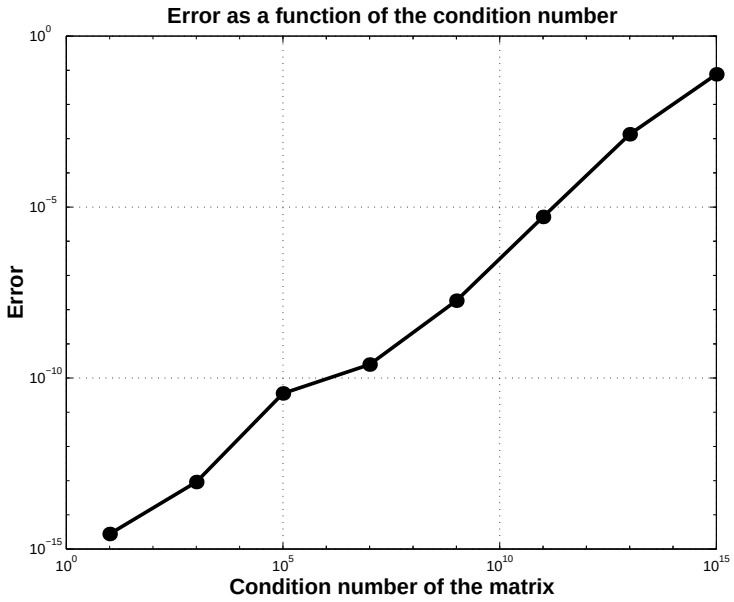
$$x_0 = [1, 1, \ldots, 1]^T;$$

(3) ratkaistaan $ax = b$ numeerisesti ja verrataan numeerista ratkaisua x_1 x_0:aan. Erotusvektorin pituus $|x_1 - x_0|$ on ratkaisun virhe.

MATLAB-toteutus:

```
% FILE  e419.m  begins.
clf;
data = [];
n =20;
a= rand(n,n);
[u, s, v] = svd(a);
for p =1:2:15
    c= 10^p;
for j=1:n
    ss(j,j) = s(1,1)- (j-1)*(s(1,1) -s(1,1)/c)/(n-1);
end;
    aa = u*ss*v';
    b =aa*ones(n,1);
    numsol=aa\b;
```

```
    d = abs(norm(numsol-ones(n,1)));
    data = [data;   [c  d]];
end;
x = data(:,1);  y = data(:,2);
%axes('FontSize',[15],'FontWeight','bold');  hold on;

pic=loglog(x,y,x,y,'k.','MarkerSize',30); grid;
xlabel(' Condition number of the matrix ', 'FontSize',[15],...
'FontWeight','bold');
ylabel(' Error ', 'FontSize',[15],...
       'FontWeight','bold');
title('Error as a function of the condition number',...
      'FontSize',[15],'FontWeight','bold');
set(pic,'LineWidth',2.5);
% FILE  e419.m  ends.
```



5 Epälineaariset yhtälöt

Perehdymme tässä luvussa epälineaaristen yhtälöiden ja yhtälöryhmien ratkaisujen numeeriseen approksimointiin. Perusongelma: Etsi $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$ siten, että yhtälöryhmä

$$\left\{ \begin{array}{rcl} f_1(x_1, \ldots, x_n) & = & b_1 \\ \vdots & \vdots & \vdots \\ f_n(x_1, \ldots, x_n) & = & b_n \end{array} \right. \tag{5.1}$$

toteutuu. Yhtälöryhmän (5.1) toteuttava vektori x on yhtälöryhmän *juuri*. Edellisessä luvussa tarkasteltiin tapausta, missä f_j:t ovat lineaarisia funktioita. Nyt on pääpaino tapauksessa, missä f_j:t ovat epälineaarisia.

5.2. Huomautuksia.

(1) Ennen numeeriseen ratkaisuun ryhtymistä on ratkaisun olemassaolo ja yksikäsitteisyys selvitettävä. Lineaarisen yhtälöryhmän tapauksessa nämä seikat saatiin lineaarialgebrasta.

(2) Mitään yleispätevää ratkaisumenetelmää ei tunneta epälineaariselle yhtälöryhmälle (5.1), jos dimensio $n > 1$. Tapauksessa $n = 1$ välinpuolitusmenetelmä on yleiskäyttöinen. Melko yleisin oletuksin funktioista f_j on olemassa iteratiivisia menetelmiä, jotka suppenevat kohti juurta, mikäli alkuarvaus on kyllin hyvä.

(3) Yhtälöryhmän (5.1) tarkkaa ratkaisua ei yleensä voida saavuttaa, ja numeerista ratkaisua etsittäessä on käytettävä jotain sopivaa katkaisukriteeriä.

(4) Lineaarisen yhtälöryhmän ratkaisun kompleksisuus Gaussin eliminoinnilla on luokkaa n^3. Epälineaarisessa tapauksessa kompleksisuutta on mahdoton arvioida.

(5) Ratkaisua etsittäessä voidaan eräissä tapauksissa käyttää Newtonin algoritmia. Newtonin algoritmin suppeneminen riippuu erittäin monimutkaisella tavalla alkuarvosta (ks. kohta 5.33 ja kuva [NR]:n sivulla 368).

5.3. Banachin kiintopistelause (1892–1945)**.** Suuri joukko iteraatiomenetelmiä voidaan kirjoittaa muotoon

$$x_{k+1} = F(x_k) \qquad (k = 0, 1, 2, \ldots)\,, \tag{5.4}$$

missä x_k voi olla luku, vektori tai jopa funktio. Jos jono x_k suppenee kohti arvoa x_0, niin F:n jatkuvuuden nojalla pätee, että

$$x_0 = F(x_0)\,. \tag{5.5}$$

Ts. yo. *peräkkäisten approksimaatioiden menetelmällä* (5.4) löydetään yhtälön (5.5) juuri, jos jono (x_k) suppenee. Seuraavassa esitetään riittävä ehto suppenemiselle. Pistettä x_0 sanotaan F:n *kiintopisteeksi* ja iteraatiota (5.4) myös *kiintopisteiteraatioksi*.

5.6. Määritelmä. Vektoriavaruus B kerroinkuntana $\mathbb{C}$ on *Banachin avaruus,* jos on määritelty normi $\|\ \|$ s.e.

(1) $\|x\| \geq 0 \quad \forall\ x \in B\,,$

(2) $\|x\| = 0 \Leftrightarrow x = 0\,,$

(3) $\|\gamma x\| = |\gamma|\,\|x\| \quad \forall x \in B \quad \forall \gamma \in \mathbb{C}\,,$

(4) $\|x + y\| \leq \|x\| + \|y\| \quad \forall\ x, y \in B$

ja lisäksi jokainen Cauchy-jono B:n alkioita suppenee kohti B:n alkiota.

Palautetaan mieleen, että B:n jono (x_k) on Cauchy-jono, jos jokaiselle $\varepsilon > 0$ on olemassa luku n_ε siten, että

$$\|x_m - x_n\| < \varepsilon\,, \quad \text{kun}\ \ m, n \geq n_\varepsilon\,.$$

5.7. Esimerkki. (1) Reaalilukujen joukko normina $|x|$ on Banachin avaruus.

(2) Euklidinen avaruus $\mathbb{R}^n$ normina $\|x\| = (\sum x_i^2)^{1/2}$ on Banachin avaruus. ($\mathbb{R}^n \setminus \{0\}$ ei ole Banachin avaruus.)

(3) Välillä $I = [0, 1]$ määriteltyjen jatkuvien funktioiden joukko $C(I)$ normina

$$\|f\| = \max\{\,|f(x)| : x \in I\,\}$$

on Banachin avaruus.

5.8. Määritelmä. Olkoon A Banach-avaruuden B suljettu osajoukko ja $F\colon A \to B$ kuvaus. Sanomme, että F on Lipschitz-jatkuva A:ssa vakiolla $L \in (0, \infty)$, jos

$$\|F(x) - F(y)\| \leq L\|x - y\| \quad \forall\ x, y \in A\,.$$

Sanotaan, että F on *kutistava*, jos $L < 1$.

5.9. Lause (Banachin kiintopistelause)**.** Olkoon A Banach-avaruuden B suljettu osajoukko ja $F\colon A \to A$ kutistava kuvaus. Silloin

(1) on olemassa täsmälleen yksi piste $s \in B$, jolle $F(s) = s$ (kiintopiste);

(2) jokaisella alkuarvolla x_0 jono (5.4) suppenee kohden kiintopistettä s;

(3) jos L on kuten 5.8:ssa, pätee arvio

$$\|s - x_k\| \leq \frac{L^{k-l}}{1-L}\,\|x_{l+1} - x_l\| \qquad (0 \leq l < k)\,.$$

Todistus. Määritelmästä (5.4) saamme

$$\|x_{k+1} - x_k\| = \|F(x_k) - F(x_{k-1})\| \leq L\|x_k - x_{k-1}\| =$$

$$L\|F(x_{k-1}) - F(x_{k-2})\| \leq L^2\|x_{k-1} - x_{k-2}\| = \cdots\,,$$

josta edelleen induktiolla

$$\|x_{k+1} - x_k\| \leq L^{k-l}\|x_{l+1} - x_l\| \qquad (0 \leq l \leq k)\,.$$

Osoitamme, että (x_k) on Cauchy-jono:

$$\|x_{k+m} - x_k\| = \|x_{k+m} - x_{k+m-1} + x_{k+m-1} - \cdots - x_k\|$$

$$\leq \sum_{j=k}^{k+m-1} \|x_{j+1} - x_j\| \leq L^k(L^{m-1} + L^{m-2} + \cdots + L + 1)\|x_1 - x_0\| \tag{5.10}$$

$$= L^k \frac{1 - L^m}{1 - L}\,\|x_1 - x_0\|\,.$$

Koska $L \in (0,1)$, nähdään että (x_k) on Cauchy-jono. Näin ollen suljettu osajoukko A sisältää raja-arvon

$$s = \lim x_k\,, \quad s \in A\,,$$

ja $F(s) = F(\lim x_k) = \lim F(x_k) = \lim x_{k+1} = s$, joten olemme osoittaneet, että jokaisella alkuarvolla jono (x_k) suppenee kohti kiintopistettä $s \in A$.

Osoitamme nyt, että kiintopisteitä on täsmälleen yksi. Jos $s_j \in A$ $(j = 1, 2)$ ovat kiintopisteitä ja $\|s_1 - s_2\| > 0$, saadaan

$$\|s_1 - s_2\| = \|F(s_1) - F(s_2)\| \leq L\|s_1 - s_2\|\,,$$

josta seuraa $L \geq 1$, joka on ristiriita. Siis $s_1 = s_2$.

Kaavan (5.10) johto antaa myös kaavan

$$\|x_{k+m} - x_k\| \leq L^{k-l} \frac{1 - L^m}{1 - L} \|x_{l+1} - x_l\| \quad (m \geq 1)\,.$$

Pidetään k ja l kiinteinä ja annetaan $m \to \infty$. Silloin $x_{k+m} \to s$, ja arvio 5.9 (5) seuraa. □

5.11. Esimerkki. Etsitään yhtälön $x = e^{-x}$ ratkaisu. Merkitään $F(x) = e^{-x}$. Valitaan $A = [0.5,\ 0.69]$, jolloin $F\colon A \to A$ ja väliarvolause implikoi

$$|F(x) - F(y)| \leq L|x - y|\,; \quad L = \max_{x \in A} |-e^{-x}| = e^{-0.5}\,.$$

Koska $L \approx 0.607 < 1$, saadaan 5.9:n nojalla kiintopiste esimerkiksi iteraatiosta $x_0 = 0.55$, $x_{k+1} = F(x_k)$. Ratkaisuksi saadaan $s = 0.5671439$.

5.12. Esimerkki. Kiintopisteiteraatiota (5.4) voidaan käyttää, ainakin periaatteessa, myös lineaaristen yhtälöryhmien ratkaisemiseen (mikäli tarvittavat oletukset ovat voimassa). Palautetaan ensin mieleen $m \times n$ -matriisin *a normi* $\|a\|$:

$$\|a\| = \sup\{\, |ax| \,:\, |x| = 1 \,\}\,, \quad |x| = \left(\sum x_i^2\right)^{1/2}\,.$$

Lineaarinen yhtälöryhmä $ax = b$ voidaan aina kirjoittaa muotoon

$$x = x + ax - b, \quad x = (x_1, \ldots, x_n)^T\,.$$

Merkitään $F(x) = x + ax$. Jos nyt $\|F\| < 1$, niin Banachin kiintopistelausetta voidaan soveltaa.

Yhtälöryhmä

$$\begin{cases} 8x_1 + x_2 - 2x_3 - 8 = 0 \\ x_1 + 18x_2 - 6x_3 + 10 = 0 \\ 2x_1 + x_2 + 16x_3 + 2 = 0 \end{cases}$$

voidaan muuntaa ekvivalenttiin muotoon

$$\begin{cases} x_1 = 0.2x_1 - 0.1x_2 + 0.2x_3 + 0.8 \\ x_2 = -0.05x_1 + 0.1x_2 + 0.3x_3 - 0.5 \\ x_3 = -0.1x_1 - 0.05x_2 + 0.2x_3 - 0.1 \end{cases} \quad \Leftrightarrow \quad X = \underbrace{AX + B}_{F(X)}\,;$$

$$A = \begin{bmatrix} 0.2 & -0.1 & 0.2 \\ -0.05 & 0.1 & 0.3 \\ -0.1 & -0.05 & 0.2 \end{bmatrix} \quad \overset{\text{MATLAB}}{\Longrightarrow} \quad \|A\| = 0.4126\,.$$

Koska $\|F(X) - F(Y)\| = \|AX - AY\| \leq \|A\|\,\|X - Y\|$ ja $F\colon \mathbb{R}^3 \to \mathbb{R}^3$ ja $\mathbb{R}^3$ on Banach-avaruus, niin ehdosta $\|A\| < 1$ seuraa, että kiintopisteiteraatio (5.4) suppenee kaikilla alkuarvoilla.

Esitetty menetelmä soveltuu vain sellaisille lineaarisille yhtälöryhmille, jotka voidaan kirjoittaa muotoon $X = AX + B$, missä $\|A\| < 1$, eli $(I - A)X = B$, missä I on yksikkömatriisi ja $\|A\| < 1$.

```
% FILE  e510.m  begins.
% Fixed point iteration for a linear system.
%
%  8x + y - 2z - 8   = 0
%  x + 18y - 6z + 10 = 0
%  2x + y + 16z + 2  = 0
%
%  or   F u = G;  F =[8 1 -2; 1 18 -6; 2 1 16]
%                 G = [8 -10 -2]'
%
%  is equivalent to:
%  x = .2x -.1y +.2z + .8
%  y = -.05x + .1y +.3z -.5
%  z = -.1x -.05y +.2z - .1
x0 =0.55; y0 = 0.1; z0 =1;
v =[x0 y0 z0]';
a =[.2 -.1 .2; -.05 .1 .3; -.1 -.05 .2]; b = [.8 -.5 -.1]';
data =[];
for k =1:10
    vold =v;
    v =a*v+b;
    data = [data; v'];
end;
disp('Fixed point iteration   (E510) ');
disp(['   x                y               z ']);
fprintf(1, ' %12.8f     %12.8f    %12.8f  \n',data(1:10,1:3)');
%disp(['|x(30) -x(31)| = ' num2str(abs(xold-x))]);
F =[8 1 -2; 1 18 -6; 2 1 16];
```

```
G = [8 -10 -2]';
u = F\G;
disp(['MATLAB solution =' mat2str(u',5) ])
disp(['Error = ' num2str(norm(data(10,:)' -u) )])
% FILE  e510.m  ends.
```

```
>> e510
Fixed point iteration   (E510)
   x                y                z
   1.10000000      -0.21750000      0.04000000
   1.04975000      -0.56475000     -0.19112500
   1.02820000      -0.66630000     -0.21496250
   1.02927750      -0.68252875     -0.21249750
   1.03160888      -0.68346600     -0.21130081
   1.03240821      -0.68331729     -0.21124775
   1.03256382      -0.68332646     -0.21132451
   1.03258051      -0.68335819     -0.21135496
   1.03258093      -0.68337133     -0.21136113
   1.03258109      -0.68337452     -0.21136175
MATLAB solution =[1.0326 -0.68338 -0.21136]
Error = 6.875e-07
```

5.13. Polynomiyhtälön juuret. Polynomiyhtälöt muodostavat tärkeän alaluokan epälineaarisista yhden muuttujan yhtälöistä. Niiden ratkaisemiseksi on kehitetty polynomien erityisluonteen huomioon ottavia ratkaisumenetelmiä, ks. esim. [NR, s. 369]. MATLABissa polynomiyhtälön $p(z) = 0$ juuret saadaan komennolla "`roots(c)`", kun $p(z) = \sum_{j=1}^{n+1} c(j) z^{n+1-j}$. Ks. 2.3.

5.14. Juurien haarukointi (bracketing). Tarkastelemme nyt yhden reaalimuuttujan funktiota $f(x)$, jonka nollakohtaa etsimme. Sanomme, että f:n nollakohta on haarukoitu välille $[a, b]$, jos f on tällä välillä jatkuva ja jos $f(a)f(b) < 0$. Tällöin nimittäin *Bolzanon* lauseen (1781–1848) nojalla f:llä on ainakin yksi nollakohta välillä $[a, b]$.

5.15. Huomautus. (1) Nollakohtia voi olla useita, jopa ääretön määrä, kuten funktiolla f, missä $f(x) = x^2 \sin(1/x)$, kun $x \in [-1, 1] \setminus \{0\}$ ja $f(0) = 0$, on välillä $[-\frac{2}{\pi}, \frac{2}{\pi}]$.

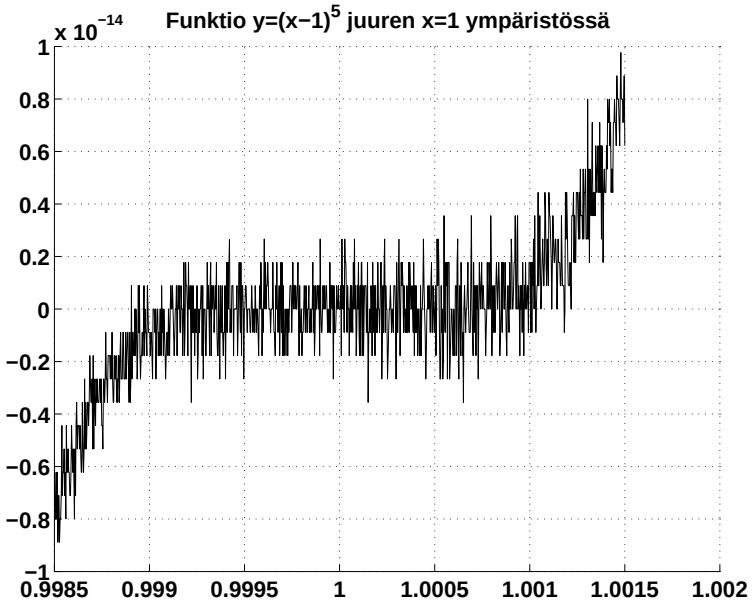
(2) Bolzanon lause ei sovellu funktiolle $g(x) = 1/x$ $(x \in [-1,1] \setminus \{0\})$, vaikka $g(-1) = -1$ ja $g(1) = 1$. Syy: sitä ei voida jatkaa jatkuvasti pisteeseen $x = 0$.

5.16. Juuren tarkkuudesta. Liukulukuaritmetiikan luonteesta johtuen funktion nollakohta ei välttämättä ole laskettavissa mepsin tarkkuudella. Tarkastelemme esimerkiksi funktion $y = (x-1)^5$ käyttäytymistä juuren $x = 1$ ympäristössä.

```
function y =gg(x)
% Analytically gg(x) = (x-1)^5
y = x.^5 -5*x.^4 +10*x.^3 -10*x.^2 +5*x-1;
% FILE  gg.m  ends.
```

Funktion kuvaaja saadaan piirretyksi seuraavilla komennoilla:

```
% FILE  tst.m  begins.
x1=0.9985; x2= 1.0015; dx=(x2-x1)/1000;
x=x1:dx:x2; y=gg(x);
clf;
axes('FontSize',[15],'FontWeight','bold');  hold on;
txt=['Funktio y=(x-1)^5 juuren x=1 ympäristössä' ];
title(txt,'FontSize',[15]);
plot(x,y), grid on   %  + ADJUST MANUALLY
hold off;
% FILE  tst.m  ends.
```



Tällainen käyttäytyminen on syytä pitää mielessä juuria numeerisesti etsiessä.

5.17. Välinpuolitusmenetelmä juuren haussa. Bolzanon lauseeseen perustuu välinpuolitusmenetelmän käyttö juuren haussa, kun juuri (eli nollakohta) on haarukoitu välille $[a, b]$ $(a < b)$. Aseta $a_1 = a$, $b_1 = b$.

Do $n = 1, \ldots, NMAX$

$$
\begin{aligned}
&w \leftarrow (a_n + b_n)/2 \\
&\text{Jos } f(a_n)f(w) < 0, \text{ niin } a_{n+1} \leftarrow a_n,\ b_{n+1} \leftarrow w. \\
&\text{Jos } f(b_n)f(w) < 0, \text{ niin } a_{n+1} \leftarrow w,\ b_{n+1} \leftarrow b_n.
\end{aligned}
$$

Juuren s approksimaatio on $\frac{1}{2}(a_{NMAX+1} + b_{NMAX+1})$ ja approksimaation virhe $< (b-a) \cdot 2^{-NMAX-1}$. Ts. jos halutaan virhe $< \varepsilon$, valitaan $NMAX$ s.e.

$$NMAX = \min\{n \in \mathbb{N} : \ (b-a) < \varepsilon \cdot 2^{n+1}\}\,.$$

Algoritmin implementointi on esim. `rtbis` teoksessa [NR, s. 354]. Ks. myös kohta 2.24.

5.18. Muita menetelmiä juuren hakuun. Välipuolitusmenetelmä ei vaadi funktion derivoituvuutta, vaan se soveltuu kaikille jatkuville funktioille. Muita jatkuville funktioille soveltuvia menetelmiä on useita, mm.

- Brentin menetelmä [NR, s. 361],
- false position ja sekanttimenetelmät [NR, s. 356–7],
- MATLABin `fzero`.

5.19. Newtonin menetelmä (1642–1727)**.** Kenties kuuluisin juurenhakumenetelmä on *Newtonin* menetelmä, joka tunnetaan myös Newtonin–Raphsonin menetelmän nimellä. Menetelmä vaatii funktion derivoituvuuden, kun taas kohdissa 5.17 ja 5.18 esitellyt menetelmät eivät vaadi. Menetelmän johto perustuu Taylorin kaavaan (1685–1731)

$$f(x+\delta) \approx f(x) + f'(x)\delta + \frac{f''(x))}{2}\delta^2 + \cdots\,. \tag{5.20}$$

Vaatimus $f(x+\delta) = 0$ (5.20):ssä johtaa kaavaan

$$\delta = -\frac{f(x)}{f'(x)}\,,$$

kun δ on pieni. Jos x_0 on alkuarvo, niin x_{n+1} määritellään kaavalla

$$x_{n+1} = x_n + \delta_n = x_n - \frac{f(x_n)}{f'(x_n)}\,. \tag{5.21}$$

Kaavaa (5.21) johdettaessa oli x reaalimuuttuja, mutta samaa kaavaa (5.21) voidaan käyttää myös kompleksimuuttujalle z. Kaavalla (5.21) on luonnollinen vastineensa myös tapauksessa, missä $x \in \mathbb{R}^n$ on vektori ja $f\colon \mathbb{R}^n \to \mathbb{R}^n$, johon jatkossa palataan. Kysymys jonon (5.21) suppenemisesta on epätriviaali ja nykyisin tunnettavat vastaukset ovat osin monimutkaisia: kompleksialueen yhtälöä $z^3 = 1$ Newtonin menetelmällä ratkaistaessa johdutaan iteraatioon

$$z_{n+1} = z_n - \frac{z_n^3 - 1}{3z_n^2} = \frac{1 + 2z_n^3}{3z_n^2}\,. \tag{5.22}$$

Jonon (5.22) suppenemiseen johtavat alkuarvot muodostavat *fraktaali*-joukon, jonka rakenne on geometrisesti mielenkiintoinen (ks. kohta 5.33 ja kuva [NR, s. 368]). Vrt. [SB].

Ns. *Newtonin–Kantorovitshin* lause antaa yleisessä n-ulotteisessa tapauksessa erään riittävän ehdon sille, kuinka läheltä juurta s alkuarvo x_0 on valittava. Todistuksen idea perustuu Banachin kiintopistelauseen käyttöön apufunktiolle, joka 1-ulotteisessa tapauksessa on

$$F(x) = x - f(x)/f'(x)\,. \tag{5.23}$$

Banachin kiintopistelauseen soveltamista varten tarvitaan mm. vaatimuksen $\|F'(x)\| < k < 1$ voimassaolo. Newtonin iteraatio palautuu nyt kiintopisteiteraatioon

$$x_{n+1} = F(x_n)\,, \tag{5.24}$$

jonka kiintopiste s on yhtälön $f(x) = 0$ juuri.

Ilman todistusta esitämme seuraavan lauseen.

5.25. Lause. Olkoon f kolmesti jatkuvasti derivoituva välillä (a, b), joka sisältää yhtälön $f(x) = 0$ juuren s; oletamme, että $f'(s) \neq 0$ (ts. juuri s on yksinkertainen). Silloin on olemassa $\delta > 0$ s.e. funktio $F\colon I \to I$, missä $I = [s-\delta, s+\delta] \subset (a, b)$ ja $F(x) = x - f(x)/f'(x)$, on kutistava. Jokaisella alkuarvolla $x_0 \in I$ Newtonin jono (5.21) suppenee kohti juurta s.

Lauseessa 5.25 esiintyvän funktion F derivaatta on

$$F'(x) = 1 - (f'(x)^2 - f(x)f''(x))/f'(x)^2 = f(x)f''(x)/f'(x)^2\,,$$

joten selvästi $F'(s) = 0$ ja tehtyjen jatkuvuusoletusten nojalla löytyy s:n ympäristö, jossa $|F'(x)| < \frac{1}{2}$. Yo. lauseen todistus seuraa helposti tästä huomiosta ja Banachin kiintopistelauseesta.

5.26. Newtonin menetelmä n-ulotteisessa tapauksessa. Yhtälöryhmä

$$\left\{\begin{array}{rcl} f_1(x_1,\ldots,x_n) &=& 0 \\ f_2(x_1,\ldots,x_n) &=& 0 \\ \vdots & \vdots & \vdots \\ f_n(x_1,\ldots,x_n) &=& 0 \end{array}\right. \tag{5.27}$$

voidaan merkinnöin $x = (x_1,\ldots,x_n)^T$, $f(x) = (f_1(x),\ldots,f_n(x))^T$ kirjoittaa

$$f(x) = 0\,. \tag{5.28}$$

Olettamalla, että f on differentioituva, voidaan kirjoittaa

$$f(x+h) = f(x) + f'(x)h + O(h^2) = f(x) + J_f(x)h + O(h^2)\,, \tag{5.29}$$

missä $J_f(x)$ on *Jacobin* (1804–1851) $n \times n$ -matriisi

$$J_f(x)_{ij} = \frac{\partial f_i(x)}{\partial x_j}\,. \tag{5.30}$$

Kaava (5.29) on tuttu [LP]:stä; laajemmin kaava (5.29) tulee käyttöön useamman muuttujan funktioiden differentiaalilaskennassa. Usein merkitään myös $f'(x) = J_f(x)$. Newtonin menetelmän johto n-ulotteisessa tapauksessa etenee nyt samoin kuin 1-ulotteisessa tapauksessa: vaaditaan että h on pieni ja $f(x+h) = 0$, jolloin ehdosta $O(h) \to 0$, kun $h \to 0$, ja kaavasta (5.29) seuraa, että

$$J_f(x)h = -f(x) \quad \Leftrightarrow \quad h = -J_f(x)^{-1}f(x)\,. \tag{5.31}$$

Siis saadaan iteraatio

$$x_{k+1} = x_k - J_f(x_k)^{-1}f(x_k)\,, \tag{5.32}$$

joka on Newtonin iteraatio n-ulotteisessa tapauksessa.

Yhden iteraatioaskeleen (5.32) suorittamiseksi joudutaan näin ollen laskemaan f:n kaikki 1. kertaluvun osittaisderivaatat ja lisäksi kääntämään näiden muodostama matriisi $J_f(x)$. Numeerisesti iteraatioaskel on siis melko raskas, sen kompleksisuus on luokkaa n^3.

5.33. Huomautuksia. (1) Newtonin iteraatioissa (5.21) ja (5.32) joudutaan vaikeuksiin, jos $f'(x_n) = 0$ tai $J_f(x_n)$ on singulaarinen.

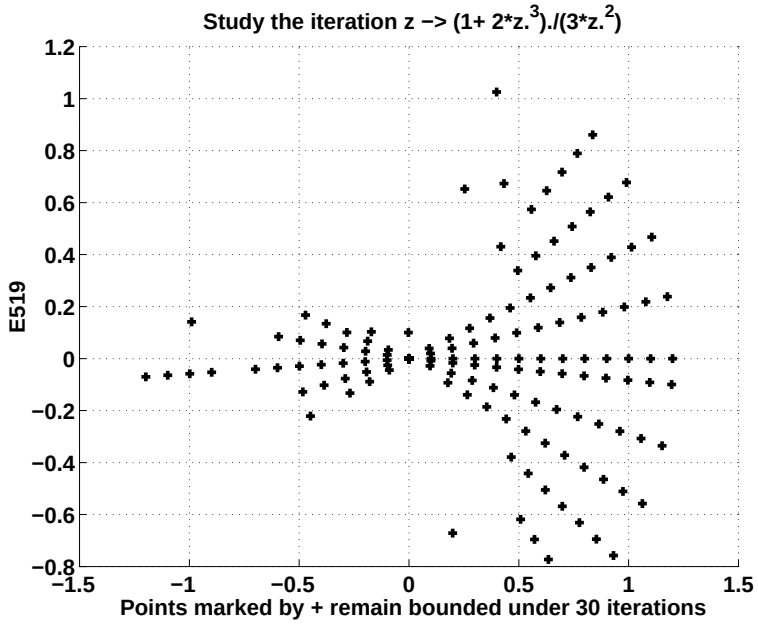
(2) Yhden kompleksimuuttujan yhtälö $f(z) = 0$ voidaan palauttaa kahden reaalimuuttujan yhtälöryhmän tapaukseen tarkastelemalla erikseen reaali- ja imaginaariosia.

(3) Yhtälön $z^3 = 1$ ratkaisu Newtonin menetelmällä johtaa iteraatioon

$$z_{n+1} = \frac{1 + 2z_n^3}{3z_n^2} \quad \text{(vrt. [NR, s. 368])}.$$

Seuraavassa ohjelmassa piirrämme tasoon ne pisteet, jotka pysyvät em. Newtonin iteraatiossa rajoitettuina.

```
% FILE  e0519.m  begins.
w =[];
for r= 0.1: 0.1: 1.2;
the = 0: 0.2: 2*pi;
zsta = r*exp(i*the); z=zsta;
for k =1:30
   z =(1+ 2*z.^3)./(3*z.^2);
end;
   tst=abs(z-1)<1;
   w= [w; tst.*zsta];
end;
clf;
axes('FontSize',[15],'FontWeight','bold');  hold on;
plt=plot(w,'k+');
grid;
txt=[' Study the iteration z -> (1+ 2*z.^3)./(3*z.^2)'];
title(txt,'FontSize',[15]);
ylabel('E519'),
xlabel(' Points marked by + remain bounded under 30 iterations');
set(plt,'LineWidth', 2.5);
hold off;
% FILE  e0519.m  ends.
```



5.34. Numeerinen Jacobin matriisi. Edellä kohdissa 3.15–3.20 esitettiin joitakin menetelmiä reaaliarvoisen yhden muuttujan funktion derivaatan numeeriselle approksimoinnille. Periaatteessa samoin voidaan n muuttujan funktiolle $f\colon \mathbb{R}^n \to \mathbb{R}^m$ muodostaa numeerinen Jacobin matriisi $f'(x) = J_f(x)$,

$$J_f(x) = \begin{bmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_m(x)}{\partial x_1} & \cdots & \frac{\partial f_m(x)}{\partial x_n} \end{bmatrix} \quad \forall x = (x_1, \ldots, x_n)\,,$$

missä kukin $\partial f_j(x)/\partial x_i$ lasketaan jollakin 3.15–3.20:ssä esitetyistä tavoista. Yksi tapa muodostaa J_f numeerisesti funktiolle $f\colon \mathbb{R}^n \to \mathbb{R}^m$ on seuraava.

```
function Jf = numjaco(f,m,x,n)
% f is a function with m components,
% x is a vector with n components,
% the result is an m by n matrix.
Jf = ones(m,n);   h = 10^(-4);
for j =1:n
    e = zeros(1,n); e(j) = 1;
    Jf(:,j) = (feval(f,x+h*e)-feval(f,x-h*e) )/(2*h);
end;
% FILE  numjaco.m  ends.
```

Funktiota `numjaco` voidaan soveltaa esim. seuraavasti määriteltyyn funktioon:

```
function y = f520(x)
y(1,1) = 3*x(1) -cos(x(2)*x(3)) -0.5;
y(2,1) = x(1)^2 -81*(x(2)+ 0.1)^2 +sin(x(3)) +10.6;
y(3,1) = exp(-x(1)*x(2)) + 20*x(3) + (10*pi -3)/3;
% FILE  f520.m  ends.
```

Yhtälöryhmän `f520(x)=0` ratkaisu alkuarvolla `x0=[3 4 5]` sujuu seuraavasti:

(1) `x0=3:5;`

Toistetaan (ylänuolinäppäintä painaen) riviä

(2) `x1=x0'-numjaco('f520',3,x0,3)\f520(x0); x0=x1',f520(x0)'`

seitsemän kertaa, jolloin saadaan

$$\texttt{x0=[0.4970 0.2575 -0.5176]}\,,$$

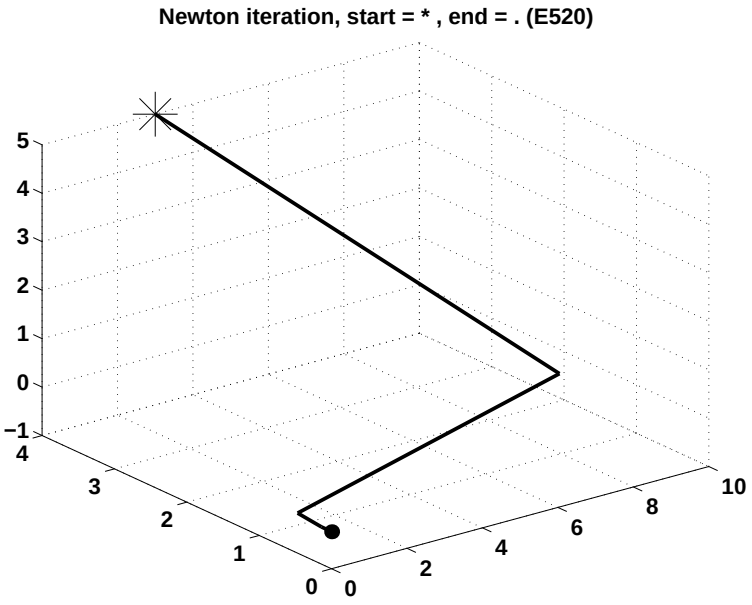
ja funktion arvo `x0`:ssa on luokkaa 10^{-11}.

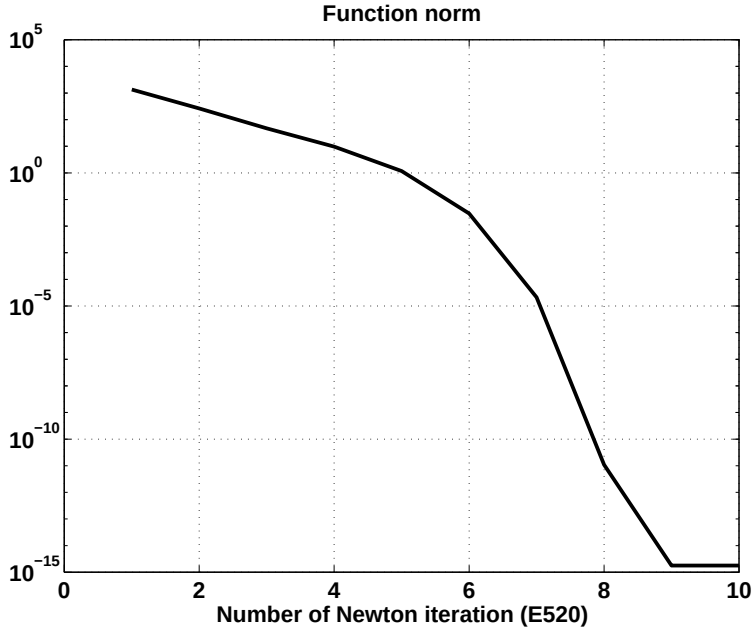
Johtopäätös: Kun numjaco.m ja f520.m on luotu, voidaan epälineaarinen yhtälöryhmä `f520(x)=0` ratkaista kirjoittamalla rivi (1) ja toistamalla riviä (2).

Esitämme vielä m-tiedostoon perustuvan ratkaisun.

```
% FILE  e520.m  begins.
% USES numjaco.m, f520.m
x0 = 3:5; hold off;
data =[0, x0, norm(f520(x0)')];
for j =1:9
   x1= x0' -numjaco('f520', 3,x0,3)\f520(x0); x0=x1';
   data =[data; [j  x1' norm(f520(x1'))  ]];
end;
x = data(:,2); y = data(:,3); z = data(:,4);
x1=  data(10,2); y1=  data(10,3); z1=  data(10,4);
clf;
plt3=plot3(x,y,z);
txt=['Newton iteration, start = * , end = . (E520)'];
title(txt,'FontSize',[15],'FontWeight','bold');
hold on;
plot3(x1,y1,z1,'k.', 3,4,5,'k*','MarkerSize',30), grid on;
set(plt3,'LineWidth', 2.5);
pause;
```

```
print -deps fig520a;
hold off;
clf;
splt=semilogy(1:10,data(:,5)), grid on;
txt=['Function norm'];
title(txt,'FontSize',[15],'FontWeight','bold');
xlabel('Number of Newton iteration (E520)','FontSize',[15],...
       'FontWeight','bold');
set(splt,'LineWidth', 2.5);
print -deps fig520b;
delete e520.dat;
diary e520.dat;
disp(' Newton iteration: ');
disp('  i      x         y          z         norm(f)  ');
fprintf(1,' %2d  %8.5f  %8.5f   %8.5f   %12.5e \n',data');
diary off;
% FILE  e520.m  ends.
```





Ohjelman `e520` tuottamat kuvat yllä ja sen tuottama datatiedosto alla.

```
Newton iteration:
 i       x          y          z         norm(f)
 0    3.00000    4.00000    5.00000    1.34745e+03
 1    9.95003    2.03851   -0.47359    2.62865e+02
 2    0.52454    0.74289   -0.47360    4.71338e+01
 3    0.49491    0.39726   -0.51435    9.67536e+00
 4    0.49717    0.27715   -0.51708    1.16853e+00
 5    0.49705    0.25801   -0.51758    2.96571e-02
 6    0.49704    0.25750   -0.51759    2.12010e-05
 7    0.49704    0.25750   -0.51759    1.08642e-11
 8    0.49704    0.25750   -0.51759    1.77636e-15
 9    0.49704    0.25750   -0.51759    1.77636e-15
```

6 Ortogonaaliset funktiot

Tarkoituksemme on esitellä funktioavaruuksien lineaarialgebraa. Euklidisen avaruuden $\mathbb{R}^n$ lineaarialgebrasta tutut käsitteet kohtisuoruus, projektio ja kanta yleistetään nyt tapaukseen, missä $\mathbb{R}^n$:n vektoreiden paikalla on reaaliakselin suljetulla välillä määritellyt jatkuvat reaali- tai kompleksiarvoiset funktiot. Rajoitumme funktioavaruuksien lineaarialgebran perusteoriaan, jota jatkossa sovelletaan mm. Fourier-sarjoihin. Pidemmälle menevässä kirjallisuudessa on tapana käyttää perustyövälineenä Lebesguen integraalia (1875–1941), mutta meille riittää matematiikan approbaturista tuttu Riemannin integraali (1826–1866).

6.1. Funktioavaruuden $C[0,1]$ laskutoimitukset. Merkitsemme $C[0,1] = \{\, f\colon [0,1] \to \mathbb{R} \;:\; f \text{ jatkuva} \,\}$. Kun $\lambda \in \mathbb{R}$ ja $f, g \in C[0,1]$, määrittelemme funktiot λf, $f+g$ ja fg asettamalla

(1) $(\lambda f)(x) = \lambda f(x) \quad \forall x$;

(2) $(f+g)(x) = f(x) + g(x) \quad \forall x$;

(3) $(f \cdot g)(x) = f(x)g(x) \quad \forall x$.

Jatkuvien funktioiden perusominaisuuksien nojalla näin määritellyt funktiot λf, $f+g$ ja $f \cdot g$ ovat myös jatkuvia, joten kukin määritellyistä operaatioista tuottaa uuden $C[0,1]$:n alkion eli on $C[0,1]$:n laskutoimitus. Vastaavasti määritellään avaruus $C[a,b]$, kun $a, b \in \mathbb{R}$ ja $a < b$. Jos taas $k = 1, 2, \ldots$, niin avaruus $C^k[a,b]$ on niiden välillä $[a,b]$ jatkuvien funktioiden avaruus, joiden kertalukuja p, missä $p \leq k$, olevat derivaatat ovat jatkuvia avoimella välillä (a,b).

6.2. Lineaarisen vektoriavaruuden määritelmä. Olkoon V joukko alkioita ja K skalaarien joukko (käytännössä joko $\mathbb{R}$ tai $\mathbb{C}$), ja olkoon määritelty operaatiot $+$ ja $\cdot$ siten, että $f, g \in V \Rightarrow f+g \in V$ ja $\alpha \in K, f \in V \Rightarrow \alpha f \in V$. Sanotaan, että V on *lineaarinen vektoriavaruus,* jos seuraavat kolme ominaisuutta ovat voimassa:

(1) $(V, +)$ on Abelin ryhmä;

(2) $\alpha(f+g) = \alpha f + \alpha g$ ja $(\alpha+\beta)f = \alpha f + \beta f \quad \forall \alpha, \beta \in K$ ja $f, g \in V$;

(3) $(\alpha\beta)f = \alpha(\beta f)$, $1 \cdot f = f \quad \forall \alpha, \beta \in K$ ja $f \in V$.

Kirjallisuudessa käytettävät nimitykset vaihtelevat jossain määrin, ja tällaista avaruutta kutsutaan myös lineaariseksi avaruudeksi tai vektoriavaruudeksi.

6.3. Esimerkkejä. (1) $C[0,1]$ on lineaarinen vektoriavaruus.

(2) Differentiaaliyhtälön $y'' + a_1 y' + a_2 y = 0$ ratkaisujen joukko on lineaarinen vektoriavaruus.

(3) Rajoitettujen lukujonojen joukko

$$\{\, (a_n) : \ \exists\ M \quad \text{s.e.} \quad |a_n| < M \quad \forall\ n = 1, 2, \dots \}$$

on lineaarinen vektoriavaruus, kun asetetaan $(a_n) + (b_n) = (a_n + b_n)$, $\lambda(a_n) = (\lambda a_n)$.

6.4. Määritelmä. Lineaarisen vektoriavaruuden V joukko $\{x_1, \dots, x_n\}$ on *vapaa*, jos ehdosta $\sum_{i=1}^{n} c_i x_i = 0$ seuraa $c_i = 0$ jokaisella i; muuten se on *sidottu*. Vapaa joukko $\{x_1, \dots, x_n\}$ on V:n *kanta*, jos jokainen $y \in V$ voidaan esittää muodossa $y = \sum_{i=1}^{n} c_i x_i$.

6.5. Määritelmä. Jos f ja g ovat välillä $[0,1]$ Riemann-integroituvia, sanotaan integraalia

$$\text{(6.6)} \qquad \int_0^1 f(x)g(x)\,dx \qquad \text{(kompleksiarvoisille: } \int_0^1 f(x)\overline{g(x)}\,dx\text{)}$$

niiden *sisätuloksi* ja sitä merkitään (f, g) (tai $(f|g)$, $\langle f, g\rangle$, $f \cdot g$). Ei-negatiivista lukua $(f, f)^{1/2}$ merkitään $\|f\|$ ja sanotaan f:n *normiksi*. Lineaarista vektoriavaruutta, jossa on määritelty sisätulo, sanotaan *sisätuloavaruudeksi*.

Koska integrointi on lineaarinen operaatio, nähdään oikeiksi seuraavat sisätulon ominaisuudet reaaliarvoisille funktioille:

$$\text{(6.7)} \quad \begin{cases} (f, g) = (g, f), \quad (cf, g) = (f, cg) = c(f, g) \\ (f_1 + f_2, g) = (f_1, g) + (f_2, g), \quad (f, g_1 + g_2) = (f, g_1) + (f, g_2) \end{cases} .$$

Kompleksiarvoisille funktioille pätevät vastaavat ominaisuudet; tosin kaavoissa tällöin esiintyy kompleksikonjugaatteja (esim. $(cf, g) = c(f, g) = (f, \overline{c}g)$).

6.8. Normiepäyhtälöitä. Sisätuloavaruuden V funktioille f ja g pätevät seuraavat epäyhtälöt:

(6.9) Cauchyn–Schwarzin epäyhtälö (1787–1857 ja 1843–1921):

$$|(f,g)| \le \|f\|\,\|g\|\,,$$

(6.10) Minkowskin epäyhtälö (1864–1909):

$$\|f+g\| \le \|f\| + \|g\|\,.$$

Molemmat epäyhtälöt ovat tuttuja $\mathbb{R}^n$:n tapauksesta. Minkowskin epäyhtälö vastaa kolmioepäyhtälöä vektoreille. Epäyhtälön vasen puoli voidaan kirjoittaa sisätulon lineaarisuuden nojalla

$$\|f+g\|^2 = (f+g, f+g) = (f,f) + 2(f,g) + (g,g) = \|f\|^2 + \|g\|^2 + 2(f,g)\,,$$

joten $\|f+g\|^2 = \|f\|^2 + \|g\|^2$ joss $(f,g) = 0$. Analogisesti Pythagoraan lauseen kanssa sanotaan, että f ja g ovat *kohtisuorassa* l. *ortogonaaliset* jos $(f,g) = 0$. Tällöin käytämme merkintää $f \perp g$.

6.11. Määritelmä. Välillä $[a,b]$ määriteltyjen funktioiden joukko $S = \{\phi_0, \phi_1, \phi_2, \ldots\}$ on *ortogonaalinen* (l. *kohtisuora*), jos

$$(\phi_n, \phi_m) = 0 \quad \forall\ m \ne n\,.$$

Jos lisäksi $\|\phi_n\| = 1$ kaikille $n = 1, 2, \ldots$, sanotaan että S on *ortonormaali*.

Jos S on ortogonaalinen ja $\|\phi\| \ne 0$ jokaiselle $\phi \in S$, niin $\{\phi_0/\|\phi_0\|, \phi_1/\|\phi_1\|\}$ on ortonormaali.

6.12. Esimerkki. Onko funktiojoukko $S = \{\phi_0, \phi_1, \ldots\}$, missä

$$\phi_0(x) = \frac{1}{\sqrt{2\pi}}, \quad \phi_{2n-1}(x) = \frac{\cos nx}{\sqrt{\pi}}, \quad \phi_{2n}(x) = \frac{\sin nx}{\sqrt{\pi}} \qquad (n = 1, 2, \ldots),$$

ortonormaali välillä $[0, 2\pi]$?

Nyt $2\sin nx \cos mx = \cos(nx - (\frac{\pi}{2} - mx)) - \cos(nx + \frac{\pi}{2} - mx)$ ja

$$\int_0^{2\pi} \cos(px + c)\,dx = 0 \quad \forall\ p = \pm 1, \pm 2, \ldots \quad \forall\ c \in \mathbb{R}\,,$$

$$\int_0^{2\pi} \cos(0 \cdot x \pm \frac{\pi}{2})\,dx = 2\pi \cos(\pm\frac{\pi}{2}) = 0\,.$$

Siis $(\phi_{2n-1}, \phi_{2m})\pi = \int_0^{2\pi} \cos nx \sin mx\,dx$

$$= 0\,,\ \text{jos } m \ne n;$$
$$= 0\,,\ \text{jos } m = n.$$

Lisäksi $(\phi_{2m-1}, \phi_{2n-1})\pi = \int_0^{2\pi} \cos(mx)\cos(nx)\,dx = 0$, jos $m \neq n$, ja

$$(\phi_{2m-1}, \phi_{2m-1})\pi = \int_0^{2\pi} \cos^2 mx\,dx = \int_0^{2\pi} \frac{1+\cos 2mx}{2}\,dx = \pi\,,$$

$$(\phi_{2m}, \phi_{2m})\pi = \int_0^{2\pi} \sin^2 mx\,dx = \pi\,,$$

$$(\phi_{2m}, \phi_{2n})\pi = \int_0^{2\pi} \sin nx \sin mx\,dx = 0\,,\ \text{jos } m \neq n\,.$$

Siis S on ortonormaali.

6.13. Määritelmä. Olkoon $f, \phi\colon [a,b] \to \mathbb{R}$ Riemann-integroituvia ja $\|\phi\| = 1$. Silloin f:n *projektio* ϕ:n suuntaan on

$$\operatorname{proj}(f : \phi) = (f, \phi)\,\phi\,.$$

6.14. Lause. $\operatorname{proj}(f : \phi) = f \quad \Leftrightarrow \quad \exists c \in \mathbb{R} : f = c\phi\,.$

Todistus. $\Rightarrow$: $\operatorname{proj}(f : \phi) = f \ \Rightarrow\ f = (f, \phi)\phi = c\phi\,.$ □
$\Leftarrow$: $f = c\phi \ \Rightarrow\ \operatorname{proj}(f : \phi) = (c\phi, \phi)\phi = c(\phi, \phi)\phi = c\phi\,.$ □

6.15. Lause. $f - \operatorname{proj}(f : \phi) \perp \phi\,.$

Todistus. $(f - (f, \phi)\phi, \phi) = (f, \phi) - (f, \phi)\overbrace{(\phi, \phi)}^{=1} = 0\,.$ □

6.16. Määritelmä. Olkoon $S_n = \{\phi_1, \phi_2, \ldots, \phi_n\}$ ortonormaali systeemi ja $f\colon [a,b] \to \mathbb{R}$ integroituva. Silloin f:n *projektio* S_n:n *virittämälle aliavaruudelle* V_n määritellään kaavalla

$$\operatorname{proj}(f : \phi_1, \ldots, \phi_n) = \sum\nolimits_{j=1}^{n} (f, \phi_j)\phi_j\,.$$

Lauseet 6.14 ja 6.15 yleistyvät helposti myös 6.16:n mielessä otetuille projektioille. Lauseiden 6.17 ja 6.18 todistukset jätetään harjoitustehtäviksi.

6.17. Lause. $\operatorname{proj}(f : \phi_1, \ldots, \phi_n) = f \quad \Leftrightarrow \quad (f \in V_n)\,.$

6.18. Lause. $f - \operatorname{proj}(f : \phi_1, \ldots, \phi_n) \perp g \quad \forall g \in V_n\,.$

6.19. Esimerkki. Olkoon $f\colon [0, 2\pi] \to \mathbb{R}$ integroituva, n kokonaisluku ja $\phi_1(x) = \frac{\sin nx}{\sqrt{\pi}}$, $\phi_2(x) = \frac{\cos nx}{\sqrt{\pi}}$. Johda lauseke $\operatorname{proj}(f : \phi_1, \phi_2)$:lle.

Nyt 6.12:sta seuraa, että $\{\phi_1, \phi_2\}$ on ortonormaali, jolloin 6.16:n mukaan

$$\operatorname{proj}(f : \phi_1, \phi_2) = \left(\int_0^{2\pi} f(t)\frac{\sin nt}{\sqrt{\pi}}\,dt\right)\frac{\sin nx}{\sqrt{\pi}} + \left(\int_0^{2\pi} f(t)\frac{\cos nt}{\sqrt{\pi}}\,dt\right)\frac{\cos nx}{\sqrt{\pi}}$$

$$= \frac{1}{\pi}\int_0^{2\pi} f(t)(\sin nt \sin nx + \cos nt \cos nx)\,dt = \frac{1}{\pi}\int_0^{2\pi} f(t)\cos n(x-t)\,dt\,.$$

6.20. Huomautus. Määritelmästä 6.4 seuraa välittömästi, että lineaarisen vektoriavaruuden V joukko $\{f_1, \ldots, f_n\}$ on sidottu (l. lineaarisesti riippuva) joss jokin alkioista f_j on muiden lineaarikombinaatio, $f_j = \sum_{i=1, i\neq j}^{n} a_i f_i$.

6.21. Esimerkki. Onko $\{f_1, f_2, f_3\}$ vapaa, kun $f_1(x) = 1$, $f_2(x) = \cos^2 x$, $f_3(x) = \cos 2x$?

Nyt

$$f_2(x) = \cos^2 x = \frac{1}{2}(1 + \cos 2x) = \frac{1}{2}(f_1(x) + f_3(x))\,,$$

joten $\{f_1, f_2, f_3\}$ on sidottu 6.20:n perusteella.

6.22. Lause. Jokainen ortogonaalinen jono on vapaa.

Todistus. Olkoon $\phi_1, \phi_2, \ldots, \phi_n$ ortogonaalinen. Pitää osoittaa, että ehdosta $\sum_{k=1}^{n} \alpha_k \phi_k = 0$ seuraa $\alpha_k = 0$. Mutta kaikille j pätee:

$$\sum_{k=1}^{n} \alpha_k \phi_k = 0 \quad \Rightarrow \quad \Big(\sum_{k=1}^{n} \alpha_k \phi_k\,,\, \phi_j\Big) = (0, \phi_j) = 0$$

$$\Rightarrow \quad \sum_{k=1}^{n} \alpha_k (\phi_k, \phi_j) = 0 \quad \overset{\phi_1,\ldots,\phi_n \text{ ortog.}}{\Longrightarrow} \quad \alpha_j(\phi_j, \phi_j) = 0 \;\Rightarrow\; \alpha_j = 0\,. \quad \square$$

6.23. Gramin–Schmidtin ortogonalisointi. Selventäen lausetta 6.22 toteamme, että jokainen vapaa jono ei ole ortogonaalinen. Kuitenkin jokaisesta vapaasta jonosta voidaan konstruoida Gramin–Schmidtin (1850–1916, 1876–1959) menettelyllä ortogonaalinen jono siten, että uuden jonon alkiot numero $1, \ldots, k$ virittävät saman aliavaruuden kuin vanhan jonon alkiot numero $1, \ldots, k$. Jos joukko $\{x_n\}$ on vapaa, niin uusi joukko $\{y_n\}$ määritellään asettamalla $y_1 = x_1$ ja

$$y_n = x_n - \sum_{k=1}^{n-1}((x_n, y_k)/(y_k, y_k))y_k \qquad (n = 2, 3, \ldots)\,. \tag{6.24}$$

6.25. Lause. Jos $\{\phi_1, \ldots, \phi_n\}$ on ortonormaali kanta lineaarisen avaruuden V n-ulotteiselle aliavaruudelle H_n ja jos $f \in V$, niin on olemassa $h \in H_n$ s.e. $\|f - h\|^2$ saa pienimmän arvon ja $h = \sum_{k=1}^{n} \alpha_k\phi_k$, kun $\alpha_k = (f, \phi_k)$. Toisin sanoen, on $h = \mathrm{proj}(f : \phi_1, \ldots, \phi_n)$. Ko. pienin arvo on $\|f\|^2 - \sum\limits_{k=1}^{n} |\alpha_k|^2$.

Todistus. Olkoon $g \in H_n$. Silloin $g = \sum \beta_k\phi_k$ ja

$$\begin{aligned}
\|f - g\|^2 &= (f - g, f - g) = (f, f) - (g, f) - (f, g) + (g, g) \\
&= \|f\|^2 - \sum_{k=1}^{n} \beta_k(\phi_k, f) - \sum_{k=1}^{n} \overline{\beta}_k(f, \phi_k) + \sum_{k=1}^{n} \beta_k\overline{\beta}_k \\
&= \|f\|^2 - \sum \beta_k\overline{\alpha}_k - \sum \overline{\beta}_k\alpha_k + \sum |\beta_k|^2 \\
&= \|f\|^2 + \sum(\beta_k - \alpha_k)(\overline{\beta_k} - \overline{\alpha}_k) - \sum |\alpha_k|^2 \\
&= \|f\|^2 - \sum_{k=1}^{n} |\alpha_k|^2 + \sum_{k=1}^{n} |\beta_k - \alpha_k|^2 \geq \|f\|^2 - \sum_{k=1}^{n} |\alpha_k|^2\,. \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \uparrow \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad = \text{ kun } \beta_k = \alpha_k \quad \forall\ k\,.
\end{aligned}$$

Siis kun $\beta_k = \alpha_k$, saa $\|f - g\|$ pienimmän arvonsa ja tällöin $g = h = \sum \alpha_k\phi_k$. □

6.26. Lause. Olkoon $\{\phi_1, \phi_2, \ldots\}$ ortonormaali välillä $[a, b]$ ja f integroituva. Silloin jos $c_k = (f, \phi_k)$,

(1) $\sum\limits_{k=1}^{\infty} |c_k|^2 \leq \|f\|^2$ (Besselin epäyhtälö, 1784–1846);

(2) $\sum_{k=1}^{\infty} |c_k|^2 = \|f\|^2 \Leftrightarrow \lim_{n\to\infty} \|f - \sum_{k=1}^{n} c_k\phi_k\| = 0$ (Parsevalin kaava, 1755–1836).

Todistus. (1) Kaavan 6.25 nojalla saadaan

$$0 \le \Big\| f - \sum_{k=1}^{n} c_k\phi_k \Big\|^2 = \|f\|^2 - \sum_{k=1}^{n} |c_k|^2 \quad \forall\, n\,,$$

joten

$$\sum_{k=1}^{n} |c_k|^2 \le \|f\|^2 \quad \forall\, n\,,$$

mistä kohdan (1) väite seuraa. □

(2) Käytä samaa kaavaa kuin (1):ssä. □

6.27. Hilbertin avaruuden määritelmä. Hilbertin avaruus (1862–1943) ℓ^2 on sellaisten skalaarijonojen $(\alpha_1, \alpha_2, \ldots)$ joukko, joille $\sum_{k=1}^{\infty} |\alpha_k|^2 < \infty$. Tämä varustetaan sisätuloavaruuden operaatioilla seuraavasti:

$$(\alpha_1, \alpha_2, \ldots) + (\beta_1, \beta_2, \ldots) = (\alpha_1 + \beta_1, \alpha_2 + \beta_2, \ldots),$$

$$k(\alpha_1, \alpha_2, \ldots) = (k\alpha_1, k\alpha_2, \ldots),$$

$$(\alpha, \beta) = \sum_{k=1}^{\infty} \alpha_k \overline{\beta}_k, \quad \|\alpha\| = (\alpha, \alpha)^{1/2}.$$

Hilbertin avaruus ℓ^2 esiintyy usein approksimaatioteoriassa.

Hilbertin avaruus ℓ^2 on kuitenkin erikoistapaus yleisemmästä käsitteestä, joka voidaan määritellä näin: Hilbertin avaruus on Banachin avaruus, jonka normi on sisätulon määrittelemä. Näin myös funktiot $f : [0, 1] \to \mathbb{R}$, joille

$$\int_0^1 |f|^2\, dx < \infty\,,$$

muodostavat, kun integroituvuus sopivasti tulkitaan ja sisätulo määritellään kaavalla

$$(f, g) = \int_0^1 f(x)\overline{g(x)}\, dx\,,$$

Hilbert-avaruuden, jota merkitään $L^2(0, 1)$. (Hilbertin avaruuksien teoriassa käytetään tavallisesti Lebesguen integraalia.)

6.28. Ortogonaaliset polynomit. Potenssit $P = \{1, x, x^2, \ldots\}$ ovat lineaarisesti riippumattomia $C[a, b]$:ssä, sillä ehdosta

$$a_0 + a_1 x + \cdots + a_n x^n = 0 \quad \forall\ x \in [a, b]$$

seuraa, että $a_i = 0$ kaikille $i = 0, 1, \ldots, n$. Jos w on $[a, b]$:llä integroituva funktio, kaava

$$(f, g) = \int_a^b w(x) f(x) g(x)\, dx$$

määrittelee sisätulon mikäli $w(x) > 0$ kaikille x. Näin ollen P voidaan ortonormeerata tämän sisätulon suhteen, ja saamme tulokseksi joukon $P^* = \{p_0^*(x), p_1^*(x) \ldots\}$, jolle

$$\int_a^b w(x) p_m^*(x) p_n^*(x)\, dx = \delta_{mn}, \quad \text{kun } m, n = 0, 1, 2, \ldots\ ,$$

missä δ_{mn} on Kroneckerin (1823–1891) delta: $\delta_{mn} = 0$ jos $m \neq n$ ja $\delta_{mm} = 1$ kaikille $m = 0, 1, 2, \ldots$. Painofunktion w on oltava sellainen, että integraalit

$$\int_a^b w(x) x^n\, dx$$

ovat olemassa, kun $n = 0, 1, 2, \ldots$. (Huomaa myös tapaus $|a|$ tai $|b| = \infty$.)

Erikoistapauksia:

(1) $a = -1$, $b = 1$, $w(x) = 1$: Legendren polynomit (1752–1833);

(2) $a = -1$, $b = 1$, $w(x) = 1/\sqrt{1 - x^2}$: Tšebyševin polynomit, 1. laji (1821–1894);

(3) $a = -1$, $b = 1$, $w(x) = \sqrt{1 - x^2}$: Tšebyševin polynomit, 2. laji;

(4) $a = -1$, $b = 1$, $w(x) = (1 + x)^\alpha (1 - x)^\beta$, $\alpha, \beta > -1$: Jacobin polynomit (1804–1851);

(5) $a = 0$, $b = \infty$, $w(x) = x^\alpha e^{-x}$, $\alpha > -1$: Laguerren polynomit (1834–1886);

(6) $a = -\infty$, $b = \infty$, $w(x) = e^{-x^2}$: Hermiten polynomit (1822–1901).

Ks. [AS, s. 773 ja s. 332].

6.29. Huomautus. Jos $f \in L^2(a,b)$ (ks. 6.27), lauseesta 6.25 seuraa, että lausekkeella

$$I(a_0,\ldots,a_n) = \int_a^b \Big(f(x) - \sum_{i=0}^{n} a_i x^i\Big)^2\, dx$$

on yksikäsitteinen minimi. Minimi voidaan löytää kahdella eri tavalla:

(1) Muodostetaan $\{x^0, x, x^2, \ldots, x^n\}$:n virittämälle aliavaruudelle Gramin–Schmidtin menettelyllä 6.23 ortonormaali kanta, johon sovelletaan lausetta 6.25.

(2) Sovelletaan lauseen 6.25 yksikäsitteisyyspuolta ja etsitään lausekkeen $I(a_0,\ldots,a_n)$ minimiä vaatimalla $\frac{\partial I}{\partial a_i} = 0$ (normaaliyhtälöt).

Sovellamme nyt tapaa (2) ja etsimme siis sellaiset luvut $c_1,\ldots,c_n$, jotka minimoivat funktion

$$f(c_1,\ldots,c_n) = \int_{r_1}^{r_2} \Big(g(x) - \sum_{k=1}^{n} c_k x^{n-k}\Big)^2\, dx\,,$$

kun $g(x) = \exp(x)$ ja $n = 2,3,4$. Normaaliyhtälöt $\partial f/\partial c_j = 0 \quad (j = 1,\ldots,n)$ johtavat ehtoihin

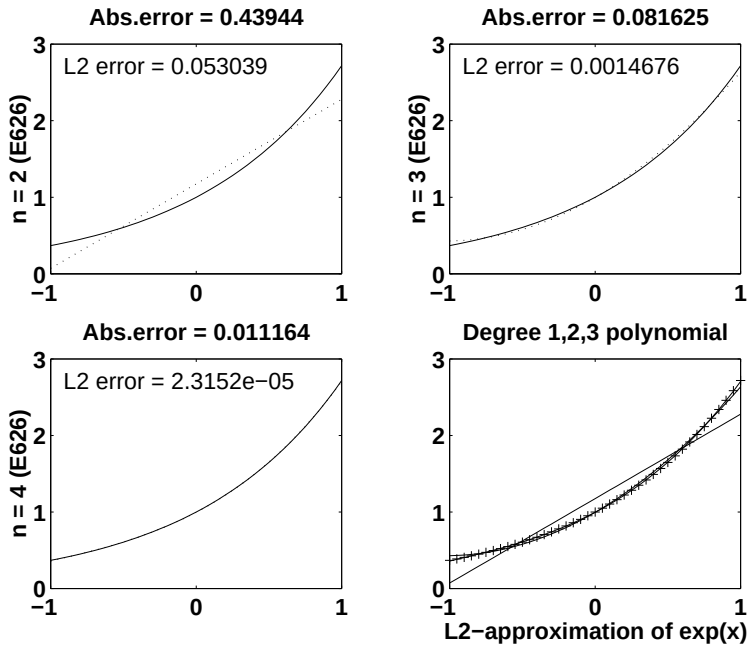
$$\sum_{k=1}^{n} c_k \int_{r_1}^{r_2} \frac{r^{2n-k-j+1}}{2n-k-j+1} = \int_{r_1}^{r_2} g(x) x^{n-j}\, dx\,.$$

Oikean puolen integrointiin voidaan käyttää esim. `quadia`.

```
% FILE  e626.m  begins.
% Let f(c1,...,cn) =
%    int_{r1}^{ r2}{ [exp(x) - sum_{k= 1}^{n}(c_k x^{n-k}) ]^2 dx}
% To minimize f find c1,...,cn such that df/dcj = 0,
% or equivalently -2 int_{r1}^{ r2}{exp(x) x^{n-j} dx} +
%      2 int_{r1}^{ r2}{sum_{k= 1}^{ n} (c_k x^{n-k}) x^{n-j} dx }
%    = -2 int_{r1}^{r2}{exp(x) x^{n-j} dx} +
%      2 sum_{k= 1}^n ((c_k /(2*(n)-k-j+1))((r2)^{2*(n)-k-j+1}-
%      (r1)^{2*(n)-k-j+1} ) )} = 0
hold off, clf, clear;  nmax =4;
data =zeros(nmax,nmax);
r1 = input('Enter r1 : ');
r2 = input('Enter r2 > r1 : ');
for n = 2:nmax;
for j = 1:n
```

```
    for k =1:n
        a(j,k) = ((r2).^(2*(n)-k-j+1) -(r1).^(2*(n)-k-j+1))/...
                 (2*(n)-k-j+1);
    end;
% The coefficients of c_j are entries of a_{i,j}
xx=r1:0.001:r2;
yy=exp(xx).*(xx.^(n-j));
% Use trapezoidal rule to compute b_j :
b(j,1)=0.001*(sum(yy)-0.5*(yy(1)+yy(length(yy))));
end;
disp(['a, b==']
disp([a b])
c = a\b;
data(n,1:n) =c';
disp(['c==']
disp(c');
h=  (r2-r1)/40;
x = r1:h:r2;
y1 = exp(x);
y2 = polyval(c,x);
erhe = abs(y2-y1);
le = length(erhe);
maxer = max(erhe);
% Use trapezoidal rule to compute error:
int1 = h*(sum(erhe)-0.5*(erhe(1)+erhe(le)));
int2 = h*(sum(erhe.^2)-0.5*(erhe(1)^2+erhe(le)^2));
txt =['Abs.error = ' num2str(maxer) ];
txt2 =['  L2 error = ' num2str(int2) ];
subplot(2,2,n-1);
plot(x,y1,x,y2,'k:'), title(txt),
text(min(x),max(y1),txt2),
ylabel(['n = ' num2str(n) ' (E626)']);
end;
x = r1: h :r2;
for j =2:nmax
   y(j,:) = polyval(data(j,1:j),x);
end;
subplot(2,2,4);
for j =2:nmax
   plot(x,y(j,:)), pause;
   hold on;
```

```
end;
subplot(2,2,4);
title('Degree 1,2,3 polynomial');
plot(x,exp(x),'k+'),
xlabel(' L2-approximation of exp(x) '),pause; hold off;
% FILE  e626.m  ends.
```



Tästä esimerkistä lähtien pyrimme noudattamaan käytäntöä, että MATLABin käynnistyshakemistossa olevan tiedoston `startup.m` sisältämät komennot säätävät automaattisesti (MATLABin käynnistyessä) kuvaotsikoiden fonttikoot yms. sopiviksi, jolloin niihin ei tarvitse puuttua joka esimerkkitiedostossa.

```
% FILE  startup.m  begins.
set(0,'DefaultAxesFontWeight','bold')
set(0,'DefaultAxesFontSize',[15])
set(0,'DefaultTextFontSize',[15])
% FILE  startup.m  ends.
```

6.30. Reuna-arvotehtäviä (RAT). Ortogonaaliset funktiot esiintyvät eräiden tavallisten differentiaaliyhtälöiden teoriaan kuuluvien tehtävien ratkaisussa, joita nyt tarkastelemme esimerkein.

(1) Millä λ:n arvoilla DY:llä $(*)$ $y'' + \lambda y = 0$ on ei-triviaaleja RAT:n $y(0) = 0$, $y(\pi) = 0$ toteuttavia ratkaisuja? Etsi ne.

Ratkaisu. $(*)$:n yleinen ratkaisu on

$$y(x) = c_1 \sin(\sqrt{\lambda}x) + c_2 \cos(\sqrt{\lambda}x)\,.$$

Nyt $y(0) = 0 \Rightarrow c_2 = 0$ ja $y(\pi) = 0 \Rightarrow \sqrt{\lambda}\pi = n \cdot \pi$. Nähdään että kysytyt arvot ovat $\lambda = n^2$ $(n = 1, 2, 3, \ldots)$ eli $\lambda = 1, 4, 9, \ldots$ ja että vastaavat ratkaisut ovat

$$\sin x,\ \sin 2x,\ \sin 3x,\ \ldots$$

vakiolla kerrottuina.

(2) Millä λ:n arvoilla kohdan (1) yhtälöllä $(*)$ on RAT:n $y'(0) = 0$, $y'(\pi) = 0$ toteuttavia ratkaisuja? Etsi ne.

Ratkaisu. Ratkaisut ovat

$$1,\ \cos x,\ \cos 2x,\ \cos 3x,\ \ldots$$

vakiolla kerrottuna. Nämä vastaavat λ:n arvoja $0, 1, 4, 9, \ldots$.

(3) Millä λ:n arvolla kohdan (1) yhtälöllä $(*)$ on RAT:n $y(0) = y(2\pi)$, $y'(0) = y'(2\pi)$ toteuttavia ratkaisuja? Etsi ne.

Ratkaisu. Kelvolliset λ:n arvot ovat samat kuin (2):ssa, ja arvoa $\lambda = n^2$ vastaavina ratkaisuina ovat $\sin nx$:n ja $\cos nx$:n lineaarikombinaatiot.

Kohdan (1) funktiojoukko $\{\sin nx\}$ on ortogonaalinen välillä $(0, \pi)$; vastaavasti kohdassa (2). Kohdan (3) jono

$$1, \sin x, \cos x, \sin 2x, \cos 2x, \ldots$$

on ortogonaalinen välillä $(0, 2\pi)$. Yo. esimerkit (1)–(3) ovat esimerkkejä yleisestä *Sturmin–Liouvillen* yhtälöstä (1803–1855, 1809–1882)

$$\frac{d}{dx}\Big(p(x)\frac{dy}{dx}\Big) + (q(x) + \lambda r(x))y = 0 \qquad (\lambda \text{ vakio}), \tag{6.31}$$

missä p, q, r ovat jatkuvia välillä $[a, b]$, $p(x), r(x) > 0$ kaikille x ja p' on jatkuva.

Sturmin–Liouvillen tehtävä on (6.31):n ratkaiseminen pian määriteltävin reuna-arvoin. Vastaavia funktioita y kutsutaan *ominaisfunktioiksi* ja vastaavia λ:n arvoja *ominaisarvoiksi*. Merkitään

$$Ly = \frac{d}{dx}\Big(p(x)\frac{dy}{dx}\Big) + q(x)y\,, \tag{6.32}$$

jolloin L vie $C^2(a, b)$:n funktion $C(a, b)$:han. Reuna-arvot halutaan määrätä niin, että ratkaisujoukko on $C^2(a, b)$:n lineaarinen aliavaruus, jota merkitään $BC^2(a, b)$. Merkintä on vähän huono, sillä siitä ei ilmene, että kutakin reuna-arvotehtävää

voi vastata mahdollisesti eri avaruus $BC^2(a,b)$. Toiseksi oletamme, että L on $BC^2(a,b)$:ssä *itseadjungoitu,* ts. että

$$(Lf,g) = (f,Lg) \qquad \forall\ f,g \in BC^2(a,b). \tag{6.33}$$

Nyt on aika määrätä Sturmin–Liouvillen tehtävään liittyvät reuna-arvot, ja teemme tämän vaatimalla, että reuna-arvot tekevät erotuksen $(Lf,g) - (f,Lg)$ nollaksi. Saamme

$$\begin{aligned}(Lf,g) - (f,Lg) &= \int_a^b Lf(x)\overline{g(x)}\,dx - \int_a^b f(x)\overline{Lg(x)}\,dx \\ &= \int_a^b [(pf')' + qf]\,\overline{g}\,dx - \int_a^b [(p\overline{g}')' + q\overline{g}]\,f\,dx \\ &= \int_a^b [pf''\overline{g} + p'f'\overline{g} - p'\overline{g}'f - p\overline{g}''f]\,dx = \Big/_a^b\, p\,[f'\overline{g} - \overline{g}'f]\,,\end{aligned} \tag{6.34}$$

joten Sturmin–Liouvillen tehtävä on ratkaista (6.31) reunaehdoin

$$\begin{aligned}(Lf,g) - (f,Lg) &= p(b)\,[f'(b)\overline{g(b)} - \overline{g'(b)}f(b)] \\ &\quad -p(a)\,[f'(a)\overline{g(a)} - \overline{g'(a)}f(a)] = 0\,.\end{aligned} \tag{6.35}$$

Huomautus. (6.35) toteutuu, jos funktiot f ja g molemmat toteuttavat välin $[a,b]$ molemmissa päätepisteissä jonkin seuraavista ehdoista:

(4) $y(x) = 0 \quad (x = a,b)\,;$

(5) $y'(x) = 0 \quad (x = a,b)\,;$

(6) $y(x) + \alpha y'(x) = 0 \quad (x = a,b) \quad (\alpha \text{ vakio}).$

Lisäksi (6.35) toteutuu, jos $p(a) = p(b)$ ja sekä f että g toteuttavat ehdot

$$y(a) = y(b) \quad \text{ja} \quad y'(a) = y'(b)\,. \tag{6.36}$$

Ehtojen (4)–(6) nojalla nähdään, että merkintä $BC^2(a,b)$ voi liittyä hyvinkin erilaisiin reunaehtoihin.

6.37. Lemma. Itseadjungoidun operaattorin L (ks. (6.32)) ominaisarvot ovat reaalisia.

Todistus. Oletetaan, että λ_0 on ominaisarvo, jolloin on olemassa $y \not\equiv 0$ s.e. $Ly + \lambda_0 ry = 0$. Kun L on itseadjungoitu, on $(Ly, y) = (y, Ly)$ eli $(\lambda_0 ry, y) = (y, \lambda_0 ry)$, jolloin $\lambda_0(ry, y) = \overline{\lambda}_0(y, ry)$ eli $\lambda_0 \int_a^b r(x)|y(x)|^2\,dx = \overline{\lambda}_0 \int_a^b r(x)|y(x)|^2\,dx$, jollon $\lambda_0 = \overline{\lambda}_0$ ts. $\lambda_0 \in \mathbb{R}$. □

6.38. Lemma. Itseadjungoidun operaattorin L (ks. (6.32)) eri ominaisarvoihin liittyvät ominaisfunktiot ovat ortogonaalisia painofunktion r suhteen.

Todistus. Olkoon $\lambda_1 \neq \lambda_2$ ja $Lf_k + \lambda_k r f_k = 0 \quad (k = 1, 2)$. Koska L on itseadjungoitu, on

$$(-\lambda_1 r f_1, f_2) = (f_1, -\lambda_2 r f_2) \quad \text{eli} \quad (\lambda_1 - \lambda_2) \int_a^b r(x) f_1(x) \overline{g_2(x)}\,dx = 0\,,$$

josta ortogonaalisuus seuraa, sillä $\lambda_1 \neq \lambda_2$.

6.39. Lause. (Sturm–Liouville) Jos $BC^2(a, b)$:n määrittelevät reunaehdot ovat sellaisia, että L on itseadjungoitu $BC^2(a, b)$:ssä, ts. $(Lf, g) = (f, Lg)$ kaikille $f, g \in BC^2(a, b)$, on olemassa ääretön jono ominaisfunktioita

$$\phi_1,\ \phi_2,\ \phi_3,\ \ldots\,, \tag{6.40}$$

jotka ovat keskenään ortogonaalisia r:n suhteen:

$$\int_a^b r(x)\phi_j(x)\overline{\phi_k(x)}\,dx = 0 \qquad \forall\ j \neq k\,. \tag{6.41}$$

Jokainen rajoitettu välillä $[a, b]$ integroituva funktio voidaan kehittää sarjaksi

$$f(x) = \sum_{n=1}^{\infty} c_n \phi_n(x)\,, \tag{6.42}$$

joka suppenee pns-mielessä kohti f:ää, ts.

$$\left\| f(x) - \sum_{k=1}^{n} c_k \phi_k(x) \right\| \to 0 \quad \text{kun } n \to \infty\,.$$

Jos lisäksi $f \in BC^2(a, b)$, sarja suppenee jopa tasaisesti ja itseisesti kohti funktiota f koko välillä $[a, b]$.

Todistus. Sivuutetaan. □

6.43. Trigonometriset sarjat. Esimerkissä 6.12 todettiin, että funktiojoukko $\{\phi_0, \phi_1, \phi_2, \ldots\}$, missä

$$\phi_0(x) = \frac{1}{\sqrt{2\pi}}, \quad \phi_{2n-1}(x) = \frac{1}{\sqrt{\pi}} \sin nx, \quad \phi_{2n}(x) = \frac{1}{\sqrt{\pi}} \cos nx\,,$$

on ortonormaali välillä $[0, 2\pi]$. Lauseen 6.25 mukaan funktion $f \in C[0, 2\pi]$ paras approksimaatio $\{\phi_0, \phi_1, \phi_2, \ldots, \phi_{2n-1}, \phi_{2n}\}$:n virittämässä aliavaruudessa on

$$\sum_{k=0}^{2n} \alpha_k \phi_k\,, \quad \text{missä} \quad \begin{cases} \alpha_{2n-1} &= \frac{1}{\sqrt{\pi}} \int_0^{2\pi} f(x) \sin nx\, dx \quad (n = 1, 2, \ldots) \\ \alpha_{2n} &= \frac{1}{\sqrt{\pi}} \int_0^{2\pi} f(x) \cos nx\, dx \quad (n = 1, 2, \ldots) \\ \alpha_0 &= \frac{1}{\sqrt{2\pi}} \int_0^{2\pi} f(x)\, dx \end{cases},$$

joten approksimaatio voidaan kirjoittaa muotoon

$$A_0 + A_1 \cos x + B_1 \sin x + \cdots + A_n \cos nx + B_n \sin nx\,, \tag{6.44}$$

missä

$$\begin{aligned} &A_0 = \frac{1}{2\pi} \int_0^{2\pi} f(x)\, dx\,, \quad A_n = \frac{1}{\pi} \int_0^{2\pi} f(x) \cos nx\, dx\,, \\ &B_n = \frac{1}{\pi} \int_0^{2\pi} f(x) \sin nx\, dx\,. \end{aligned} \tag{6.45}$$

Approksimaatio (6.44) on f:n *Fourier-sarjan* osasumma.

Tapa, jolla kertoimet A_n ja B_n muodostetaan f:stä, ansaitsee lähempää tarkastelua. Näin johdutaan konvoluution käsitteeseen.

6.46. Konvoluutio. Kahden funktion $f, g \in C(\mathbb{R})$ *konvoluutio* on

$$(f * g)(x) = \int_0^{2\pi} f(x - t) g(t)\, dt = \int_0^{2\pi} f(t) g(x - t)\, dt\,. \tag{6.47}$$

(Yleisemmin: Riittää että f^2 ja g^2 ovat integroituvia.) Kaavan (6.47) toinen yhtälö seuraa helposti muuttujanvaihtokaavasta. Konvoluution perusominaisuuksia:

$$\begin{aligned} &(\alpha f + \beta g) * h = \alpha(f * h) + \beta(g * h)\,, \\ &f * (\alpha g + \beta h) = \alpha(f * g) + \beta(f * h)\,, \\ &f * g = g * f \quad \text{(kommutatiivisuus)}\,, \\ &(f * g) * h = f * (g * h) \quad \text{(assosiatiivisuus)}\,. \end{aligned} \tag{6.48}$$

Konvoluutiolla on seuraava yhteys Fourier-kertoimiin. Olkoon S_0 vakiofunktioiden (1-ulotteinen) avaruus ja S_k $\{\cos kx, \sin kx\}$:n virittämä (2-ulotteinen) avaruus. Avaruuksilla S_0 ja S_k on vastaavasti kantoina $\{\phi_0\}$ ja $\{\phi_{2k-1}, \phi_{2k}\}$.

Esimerkin 6.19 nojalla

$$\mathrm{proj}(f : \phi_{2k-1}, \phi_{2k}) = f * g_k\,, \quad \text{missä } g_k(x) = \frac{1}{\pi}\cos(kx)\,.$$

6.49. Esimerkki. Edellä esitetty Gramin–Schmidtin ortogonalisointi $\mathbb{R}^n$:n vektorijoukolle $\{x_1, \ldots, x_n\}$ voidaan toteuttaa seuraavasti. Muodostetaan $n \times n$ -matriisi x, jonka j:nnellä vaakarivillä on x_j, ja annetaan x syötteeksi funktiolle `gram`, joka tuottaa matriisin y, jonka j:s vaakarivi on ortogonalisoidun vektorijoukon $\{y_1, \ldots, y_n\}$ j:s vektori y_j. Funktion `gram` toimintaa testataan alla ohjelmassa `e633`, jossa kriteerinä on, että satunnaismatriisista x tuotetulle y:lle pätee, että `norm(y-gram(y))` on pieni.

```
function gr = gram(x)
% Given n by n real matrix x containing xj on row j
% orthogonal vectors yj are constructed by Gram-Schmidt process
    [m1,n1]= size(x);
    if ( m1 ~= n1)
       disp('Argument error in gram');
       break;
    end;
    dim = m1;
    y = zeros(dim,dim);
    y(1,:) = x(1,:);
    for n =2:dim
        s= zeros(1,dim);
        for k =1:n-1
            s = s +sum(x(n,:).*y(k,:))*y(k,:)/(norm(y(k,:))^2);
        end;
        y(n,:) = x(n,:)-s;
    end;
  gr = y;
% FILE  gram.m  ends.

% FILE  e633.m  begins.
% USES: gram.m
for dim = 2:10
```

```
    x = rand(dim,dim);
    y = zeros(dim,dim);
    y(1,:) = x(1,:);
    for n =2:dim
        s= zeros(1,dim);
        for k =1:n-1
            s = s +sum(x(n,:).*y(k,:))*y(k,:)/(norm(y(k,:))^2);
        end;
        y(n,:) = x(n,:)-s;
    end;
end;
disp('Tarkistus y:n ortogonaalisuudelle');
disp('on etta norm(y - gram(y)) = 0 ')
z =gram(y);
norm(z-y)
% FILE  e633.m  ends.
```

Ohjelman `e633` tulostama `norm(z-y)` on luokkaa 10^{-15}.

7 Fourier-sarjat

Muotoa

$$\frac{1}{2}a_0 + \sum_{n=1}^{\infty}(a_n \cos nx + b_n \sin nx) \tag{7.1}$$

olevaa sarjaa kutsutaan *trigonometriseksi sarjaksi*. Sitä sanotaan *Fourier-sarjaksi*, jos kertoimet a_n ja b_n on saatu integroituvasta funktiosta f seuraavasti:

$$\begin{cases} a_n = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos nx\,dx\,, & \text{kun } n = 0, 1, 2, \ldots \\ b_n = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin nx\,dx\,, & \text{kun } n = 1, 2, 3, \ldots \end{cases} . \tag{7.2}$$

Matematiikan alaa, joka tutkii trigonometrisia sarjoja ja niiden yleistyksiä, kutsutaan harmoniseksi analyysiksi tai Fourier-analyysiksi. Trigonometristen funktioiden jaksollisuus kuvastuu luonnollisella tavalla myös Fourier-analyysissä.

Tässä luvussa kertaamme lähinnä [LP]:ssä esitettyjä Fourier-sarjojen ominaisuuksia ja tutkimme mm. seuraavia kysymyksiä:

- Esittääkö f:n Fourier-sarja funktiota f?
- Mitä tapahtuu, jos funktio f on yhdessä pisteessä epäjatkuva?
- Mitä tapahtuu, jos $[0, 2\pi]$ vaihdetaan toiseksi väliksi?

Samaan alueeseen liittyvät mm. aihepiirit

- Fourier-integraalit ja -muunnokset,
- diskreetti Fourier-analyysi,

joihin myös tulemme viittaamaan.

Merkittäviä sovelluksia Fourier-analyysillä on mm.

- osittaisdifferentiaaliyhtälöiden teoriaan,
- signaalianalyysiin.

Kirjallisuutta: [D], [S1], [LI].
Sovellettua Fourier-analyysiä: [**?**], [Be], [K].

7.3. Määritelmä. Funktion f *jakso* on T, jos $f(x+T) = f(x)$ kaikille x ja T on pienin tällainen luku. Jos f:n jakso on T, niin $\int_a^{a+T} f(x)\,dx = \int_0^T f(x)\,dx$ kaikille a.

Edellä kohdassa 6.43 johdettiin funktion $f \in C[0, 2\pi]$ Fourier-sarjan kertoimet:

$$(7.4) \qquad S_f^n(x) = \frac{1}{2}a_0 + \sum_{k=1}^{n}(a_k \cos kx + b_k \sin kx)\,; \quad S_f(x) = \lim_{n\to\infty} S_f^n(x)\,;$$

$$\frac{1}{2}a_0 = \frac{1}{2\pi}\int_0^{2\pi} f(x)\,dx\,, \;\; a_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos kx\,dx\,, \;\; b_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin kx\,dx\,.$$

Ortogonaalifunktioita koskeva Parsevalin kaava 6.26 (2) antaa tietoa siitä, missä mielessä $f(x)$:ää voidaan approksimoida $S_f^n(x)$:llä (ks. myös 6.25), vertailusuureena

$$\delta_n(f) = \|f - S_f^n\|\,.$$

Siitä, että $\|f - f_n\| \to 0$, ei voida päätellä, että $f(x) \to f_n(x)$ jossakin pisteessä $x \in [0, 1]$. Näin ollen ei voida odottaa, että ehto $\delta_n(f) \to 0$ kertoisi jotain Fourier-sarjan pisteittäisestä suppenemisesta. On mm. olemassa jatkuvia funktioita, joiden Fourier-sarja hajaantuu äärellisen monessa pisteessä. (Ei tiedetä, voiko integroituvan funktion Fourier-sarja divergoida joka pisteessä.) Seuraava lause antaa valaistusta Fourier-sarjan suppenemiseen.

7.5. Lause. Jos f on integroituva funktio, jolla on jaksona 2π, niin $S_f(x)$ suppenee kohti lukua $(f(x+) + f(x-))/2$ jokaisessa pisteessä x, jossa f:llä on oikean- ja vasemmanpuoleinen derivaatta. Tässä

$$f(x+) = \lim_{\substack{y\to x\\ y>x}} f(y)\,, \quad f(x-) = \lim_{\substack{y\to x\\ y<x}} f(y)\,.$$

Todistus. Sivuutetaan (Ks. [D, s. 94]). □

7.6. Määritelmä. Funktio $f\colon \mathbb{R} \to \mathbb{R}$ on *parillinen*, jos $f(x) = f(-x)$ kaikille $x \in \mathbb{R}$, ja *pariton*, jos $f(x) = -f(-x)$ kaikille $x \in \mathbb{R}$.

7.7. Esimerkki. (1) cos on parillinen, sin pariton.
(2) f parillinen, g pariton $\Rightarrow$ fg pariton.
(3) f parillinen, g parillinen $\Rightarrow$ fg parillinen.
(4) f parillinen $\Rightarrow$ $\int_{-T}^{T} f(x)\,dx = 2\int_0^T f(x)\,dx\,.$
(5) f pariton $\Rightarrow$ $\int_{-T}^{T} f(x)\,dx = 0\,.$

7.8. Lemma. (1) Jos f on parillinen, niin S_f sisältää vain cos-termejä (missä vakiotermi luetaan kosinitermiksi: $1 = \cos 0x$).

(2) Jos f on pariton, niin S_f sisältää vain sin-termejä.

Todistus. (1) Kaavan 7.7 (2) nojalla nähdään, että $f(x)\sin nx$ on pariton, jolloin

$$b_k = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin kx\,dx = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin kx\,dx = 0\,.$$

(2) Samoin. □

7.9. Fourier-sarja välillä $(-\frac{T}{2}, \frac{T}{2})$. Välin $[0, 2\pi]$ valinta Fourier-sarjan määrittelyssä (7.4) on mielivaltainen. Yhtä hyvin voidaan muodostaa sarja välillä $[-T/2, T/2]$. Tällöin ($\omega_0 = 2\pi/T$)

$$f(t) = \frac{1}{2}a_0 + \sum_{k=1}^{\infty}(a_k\cos(k\omega_0 t) + b_k\sin(k\omega_0 t))\,;$$

$$a_n = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\cos(n\omega_0 t)\,dt \quad (n = 0, 1, 2, \ldots)\,,$$

$$b_n = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\sin(n\omega_0 t)\,dt \quad (n = 1, 2, \ldots)\,.$$

7.10. Esimerkkejä Fourier-sarjoista.

$$\sum_{n=1}^{\infty}\frac{\sin nx}{n} = \frac{\pi - x}{2} \qquad (0 < x < 2\pi)\,. \tag{7.11}$$

Olkoon annettu funktio

$$f(t) = \begin{cases} -1, \text{ kun } -T/2 < t < 0 \\ +1, \text{ kun } 0 < t < T/2 \end{cases} \quad ; \quad f(t+T) = f(t)\,. \tag{7.12}$$

Etsi f:n Fourier-sarja. Nyt ($\omega_0 = 2\pi/T$)

$$b_n = \frac{2}{T}\int_{-T/2}^{T/2} f(t)\sin(n\omega_0 t)\,dt = \frac{2}{T}\Big(\int_{-T/2}^{0} -\sin(n\omega_0 t)\,dt + \int_0^{T/2}\sin(n\omega_0 t)\,dt\Big)$$

$$= \frac{2}{T}\left(\frac{1}{n\omega_0} \int_{-T/2}^{0} \cos(n\omega_0 t) - \frac{1}{n\omega_0} \int_{0}^{T/2} \cos(n\omega_0 t) \right) = \frac{2}{n\pi}(1 - \cos(n\pi)) \,.$$

Koska $\cos(n\pi) = (-1)^n$, saamme että

$$b_n = \begin{cases} 0\,, & \text{kun } n \text{ parillinen} \\ \frac{4}{n\pi}\,, & \text{kun } n \text{ pariton} \end{cases} ;$$

$$f(t) = \frac{4}{\pi} \sum_{k=0}^{\infty} \frac{1}{2k+1} \sin((2k+1)\omega_0 t) \,.$$

(Kertoimet a_n häviävät automaattisesti, koska (7.12):n funktio f on pariton.)

Olkoon kolmanneksi

(7.13) $$f(x) = \cos^2 x \,.$$

Koska $f(x) = \frac{1}{2} + \frac{1}{2}\cos 2x$, on $a_0 = 1$, $a_2 = \frac{1}{2}$, kun taas muut kertoimet ovat 0.

7.14. Huomautus. Fourier-sarjojen yhteydessä on tapana automaattisesti laajentaa funktio perusjaksovälin (esim. $(0, 2\pi]$) ulkopuolelle jaksollisesti, jolloin $f(x) = f(x + 2\pi)$ kaikille x (juuri mainitussa esimerkissä).

7.15. Fourier-sarjan kompleksinen muoto. Eulerin kaavan (2.15) nojalla $e^{iy} = \cos y + i \sin y$, joten on odotettavissa, että Fourier-sarja voidaan esittää vaihtoehtoisesti e^{iy}:n avulla. Kompleksinen muoto on yleinen esim. signaalianalyysissä.

Lähtökohtana on ortogonaalisuus e^{ikx}:ille:

(7.16) $$\int_0^{2\pi} e^{ikx} e^{-ihx} \, dx = 0 \quad \text{kaikille } k, h \in \mathbb{Z}, \text{ joille } k \neq h.$$

Käyttäen myös tulosta $\int_0^{2\pi} e^{ikx} e^{-ikx} \, dx = 2\pi$ voidaan periaatteessa samaan tapaan kuin em. trigonometristen sarjojen tapauksessa nytkin johtaa integroituvan funktion Fourier-sarjan kertoimet:

(7.17) $$f(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx} \,; \quad c_k = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-ikx} \, dx = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) e^{-ikx} \, dx \,.$$

Yhteys kertoimiin a_k ja b_k yhtälöissä (7.2) saadaan Eulerin kaavasta:

$$\int f(x) e^{-ikx} \, dx = \int f(x) \cos kx \, dx - i \int f(x) \sin kx \, dx \,,$$

joten

$$(7.18)\quad c_k = \frac{1}{2}a_k - \frac{1}{2}ib_k\,, \quad c_0 = \frac{1}{2}a_0\,, \quad c_{-k} = \frac{1}{2}a_k + \frac{1}{2}ib_k \qquad (k = 1, 2, \ldots)\,.$$

Kääntäen

$$a_k = c_k + c_{-k} \quad (k = 0, 1, 2, \ldots)\,, \qquad b_k = i(c_k - c_{-k}) \quad (k = 1, 2, \ldots)\,.$$

Huomattakoon, että Fourier-sarjan kompleksiversion johtamiseksi olisimme myös voineet lähteä suoraan muodosta (7.1)–(7.2), käyttää kaavoja

$$\cos kx = \frac{1}{2}(e^{ikx} + e^{-ikx})\,, \quad \sin kx = \frac{1}{2i}(e^{ikx} - e^{-ikx})$$

ja termejä järjestäen — päättymättömiä summia varovasti käsitellen — päätyä muotoon (7.17).

7.19. Huomautus. Tärkeä ero Taylor- ja Fourier-sarjojen välillä on, että Taylor-sarja käyttää hyväkseen vain tietoja funktiosta yhdessä pisteessä (derivaatat), kun taas Fourier-sarja käyttää tietoja funktion arvoista välillä $[0, 2\pi]$ (integraalit).

7.20. Fourier-sarjan suppenemisesta. Lauseessa 7.5 todettiin, että Fourier-sarja pisteessä x_0 suppenee ko. ehdoin kohti arvoa $\frac{1}{2}(f(x_0+) + f(x_0-))$ (joka on $f(x_0)$, jos f on jatkuva kohdassa x_0). Jos f on epäjatkuva kohdassa x_0, esiintyy Fourier-sarjojen osasummilla $S_f^n(x)$ kohdassa x_0 erikoinen "hyppyilmiö", ns. *Gibbsin ilmiö,* joka voidaan helposti kokeellisesti todentaa MATLAB-testein.

Gibbsin ilmiö on ehkä yllättävä, kun ajatellaan tavanomaista suppenemiskäsitettä. Toisaalta on hyvä tietää, että jatkuville funktioille, joiden derivaatta on myös jatkuva paitsi äärellisen monessa pisteessä, pätee Fourier-sarjan *tasainen suppeneminen.*

7.21. Lause. Jos $f\colon [0, 2\pi] \to \mathbb{R}$ on jatkuva ja $f'\colon [0, 2\pi] \setminus \{a_1, \ldots, a_p\} \to \mathbb{R}$ on myös jatkuva, niin $S_f^n \to f$ välillä $[0, 2\pi]$ tasaisesti: kaikille $\varepsilon > 0$ on sellainen n_0, että $|f(x) - S_f^n(x)| < \varepsilon$ aina kun $n \geq n_0$ ja $x \in [0, 2\pi]$. □

Tasaisesti suppenevan sarjan kätevä ominaisuus on, että sen voi integroida termeittäin.

7.22. Diskreetti Fourier-sarja. Käytännön harmonisessa analyysissä tarkastellaan diskreettejä x:n arvoja $x_k = 2\pi k/n$ $(k = 0, 1, \ldots, n-1)$ ja vastaavia

funktion arvoja f_k. Esimerkiksi x voi vastata aikamuuttujaa ja f signaalin arvoja, ja johdumme seuraavaan perustehtävään.

Tehtävä: Etsi luvut $c_0, \ldots, c_{n-1}$ s.e. yhtälö

$$f(x) = c_0 + c_1 e^{ix} + \cdots + c_{n-1} e^{i(n-1)x} \tag{7.23}$$

toteutuu pisteissä $x = x_k \quad (k = 0, 1, \ldots, n-1)$.

Erikoistapaus. $n = 4$ ja $f(x_k) = 2 + 2k$, $x_k = 2\pi k/4 \quad (k = 0, 1, 2, 3)$. Nyt $e^0 = 1$, $e^{i\frac{\pi}{2}} = i$, $e^{i\pi} = -1$, $e^{i\frac{3\pi}{2}} = -i$ jne., joten yo. tehtävä antaa yhtälöryhmän

$$\begin{array}{ll} x_0: & c_0 + c_1 + c_2 + c_3 = 2 \\ x_1: & c_0 + ic_1 - c_2 - ic_3 = 4 \\ x_2: & c_0 - c_1 + c_2 - c_3 = 6 \\ x_3: & c_0 - ic_1 - c_2 + ic_3 = 8 \end{array} \quad . \tag{7.24}$$

Kerroinmatriisi on

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & i & i^2 & i^3 \\ 1 & i^2 & i^4 & i^6 \\ 1 & i^3 & i^6 & i^9 \end{bmatrix} . \tag{7.25}$$

Tehtävä (7.23) saa siis nyt muodon

$$Ac = f\,. \tag{7.26}$$

Yhtälön (7.26) ratkaisemiseksi tarvitaan A^{-1}. Käytämme hyväksi A:n erityistä muotoa (7.25), jolloin A^{-1} löytyy seuraavasti. Muodostetaan kompleksikonjugoitu matriisi

$$\overline{A} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & (-i)^2 & (-i)^3 \\ 1 & (-i)^2 & (-i)^4 & (-i)^6 \\ 1 & (-i)^3 & (-i)^6 & (-i)^9 \end{pmatrix} . \tag{7.27}$$

Tällöin lasku osoittaa, että

$$\overline{A}A = 4I, \quad \text{joten} \quad A^{-1} = \frac{1}{4}\overline{A}\,. \tag{7.28}$$

Saadaan (7.24):n ratkaisu $c = \frac{1}{4}\overline{A}f$ eli

$$c_0 = 5\,, \quad c_1 = -1 + i\,, \quad c_2 = -1\,, \quad c_3 = -1 - i\,. \tag{7.29}$$

Yleinen tapaus. Yo. ratkaisumenettely soveltuu ongelmaan (7.23) myös yleisesti. Merkitään $w = e^{2\pi i/n}$, jolloin (7.24):ää vastaava yhtälöryhmä saa muodon

$$\begin{array}{rl} x_0: & c_0 + c_1 + \cdots + c_{n-1} = f_0 \\ x_1: & c_0 + c_1 w + \cdots + c_{n-1} w^{n-1} = f_1 \\ \vdots & \vdots \\ x_{n-1}: & c_0 + c_1 w^{n-1} + \cdots + c_{n-1} w^{(n-1)(n-1)} = f_{n-1} \end{array} \quad .$$

Vastaava matriisi on nyt (F — Fourier)

$$F_n = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{n-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & w^{n-1} & w^{2(n-1)} & \cdots & w^{(n-1)^2} \end{bmatrix}, \tag{7.30}$$

joten tehtävä (7.23) palautuu muotoon

$$F_n c = f\,. \tag{7.31}$$

Matriisin F_n käänteismatriisi on $\frac{1}{n}\overline{F}_n$, ja pätee seuraava.

7.32. Lause. Yhtälön (7.23) ratkaisun antaa

$$c = \Big(\frac{1}{n}\overline{F}_n\Big) f = (F_n^{-1}) f\,.$$

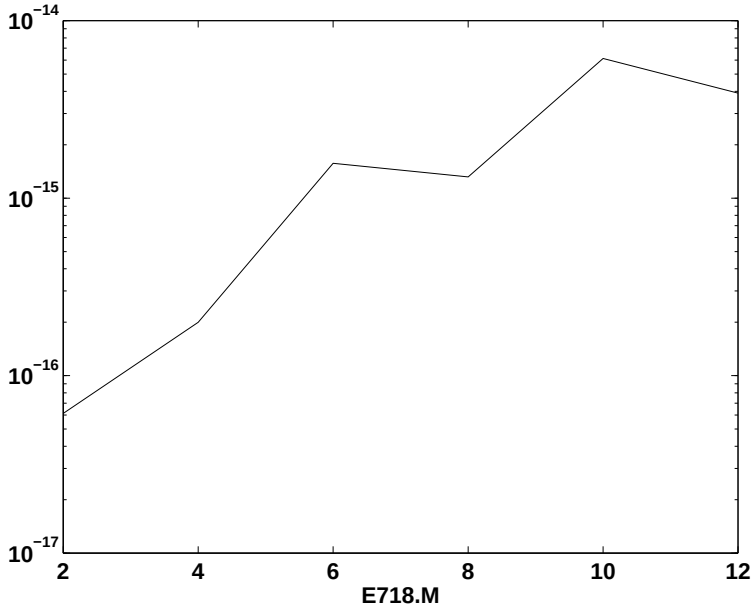
FFT:n avulla tulemme osoittamaan, miten lauseen 7.32 matriisikertolasku on tehokkainta tehdä.

7.33. Esimerkki. Haluamme vakuuttua MATLAB-kokein, että lauseen 7.32 kaava pätee. Teemme seuraavan ohjelman.

```
% FILE  e718.m  begins.
% This experiment shows that the inverse of the n x n
% Fourier matrix Fw  is:  (1/n)Fconj(w).
% Here: Fw(p,q) = w^((p-1)(q-1)) and w is the nth root of unity.
clear;
data =[];
i = sqrt(-1);
for n=2:2:12
  w =exp(i*2*pi/n);
  clear F;
  for p=1:n
    for q=1:n
      F(p,q) = w^((p-1)*(q-1));
    end;
  end;
  wc =exp(-i*2*pi/n);
```

```
  clear FC;
  for p=1:n
    for q=1:n
      FC(p,q) = wc^((p-1)*(q-1));
    end;
  end;
  c=norm((1/n)*FC*F- eye(n));
  data =[data; [n c]];
  % F, pause;
end;
x= data(:,1);  y = data(:,2);
semilogy(x,y),  xlabel('E718.M');
% FILE  e718.m  ends.
```

Ohjelman tulostama alla oleva kuva osoittaa, että lause 7.32 pätee numeerisestikin lähes kone-epsilonin tarkkuudella.



7.34. Nopea Fourier-muunnos (FFT). Tarkastellaan matriisia F_n, joka esiintyy n:n pisteen diskreetin Fourier-muunnoksen yhteydessä:

$$F_n = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & w & w^2 & \cdots & w^{n-1} \\ 1 & w^2 & w^4 & \cdots & w^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & w^{n-1} & w^{2(n-1)} & \cdots & w^{(n-1)^2} \end{bmatrix} \quad \text{eli} \quad (F_n)_{ij} = w_n^{(i-1)(j-1)},$$

missä $w_n = e^{2\pi i/n} = \cos\frac{2\pi}{n} + i\sin\frac{2\pi}{n}$. Luvun w_n^p $(p = 0, 1, 2, \ldots)$ mahdollisia arvoja on n kpl, joten matriisissa F_n toistuvat samat luvut alkioina useita kertoja eri paikoissa. Tätä seikkaa käytetään matriisitulon $F_n x$ laskemisen nopeuttamiseksi seuraavasti.

Olkoon $n = 2m$. Silloin $w_n^2 = w_m$. Kirjoitetaan

$$\begin{aligned} x &= (x_0, x_1, \ldots, x_{n-2}, x_{n-1})\,, \\ x' &= (x_0, x_2, \ldots, x_{n-4}, x_{n-2})\,, \\ x'' &= (x_1, x_3, \ldots, x_{n-3}, x_{n-1})\,. \end{aligned}$$

Merkitään $y' = F_m x'$ ja $y'' = F_m x''$. Matriisin koko on alkuperäisestä $n \times n$:stä muuttunut $m \times m = \frac{n}{2} \times \frac{n}{2}$:ksi. Kuitenkin näistä saadaan $F_n x$.

7.35. Lause. Jos $n = 2m$, matriisin $y = F_n x$ komponentit ovat

$$\begin{aligned} y_j &= y'_j + w_n^j y''_j \qquad (j = 0, \ldots, m-1)\,, \\ y_{j+m} &= y'_j - w_n^j y''_j \qquad (j = 0, \ldots, m-1)\,. \end{aligned}$$

Todistus:

$$y_j \;=\; \sum_{k=0}^{n-1} w_n^{kj} x_k \;=\; \sum_{k=0}^{m-1} w_n^{2kj} x_{2k} \;+\; \sum_{k=0}^{m-1} w_n^{(2k+1)j} x_{2k+1}$$

$$= \sum_{k=0}^{m-1} w_m^{kj} x'_k \;+\; w_n^j \sum_{k=0}^{m-1} w_m^{kj} x''_k\;.$$

Sijoitetaan tulokseen j:n paikalle $j+m$ ja huomataan, että $w_m^{km} = 1$, $w_n^{j+m} = -w_n^j$:

$$y_{j+m} \;=\; \sum_{k=0}^{m-1} w_m^{k(j+m)} x'_k \;+\; w_n^{j+m} \sum_{k=0}^{m-1} w_m^{k(j+m)} x''_k$$

$$= \sum_{k=0}^{m-1} w_m^{kj} x'_k \;-\; w_n^j \sum_{k=0}^{m-1} w_m^{kj} x''_k\;. \quad \square$$

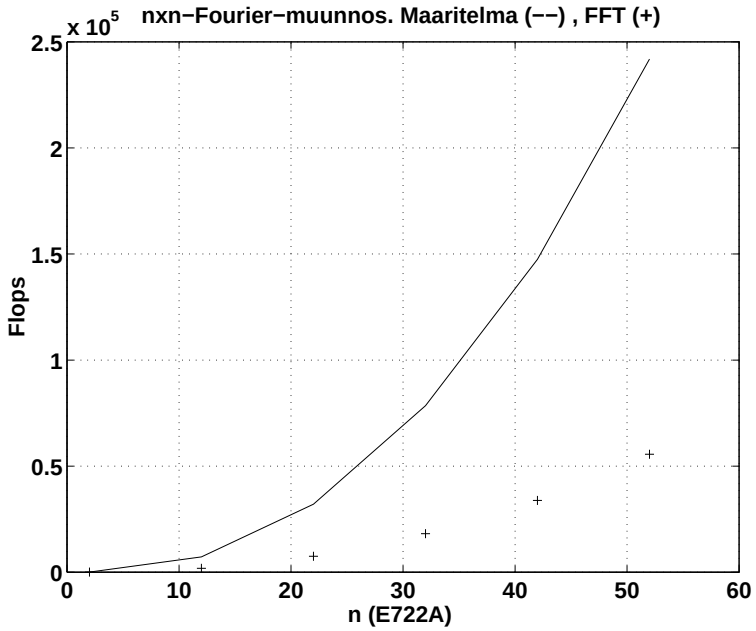
7.36. Huomautus. Lauseen 7.35 merkitys on siinä, että lausekkeen $F_n x$ kompleksisuus vähenee. Lausekkeen $F_n x$ laskeminen vaatii n^2 operaatiota tavanmukaisesti tehtynä. Lause 7.35 palauttaa tilanteen $F_m x'$:n ja $F_m x''$:n laskemiseen, joista kumpikin on kompleksisuutta $m^2 = (\frac{n}{2})^2$, sekä lauseessa esiintyviin kerto- ja yhteenlaskuihin, joiden kompleksisuus on $4m = 2n$. Siis $F_m x$ saadaan lasketuksi

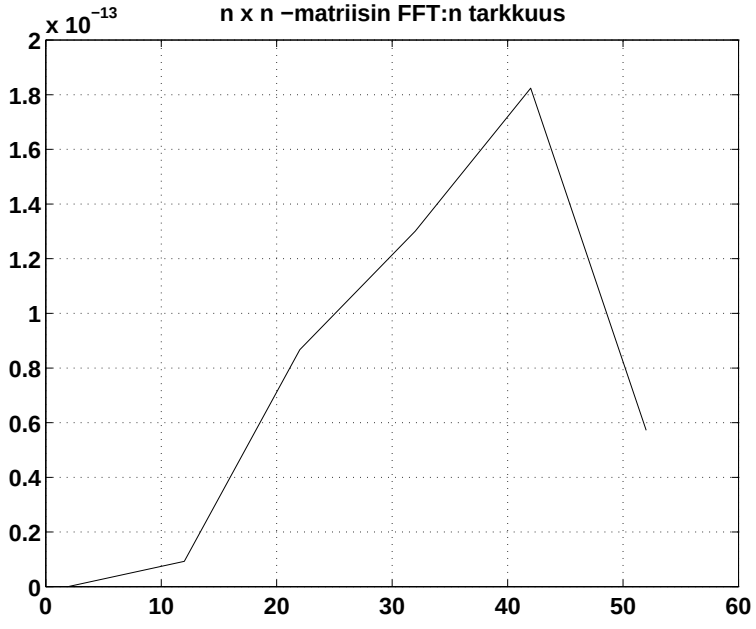
$\frac{n^2}{4} + 2n$ operaatiolla, joka on $<< n^2$ jos n on suuri. Siinä tapauksessa, että $n = 2^p$, voidaan lausetta soveltaa toistuvasti, jolloin kompleksisuus edelleen alenee.

7.37. Esimerkki. Haluamme vakuuttua MATLAB-kokein, että lauseen 7.35 kaava pätee. Teemme seuraavan ohjelman.

```
% FILE  e722.m  begins.
% Given a positive integer m = 1,2,3,.. let n = 2m, i = sqrt(-1),
% w = exp(i*2*pi/n), wm = exp(i*2*pi/m), and let F be the n by n
% and Fm the m by m Fourier matrices. Let x be a n by 1 column
% vector and x1 its odd components and x2 its even components,
% x1 = x(1:2:n-1),  x2 = x(2:2:n), y = Fx, y1 = Fm*x1, y2 = Fm*x2.
% Note that for big n, computation of y is much more tedious
% than the computation of y1 since it requires n by n matrices
% while the latter one only needs (n/2) by (n/2) matrices.
% FFT tells us that y can be obtained as ym
% from y1 and y2 in a simple way, thus:
%    for j =1:m
%    ym(j,1)  = y1(j,1) + (w^(j-1))*y2(j,1);
%    ym(m+j,1) = y1(j,1) -(w^(j-1))*y2(j,1);
%    end;
%
clear;
i = sqrt(-1);
info = [];
for n=2:10:60
  i = sqrt(-1);
  x = rand(n,1);
  clear y;
  w =exp(i*2*pi/n);
  clear F;
  flo1 = flops;
    for p=1:n
      for q=1:n
        F(p,q) = w^((p-1)*(q-1));
      end;
    end;
  y = F*x;  % Tassa tehdaan raaka lasku
  flo1 = flops-flo1;
  m =n/2;
```

```
  flo2 = flops;
  wm =exp(i*2*pi/m);
  clear Fm;
  for p=1:m
    for q=1:m
      Fm(p,q) = wm^((p-1)*(q-1));
    end;
  end;
  x1 = x(1:2:n-1,1);
  x2 = x(2:2:n,1);
  y1 = Fm*x1;
  y2 = Fm*x2;  % Tassa tehdaan FFT:n mukainen
               % lasku
  for j =1:m
    ym(j,1)  = y1(j,1) + (w^(j-1))*y2(j,1);
    ym(m+j,1) = y1(j,1) -(w^(j-1))*y2(j,1);
  end;
  flo2 = flops -flo2;
  info = [info; n flo1 flo2 (norm(y-ym))];
end;
plot(info(:,1),info(:,2),info(:,1),info(:,3),'k+'), grid,
title('nxn-Fourier-muunnos. Maaritelma (--) , FFT (+) '),
ylabel('Flops'),
xlabel('n (E722A) '), pause;
print -deps fig722a;
plot(info(:,1),info(:,4)), grid,
title('n x n -matriisin FFT:n tarkkuus'),
%ylabel('FFT - tarkka'), xlabel('n  (E722B) '), pause;
print -deps fig722b
% FILE  e722.m  ends.
```


nxn−Fourier−muunnos. Maaritelma (−−) , FFT (+)
x 10^5
Flops
n (E722A)
0
0.5
1
1.5
2
2.5
0
10
20
30
40
50
60


n x n −matriisin FFT:n tarkkuus
x 10^−13
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
10
20
30
40
50
60

8 Maplen perusteet

Maple on alun perin kanadalaisen Waterloon yliopiston ja zürichläisen ETH:n vuodesta 1980 kehittämä symbolisen ja numeerisen laskennan ohjelmisto (ks. [MAPLE]), nykymuodossaan tarkemmalta nimeltään Maple V (ks. [MplV]). Se on vakiinnuttanut asemansa muiden symbolisten ohjelmistojen (mm. Mathematica, Macsyma, Reduce, Derive) joukossa. Maplen käyttäjille on mm. oma sähköpostilistansa (MUG), ja tätä ohjelmistoa käytetään matematiikan opetuksessa esim. useimmissa USA:n huippuyliopistoista. Täysin akateeminen alkuperä heijastuu ohjelmiston avoimessa luonteessa: yli 80% Maple V:n tuhansista komennoista (funktioista) on ohjelmoitu helppotajuisella Maple-kielellä ja on käyttäjän luettavissa ja muutettavissa. (Maplen ydin on sen sijaan C-kielestä konekielelle käännettynä.)

Kuten MATLAB, myös Maple on tulkki ja sillä työskentely muistuttaa MATLABilla työskentelyä. Maple-kieli on kuitenkin luonteeltaan varsin toisenlainen kuin MATLAB-kieli. Maplen ja Maple-kielen peruspiirteitä on kuvattu ohjelmiston mukana toimitettavissa "virallisissa" oppaissa [Mpl/L], [Mpl/P] ja [Mpl/H], graafisen Maple-version opastusjärjestelmässä (esim. Help-valikon New User's Tour) sekä suomenkielisessä kirjassa [A]. Lukijan oletetaan perehtyvän johonkin näistä, lähinnä [Mpl/L]:ään, New User's Touriin tai [A]:han, ja käyttävän omatoimisesti Maplen help-komentoja rinnakkain tämän luvun opiskelun kanssa. Muusta kirjallisuudesta voidaan mainita [SK].

Laaja lista ja linkkikokoelma erilaisista Maple V -lähteistä on osoitteessa

`http://www.maplesoft.com/www/cybermath/sh_resources.html` .

Maple on kielenä hyvin laaja ja siihen on lisäksi kehitetty monia erityistarkoituksiin soveltuvia ohjelmapaketteja, jotka tulevat itse ohjelman mukana. Numeerisessa laskennassa Maple on usein tuskallisen hidas MATLABiin verrattuna eikä sitä kannata käyttää ongelmiin, jotka voidaan ratkaista MATLABilla.

Perehdymme nyt esimerkkien avulla Maplen käytön peruspiirteisiin DOS/Windows- (tai NT-) ympäristössä. Unix/XWindows-ympäristössä käyttö on pienin eroin samanlaista. Kummassakin voidaan käyttää sekä tekstipohjaista että graafista käyttöliittymää, mutta kurssilla esiintyvät todennäköisimmin yhdistelmät NT-ympäristö & graafinen käyttöliittymä ja Unix-ympäristö & tekstipohjainen käyttöliittymä. Edellisessä käynnistys tapahtuu kuvakkeesta, jälkimmäisessä komennolla `maple`. (XWindows-ympäristössä käynnistetään graafinen versio komennolla `xmaple`.) Tekstipohjainen versio lopetetaan komennolla `quit`, `done` tai `stop`. Graafisessa liittymässä ovat käytössä ns. *työarkit* (Maple WorkSheet — talletettuina `.mws`-tiedostot), joissa viimeksi mainitut komennot lopettavat (tallettamatta ja varoittamatta!) ko. työarkin.

Ohjelman syöterivin normaali hopute on muotoa "`> `". Muut paitsi help- ja lopetuskomennot on päätettävä tiettyyn *loppumerkkiin:* jos komennon lopussa on puolipiste, komennon antama tulos näytetään (normaali tilanne); jos komento loppuu kaksoispisteeseen, komento suoritetaan, mutta sen palauttama arvo jää piiloon. Loppumerkkejä ei typografisista syistä tässä tekstissä *aina* näytetä.

Käyttöjärjestelmän komento, esim. `dir/p`, voidaan Maplessa toteuttaa komentamalla `system("dir/p")` — tekstipohjaisessa käyttöliittymässä myös huutomerkkitekniikalla (kuten MATLABissa), esim. `!dir *.mws`. Oletushakemistossa sijaitsevan tiedoston `test.1st` sisältö saadaan kuvaruudulle komennolla `!type test.1st` tms., ja se saadaan luetuksi työtilaan komennolla `read "test.1st";` — tällöin suoritetaan kaikki tiedostossa olevat Maple-komennot (ensimmäiseen virheeseen asti). Muuttujan `a` arvo saadaan talletetuksi tiedostoon `info.txt` komennolla `save a, "info.txt"`. Lainausmerkit tiedostojen nimissä ovat Maplen sisällä tarpeen vain, mikäli niissä on sellaisia "komentomerkkejä" kuin '`.`' tai '`/`'. Niinpä lukukäsky voi vaihdella muodosta `read test` aina muotoon `read "/home/okanerva/maple/test.mpl"`. Huomattakoon, että tässä on myös DOS/Windows-järjestelmän alla käytettävä *vino*viivaa — tai sitten koodattava *keno*viiva muodossa '`\\`'.

Yhtenä erityispiirteenä on sulkujen `(), [], {}` käyttö. Toinen tärkeä seikka on isojen ja pienten kirjaimien erottelu: esim. `a` ja `A` ovat eri symboleita, ja kirjaimen kokoa voi käyttää ilmaisemaan (ihmiselle) esimerkiksi haluamaansa sävyä.

Symboli	Käyttö
`()`	termien ryhmittely ja argumenttien merkitseminen; esim. `(a+b)*sqrt(3)`
`[]`	listojen merkintä ja listan alkion poiminta; esim. `L:=[1,2,3,4];` ja `L[2]`
`{}`	joukkojen merkintä; esim. `S:={1,2,3,4};`
`%`	viittaa edellisen käskyn tulokseen (ennen Release 5:ttä merkki `"` oli varattu tähän)

Opastusta käskyjen käytöstä saadaan seuraavaan tyyliin:

```
?
?plot
??plot
?plot,options
```

Graafisen käyttöliittymän tapauksessa käytettävissä on lisäksi monipuolinen hiiren avulla toimiva helppi, jossa on mm. hypertekstiominaisuudet.

Esitämme alla pikakurssimaisesti Maplen käyttöä esimerkkien valossa. Teemme esimerkit tavallisesti tiedostoiksi, jolloin niiden muokkaus on helppoa ja uudelleenkirjoituksen vaiva toistettaessa säästyy. (Toisaalta tekstipohjaisessa Maplessa

on käytössä *komentopino*, ja työarkeilla taas voi vapaasti liikkua ylös, alas muuttamassa ja uudelleen suorittamassa komentojaan — jolloin on muistettava esimerkiksi, että merkki '%' viittaa aina *viimeksi suoritetun* komennon tulokseen, eikä ensisijaisesti yläpuolella näkyvään tulokseen, ja että kuva tapahtumien kulusta voi muutenkin häiriintyä.)

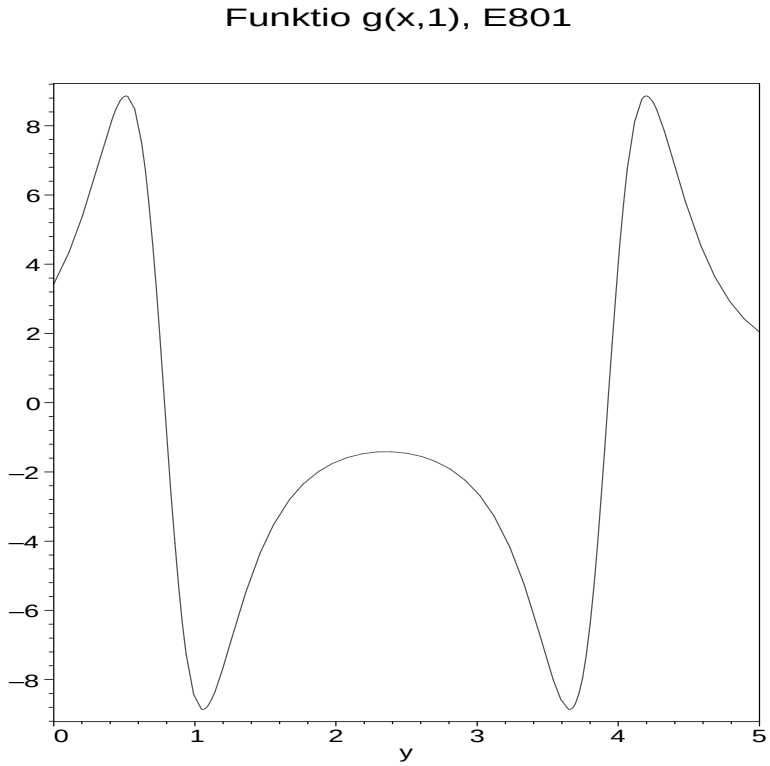
Kuten jo edellä todettiin, tiedoston luku työtilaan tapahtuu esim. komennolla `read "e801.mpl"`, jolloin myös tiedoston sisältämät Maple-käskyt toteutetaan. (Näiden tiedostojen nimien loppuosaksi olemme valinneet selvyyden vuoksi `mpl`:n — sillä ei sinänsä ole väliä (poikkeuksena ovat kuitenkin Maplen sisäisen esitysmuodon `.m`-tiedostot). Jos nimessä ei ole erikoismerkkejä, ei tarvita lainausmerkkejä: `read e801`.) Työtila on syytä ajoittain puhdistaa komennolla `restart` . Yksittäinen muuttuja `a` voidaan puhdistaa komentamalla `a:='a'` .

8.1. Funktion määrittely. Maplessa funktion määrittely tapahtuu toisin kuin monissa muissa ohjelmointikielissä, nimittäin nuolimerkintää käyttäen. Tuloksena on erikoistyyppinen proseduuri; lähes sama tulos saataisiin Maplen yleisempää proseduurimäärittelyä normaaliin tapaan suoraviivaisesti käyttäen. Alla olevassa tiedostossa määritellään eräs funktio ja sen n:s derivaatta.

```
# FILE  e801.mpl  begins.
# Funktion maarittely ja derivointi, for-lause

x:='x';  y:='y':     # Variables for 'diff' and 'plot'
                     # must be cleared.

f:= x-> tan(sin(x) + cos(x));

#  Huomaa nuolimerkinnan kaytto funktion maarittelyssa!
#
#  Kirjoittamalla f(x):= sin(x) EI saada aikaan funktiota f,
#  vaan ainoastaan symbolinen nimi lausekkeelle sin(x):
#  funktionaalista yhteytta x:n ja sin(x) valille ei synny.
#
#  Toinen toimiva, ehka suositeltavampi, tapa Maplessa olisi seuraava:
#  f:= tan@(sin + cos);

g:= (y,n) -> eval( diff(f(x),x$n), x=y);

#  Derivointi n kertaa.   ! Tallainen eval ei toimi Release 4:ssa.
#                         ! Siina voisi kayttaa eval@subs tms.
#
```

```
#  Toinen tapa olisi
#  g:= (y,n) -> (D@@n)(f)(y);
#
#  Mutta myos seuraava toimii!
#  g:= (x,n) -> diff(f(x),x$n);

for p to 3 do print(p,‘. derivaatta : ‘, g(y,p)) od;
plot(g(y,1), y=0..5, axes=BOXED, title=" Funktio g(x,1), E801 ");
# FILE  e801.mpl  ends.
```

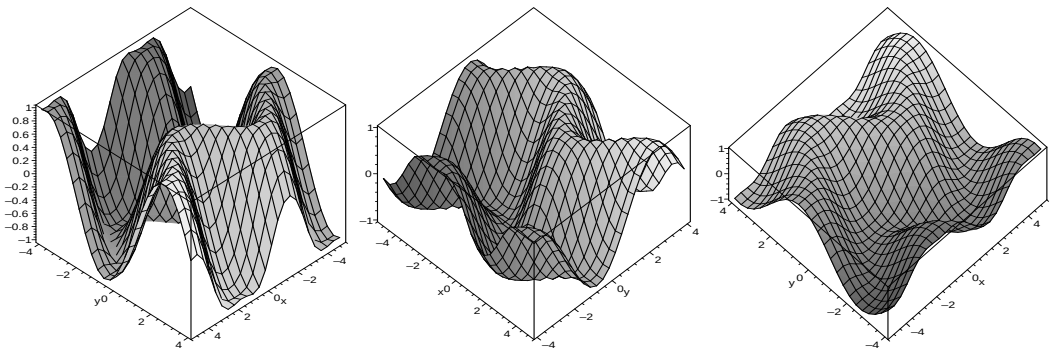
```
                                  x := x

                        f := x -> tan(sin(x) + cos(x))

                    g := (y, n) -> diff(f(x), x $ n)|
                                                    |x = y

                                                    2
       1, . derivaatta : , (1 + tan(sin(y) + cos(y)) ) (cos(y) - sin(y))

2, . derivaatta : ,

                                                    2                   2
    2 tan(sin(y) + cos(y)) (1 + tan(sin(y) + cos(y)) ) (cos(y) - sin(y))

                               2
     + (1 + tan(sin(y) + cos(y)) ) (-sin(y) - cos(y))

                             2 2                  3
3, . derivaatta : , 2 (1 + %1 )  (cos(y) - sin(y))

           2        2                   3
     + 4 %1  (1 + %1 ) (cos(y) - sin(y))

                   2
     + 6 %1 (1 + %1 ) (cos(y) - sin(y)) (-sin(y) - cos(y))

              2
     + (1 + %1 ) (-cos(y) + sin(y))

%1 := tan(sin(y) + cos(y))
```



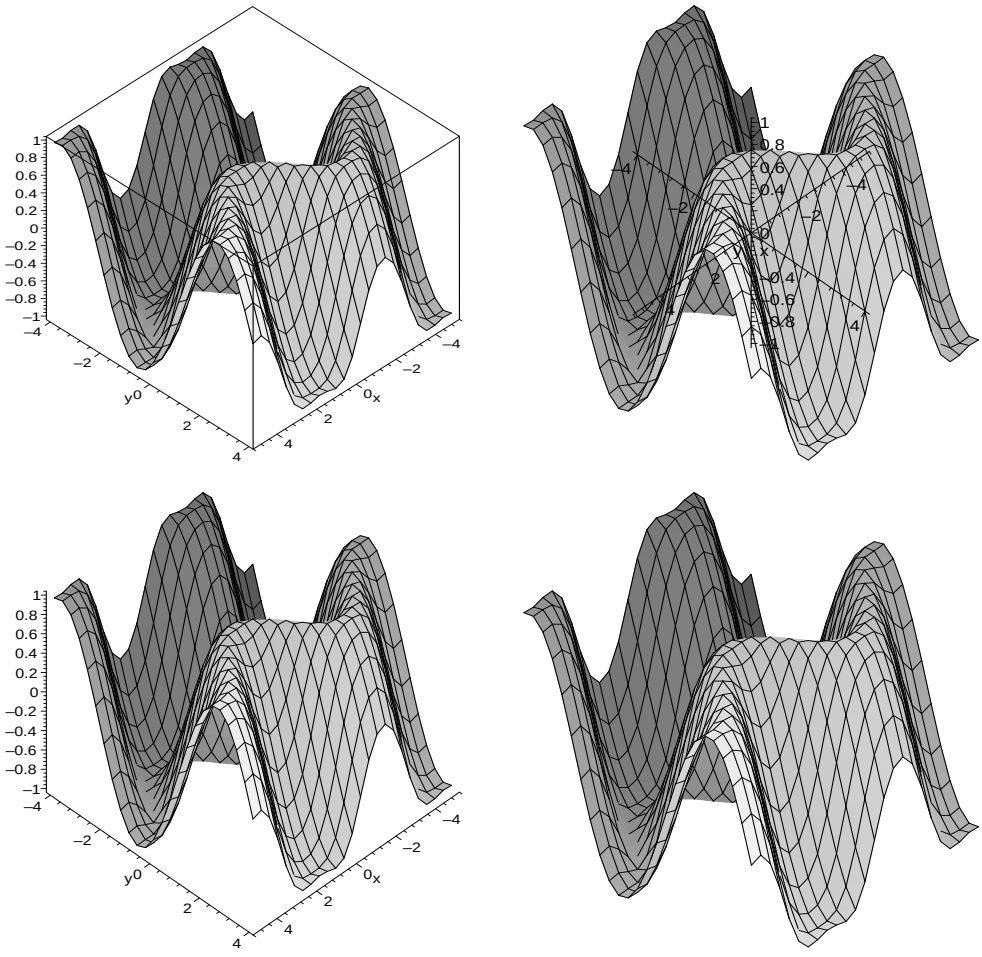
8.2. Piirtokäskyt. Kuten komennolla `?plots` voidaan todeta, on Maplessa (`plots`-paketissa) lukuisia plot- eli piirtokäskyjä ja näiden käyttöön liittyviä apukäskyjä — peruskäskyjen `plot` ja `plot3d` lisäksi. Käskyt liittyvät funktioiden ja pintojen piirtoon; myös lyhyitä "elokuvia" eli animaatioita saadaan helposti aikaan (`?animate`). Alla oleva tiedosto antaa viitteitä näiden käskyjen käytöstä.

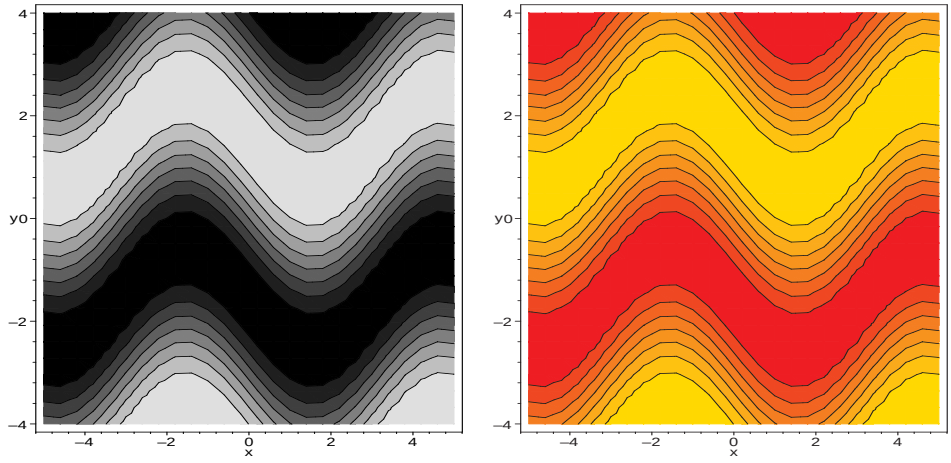
```
# FILE  e802.mpl  begins.

x:='x':  y:='y':       # plot3d-muuttujien oltava vapaita.
g:= (x,y)->sin(sin(x)+y);
a:= plot3d( g(x,y), x=-5..5, y=-4..4, axes=BOXED): # Huomaa kaksoipiste!
with(plots):
for j to 3 do display(a,orientation=[135-90*j, 60-15*j]) od;

#  Seuraavakin onnistuu:

for j in [BOXED,NORMAL,FRAMED,NONE] do display(a,axes=j) od;

#  Tasa-arvo- eli korkeuskayria askeiselle funktiolle:
```

```
contourplot(g(x,y), x=-5..5, y=-4..4, axes=boxed,
                    filled=true, coloring=[black,white]);
contourplot(g(x,y), x=-5..5, y=-4..4, axes=boxed, filled=true);
# FILE  e802.mpl  ends.
```



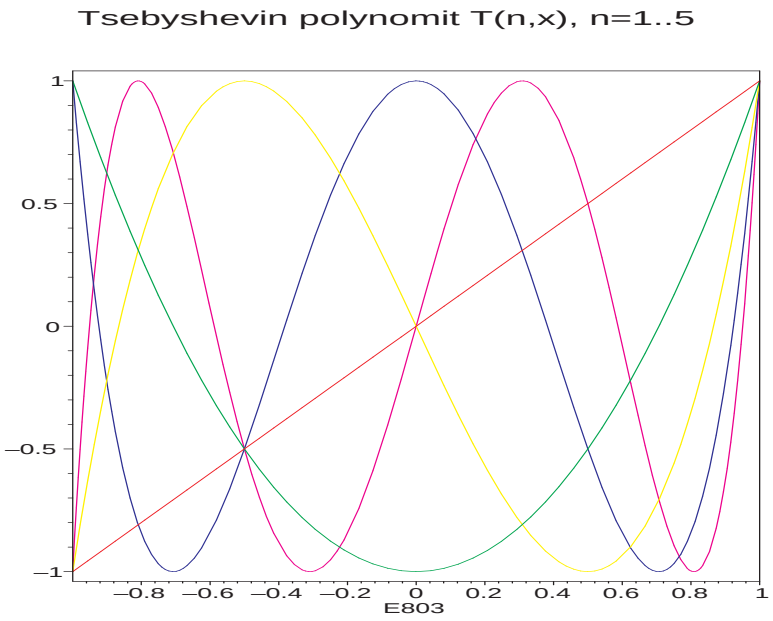
(Kuvat on ladottaessa asetettu rinnakkain.)



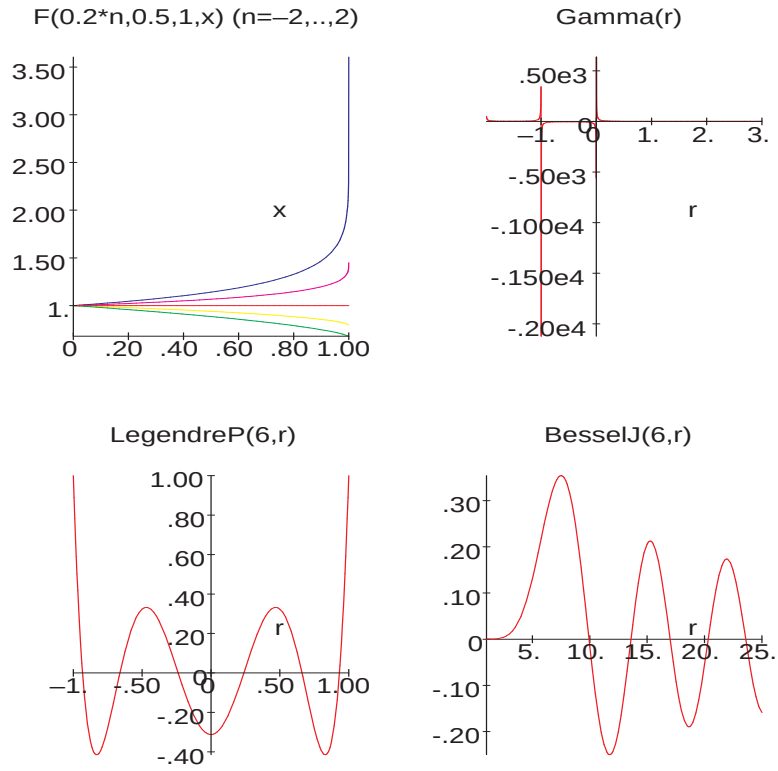


8.3. Erikoisfunktiot. Useimmat matemaattisen fysiikan erikoisfunktiot ja ortogonaalifunktiot ovat helposti käytettävissä Maplessa. Usein esiintyville funktioille on kätevää ottaa käyttöön sopivat lyhenteet, mitä kuitenkaan alla ei oikeastaan tarvitsisi tehdä.

```
# FILE  e803a.mpl  begins.

TT:= orthopoly[T];

#  Toinen vaihtoehto on komentaa ensin  with(orthopoly,T);
#  jolloin ladataan Chebyshevin polynomin T maaritelma
#  orthopoly-paketista
#  (komennolla  with(orthopoly)  koko paketti).
#  Silloin  T  merkitsee samaa kuin  TT  ylta.

plot([seq(TT(n,x), n=1..5)], x=-1..1,
     axes=boxed, title="Tsebyshevin polynomit T(n,x), n=1..5",
     labels=["E803", " "]);

#  Huomaa, etta samaan kuvaan haluttavista funktiolausekkeista
#  muodostettiin lista plot:n ensimmaiselle argumenttipaikalle!
# FILE  e803a.mpl  ends.
```




```
# FILE  e803b.mpl  begins.

F:=(a,b,c,z) -> hypergeom([a,b],[c],z);

k1:= plot({seq(F(0.2*n,0.5,1,x),n=-2..2)}, x=0..1,
          title="F(0.2*n,0.5,1,x) (n=-2,..,2)"):
k2:= plot(GAMMA(r), r=-1.99..3, title="Gamma(r)"):
k3:= plot(LegendreP(6,r), r=-1..1, title="LegendreP(6,r)"):
k4:= plot(BesselJ(6,r), r=1..25, title="BesselJ(6,r)"):
k5:= plots[display](array(1..2,1..2,[[k1,k2],[k3,k4]])):
k5;

#  Huomaa kaksois- ja puolipisteiden kaytto.
#  Talla kertaa (k1:ssa) kaytettiin joukkoa
#  usean kuvaajan saamiseksi yhteen kuvaan.
# FILE  e803b.mpl  ends.
```



Muita tarpeellisia funktioita ovat mm. `Zeta(z)`, `ithprime(i)`, `GAMMA(z)`, `irem(m,n)`.

8.4. Työarkit ja tallettaminen. Kuten aiemmin on sanottu, graafisen Maple-version työarkit (joita voi avata useita; kaikissa samaan Maple-istuntoon kuuluvissa työarkeissa ovat voimassa samat määritelmät ja asetukset) voidaan tallettaa mws-tiedostoiksi. Jos tällainen avataan myöhemmässä Maple-istunnossa, nähdään kaikki sellaisena kuin se oli viimeksi ennen talletusta. Yhtään komentoa ei kuitenkaan nyt ole suoritettu eikä määritelmää asetettu. Komennot voi tarvittaessa suorittaa yksitellen uudelleen.

Muistamme, että `read`-komennolla luetun, tosin aivan eri tyyppisen tiedoston, sisältämät komennot suoritetaan automaattisesti.

Työarkki graafisena oliona voidaan File-valikon Export As -alavalikosta käsin tallettaa myös LaTeX- ja HTML-muodossa. Niin ikään yksittäiset kuvat voidaan saattaa esim. ps- tai gif-muotoon, mistä hieman enemmän myöhemmin.

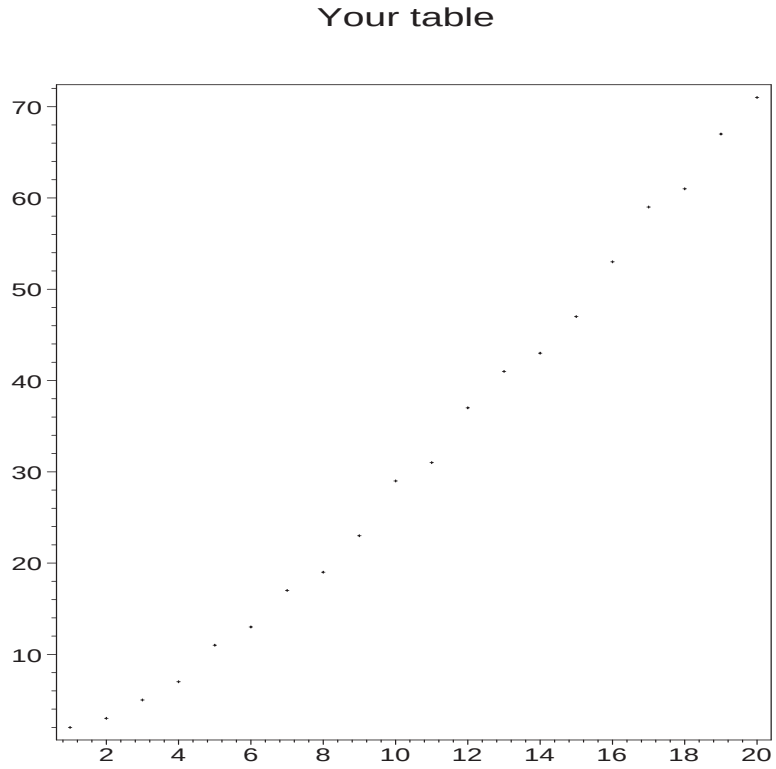
Laskutulosten dokumentoimiseksi ASCII-tiedostoon voidaan käyttää `save`-käskyä, kuten alla olevassa tiedostossa näytetään. Talletetut tulokset saadaan myöhemmin käyttöön `read`-käskyä käyttäen.

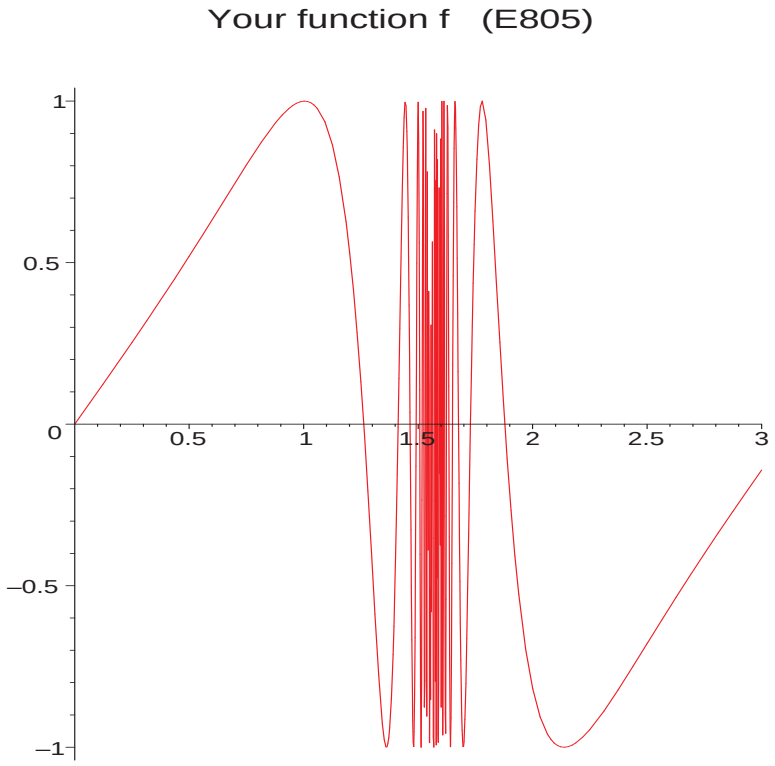
```
# FILE  e804.mpl  begins.

system("del e804");
system("del a:/maple/e804.txt");

#  Yo. ei ole tarpeellista ennen save-kaskya (joka ei ole append).

a:= [seq(ithprime(n), n=1..200)];
save a, "a:/maple/e804.txt";

#  A text file containing the definition (assigment statement)
#  of a is now on your diskette.
#  In a later session it can be read in (and performed) by
#  read "a:/maple/e804.txt";

save e804;

#  Ei toimi R5:ssa, vaikka helpin mukaan pitaisi toimia!
# FILE  e804.mpl  ends.
```

8.5. Input-käsky. Ohjelmoinnin yksi perusmekanismi on syötteen saaminen käyttäjältä. Maplessa on MATLAB- ja BASIC-kielten komentoa 'Input' vastaava komento `readstat`. Alla esimerkki sen käyttömahdollisuuksista.

```
# FILE  e805.mpl  begins.
# proc, readstat, nimettoman proseduurin suoritus, pointplot, eval

x:='x':                 # Plotting variable
for i to 2 do
    proc() local a,b,x;
        a:=readstat("Enter a number a in (0,1): ");
        b:=readstat("Enter a number b > a : ");
        plot(sin(x), x=a..b, axes=BOXED, title="sin(x)")
    end()
od;

#  Talla proseduurilla ei ollut nimea
#  eika myoskaan argumentteja (sulkujen valissa).
#  Proseduurin kutsu saatettiin suorittaa heti sen maaritelman peraan
#  (mista sulut end:n jalkeen)!
```

```
c:=readstat(
   "Enter n by 2 matrix \n (e.g., [seq([n, ithprime(n)], n=1..20)]; ): "
            );

#  Not a matrix technically...

with(plots):
fig805a:=pointplot(c,axes=BOXED,title="Your table"):
fig805a;

readstat("Define a function f, e.g., f:= x-> sin(tan(x)); : ");
f:=eval(f);
fig805b:=plot(f, 0..3, title="Your function f   (E805)"):
fig805b;

f:='f';  c:='c';
# FILE  e805.mpl  ends.
```





8.6. Funktion taulukointi. Funktioiden taulukointi ja kuvaajien piirtäminen on helppo tehdä seuraavan ohjelman avulla.

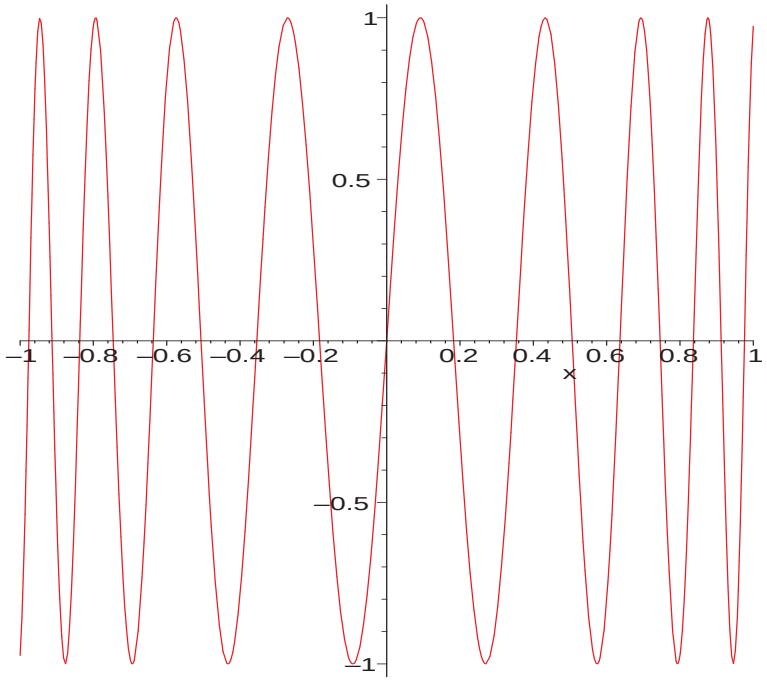
```
# FILE  e806.mpl  begins.
# series, lprint     !  Parempi tulos saataisiin  printf:aa  tms.
#                    !  kayttamalla.

x:='x':    # plotting variable

readstat("Define a function f on (-1,1), ".
         "e.g., f:= x-> sin(17*tan(x)); : "):
f:= eval(f);
print(`f(0.5) = `, f(0.5));
fig806:= plot(f(x), x=-1..1, title="Your function f  (E806)"):
fig806;
print(``);
lprint(`The series at x = 0 is : `);
a:= series(f(x), x=0,9):
print(a);
```

```
print(‘‘);
#print(‘        x              f(x)    \n‘);
dat:= lprint(‘ x         f(x)    ‘),
      seq(lprint(.100000*j, evalf(f(.1*j), 6)), j=1..10):
dat;
f:=’f’:
a:=’a’:
# FILE  e806.mpl  ends.
```

```
f := sin@(17 tan)
```

```
f(0.5) = , .1372014935
```

Your function f (E806)



The series at x = 0 is :

$$17\,x - \frac{4879}{6}\,x^3 + \frac{440623}{40}\,x^5 - \frac{313970467}{5040}\,x^7 + O(x^9)$$

```
 x          f(x)
.100000    .990914
.200000    -.299795
.300000    -.854442
.400000    .785989
.500000    .137214
.600000    -.805234
.700000    .983532
.800000    -.974763
.900000    .538325
1.000000    .974225
```

8.7. Paketteja. Maplessa on mukana useita standardipaketteja, jotka pitää erikseen lukea työtilaan, mikäli niitä tarvitaan (`with`). (Itse asiassa paketista voidaan myös lukea yksittäinen funktio tai viitata sellaiseen nimeä lataamatta.) Lisäksi on sellaisia kirjastofunktioita, jotka luetaan komennolla `readlib`.

Alla on komennon `?index,package` tuottama luettelo Maplen paketeista.

```
DEtools     differential equations tools
Domains     create domains of computation
GF          Galois Fields
GaussInt    Gaussian Integers
LREtools    manipulate linear recurrence relations
Matlab      Matlab Link
algcurves   Algebraic Curves
codegen     Code Generation
combinat    combinatorial functions
combstruct  combinatorial structures
diffalg     differential algebra
difforms    differential forms
finance     financial mathematics
genfunc     rational generating functions
geom3d      Euclidean three-dimensional geometry
geometry    Euclidean geometry
grobner     Grobner bases
group       permutation and finitely-presented groups
inttrans    integral transforms
liesymm     Lie symmetries
```

```
linalg      Linear algebra
logic       Boolean logic
networks    graph networks
numapprox   numerical approximation
numtheory   number theory
orthopoly   orthogonal polynomials
padic       p-adic numbers
plots       graphics package
plottools   basic graphical objects
powseries   formal power series
process     (Unix)-multi-processing
simplex     linear optimization
stats       statistics
student     student calculus
sumtools    indefinite and definite sums
tensor      tensor computations and General Relativity
totorder    total orders on names
```

Olemme jo nähneet, miten esimerkissä 8.2 käytettiin yhtä tällaista pakettia hyväksi funktioiden kuvaamisessa ja esimerkissä 8.3 ortogonaalipolynomien käyttöönotossa. Alla valaisemme ensin pakettien `DEtools` ja `linalg` sekä lopuksi Maplen MATLAB-linkin käyttöä. Viimeksi mainitussa voidaan paketin `Matlab` avulla käynnistää Maplesta käsin MATLAB, ajaa sillä ohjelmia ja välittää tietoa sen ja Maple-prosessin kesken (jos linkki on koneeseen asennettu)!

Olkoon annettu alkuarvotehtävä

$$X'(t) = \begin{bmatrix} 1 & 2 & -1 \\ 1 & 0 & 1 \\ 4 & -4 & 5 \end{bmatrix} X(t)\,, \qquad X(0) = \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}.$$

Ratkaisemme tämän ensin `DEtools`-paketin `matrixDE`-funktion avulla.

```
> restart;
> interface(echo=2);
> read "e807a.mpl";
# FILE  e807a.mpl  begins.
> with(DEtools):
> with(linalg):
Warning, new definition for adjoint
Warning, new definition for norm
Warning, new definition for trace
>
```

```
> A := matrix(3,3,[1,2,-1, 1,0,1, 4,-4,5]);
                                    [1     2    -1]
                                    [              ]
                               A := [1     0     1]
                                    [              ]
                                    [4    -4     5]

> B := matrix(3,1,[-1,0,0]);
                                         [-1]
                                         [  ]
                                     B := [ 0]
                                         [  ]
                                         [ 0]

> sol := matrixDE(A,t);
bytes used=1000168, alloc=851812, time=0.14
bytes used=2098172, alloc=1244956, time=0.32
                [ exp(t)         exp(2 t)        exp(3 t)  ]
                [                                          ]
       sol := [[ -exp(t)     - 1/2 exp(2 t)     -exp(3 t) ], [0, 0, 0]]
                [                                          ]
                [-2 exp(t)     -2 exp(2 t)      -4 exp(3 t)]

>
#  C := matrix(3,1);
#  Yleinen ratkaisu on nyt muotoa  X = sol[1]*C + sol[2].
#  Ratkaistaan  C  alkuehdosta ja sijoitetaan (tassa sol[2]=0):
>
> sol[1];
                  [ exp(t)         exp(2 t)        exp(3 t)  ]
                  [                                          ]
                  [ -exp(t)     - 1/2 exp(2 t)     -exp(3 t) ]
                  [                                          ]
                  [-2 exp(t)     -2 exp(2 t)      -4 exp(3 t)]

> eval(%,t=0);   # R4:ssa subs(t=0,%) samantapainen
                               [ 1     1       1]
                               [                ]
                               [-1    -1/2    -1]
                               [                ]
                               [-2     -2     -4]

> C:= evalm(inverse(%)&*B);
```

```
                                    [ 0]
                                    [  ]
                               C := [-2]
                                    [  ]
                                    [ 1]

> evalm(sol[1]&*C);
                           [-2 exp(2 t) + exp(3 t) ]
                           [                       ]
                           [  exp(2 t) - exp(3 t)  ]
                           [                       ]
                           [4 exp(2 t) - 4 exp(3 t)]

# FILE  e807a.mpl  ends.
```

(Asetimme Maplen käyttöliittymämuuttujan `echo` arvon sellaiseksi, että `read`-käskyä käytettäessä myös itse komennot, jotka tiedostosta luettiin, "kaiutettiin" näkyviin.)

Funktion `matrixDE` yms. laajoista käyttömahdollisuuksista katso tarkemmin helpistä. Ratkaisemme nyt saman alkuarvotehtävän ilman paketteja, funktion `dsolve` avulla.

```
> interface(echo=2);
> read "e807b.mpl";
# FILE  e807b.mpl  begins.
> yhtalot:= {diff(x(t),t) = x(t)  +2*y(t)-  z(t),
>            diff(y(t),t) = x(t)         +  z(t),
>            diff(z(t),t) = 4*x(t)-4*y(t)+5*z(t)};
           d                                  d
yhtalot := {-- x(t) = x(t) + 2 y(t) - z(t), -- y(t) = x(t) + z(t),
           dt                                 dt

    d
    -- z(t) = 4 x(t) - 4 y(t) + 5 z(t)}
    dt

> alkuehto:= {x(0)=-1, y(0)=0, z(0)=0};
                 alkuehto := {x(0) = -1, y(0) = 0, z(0) = 0}

> dsolve(yhtalot union alkuehto, {x(t),y(t),z(t)});
bytes used=1004652, alloc=786288, time=0.18
```

```
bytes used=2092088, alloc=1638100, time=0.54
{y(t) = -exp(3 t) + exp(2 t), z(t) = 4 exp(2 t) - 4 exp(3 t),

    x(t) = exp(3 t) - 2 exp(2 t)}

# FILE  e807b.mpl  ends.
```

Kuten huomataan, saimme saman (eksaktin) tuloksen kuin edellisellä menetelmällä.

Lopuksi katsomme lyhyesti kolmea asiaa yhdessä esimerkissä: MATLAB-linkin käyttoä, nopeaa Fourier-muunnosta ja erityisesti sen soveltamista numeerisessa data-analyysissä, tarkemmin frekvenssianalyysissä. Maplessakin on `FFT`-funktio, mutta demonstroimme tässä MATLAB-linkin kautta käyttöön saatavan `fft`:n (nimi Maplen sisällä) periaatteessa tehokkaampaa numeerista toimintaa.

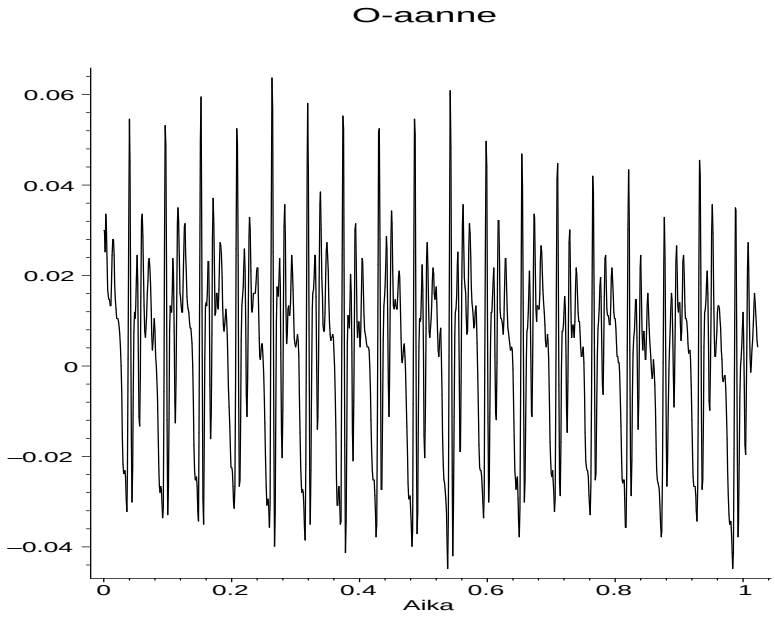
Olkoon annettu tiedosto oaanne.dat, joka sisältää mikrofoniin lausutusta O-äänteestä digitalisoitua dataa:

```
 0.03
 0.03
 2.520E-02
 3.360E-02
    .
    .
    .
 5.600E-03
 4.200E-03
```

Tehtävänä on selvittää, mitä osaääneksiä ko. ääneen sisältyy: mitä frekvenssejä milläkin painolla. Aineisto ladataan Mapleen `fscanf`-funktiolla "hardware float" -muotoon, jollaista MATLABissa käytetään. (Tässä on huomattava, että — helpin antamaa informaatiota ajatellen hieman yllättävästi — `fscanf` tulkitsee tiedostossa olevat '+'-merkit lopetusmerkeiksi, joten riveistä `0.000E+00` on ensin poistettava nuo plussat.) Nopeasta Fourier-muunnoksesta saadaan kompleksinen vektori Y, jolle $(\bar{Y}_i Y_i)_{i=1}^{1024}$ kuvaa energian jakautumista alkuperäisen datan eri frekvensseille.

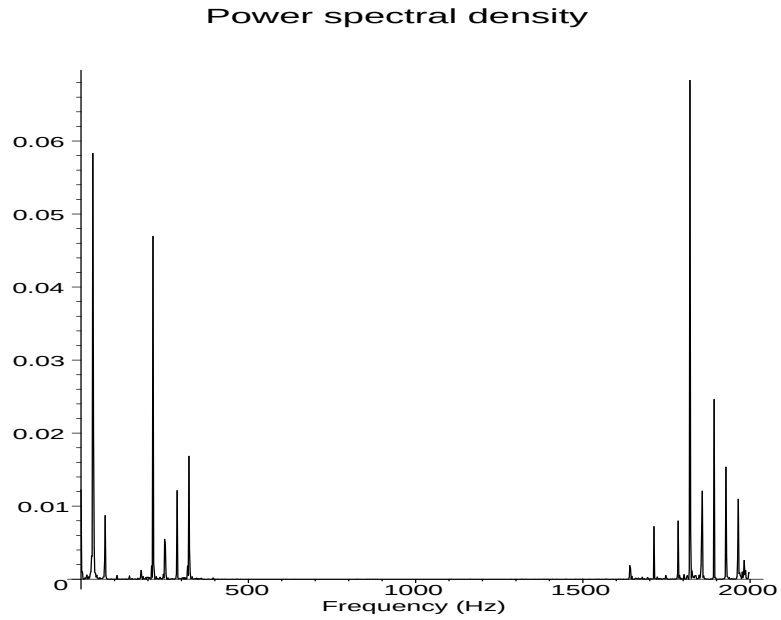
Tehtävä oli alun perin ratkaistu MATLABilla. M-tiedoston aani.m pohjalle on rakennettu Maple-komentoja sisältävä tiedosto aani.mpl, johon on jätetty alkuperäisiä MATLAB-komentoja "kommenteiksi". Asiaa ei pidä sekoittaa siihen, että aani.mpl:a Maplessa ajettaessa osa tietojenkäsittelystä itse asiassa tapahtuu MATLABissa.

```
> interface(echo=2);
> read "aani_a.mpl";
# FILE  aani_a.mpl  begins.
#       Tassa esimerkissa kaytetaan Maplen Matlab-linkin kautta
#       MATLABin fft-funktiota spektraalianalyysiin.
#       Aineistona on Fysiikan laitoksella aanitetty O-aanne
#       digitaaliseen muotoon saatettuna.

#       A common use of FFT's is to find the frequency
#       components of a signal buried in a noisy time domain signal.

#  Kaynnistetaan Matlab-linkki ja ladataan tarpeellisia paketteja:
>
> restart:
> with(Matlab):
> with(plots):
>
#load oaanne.dat
#y = oaanne;
> data:=fscanf("oaanne.dat",%hf):
> fclose("oaanne.dat"):
>
> t:= [seq(i*.001,i=0..1023)]:
>

#pituus= length(y);
#clf
#       Let's take a look at our noisy signal y by plotting it.

#plot(y(1:pituus)), title('O-aanne'), xlabel('Aika'), pause, clf
> pointplot([seq([t[i],op(data)[i]],i=1..nops(t))],style=LINE,
>           axes=FRAME, title="O-aanne", labels=["Aika"," "]);
```




```
>

#       Clearly, it is difficult to identify the frequency components
#       from looking at the original signal; that's why spectral analysis
#       is so popular.

#       Finding the discrete Fourier transform of the noisy signal y
#       is easy; we just take the fast-Fourier transform (FFT) :

#Y = fft(y,pituus);
> Y:=fft(op(data)):
>
> real_Y:= op(1,Y)[1]:
> complex_Y:= op(2,Y)[1]:
>

#       The power spectral density, a measurement of the energy at
#       various frequencies, is found with:

#Pyy = Y.*conj(Y)/pituus;
> Pyy:= [seq((real_Y[i] + complex_Y[i])*
>             conjugate(real_Y[i] + complex_Y[i]) / 1024, i=1..1024)]:
bytes used=1000140, alloc=851812, time=0.27
>

#       To plot the power spectral density, we must first form a
```

```
#       frequency axis:

#f = 4000/(2*pituus)*(0:pituus);
> freq:= [seq(4000*i/2048., i=0..1023)]:
>

#       We can now plot the power spectral density:

#plot(f(1:pituus-1),Pyy(1:pituus-1)'), ...
#     title('Power spectral density'), ...
#     xlabel('Frequency (Hz)')
> pointplot([seq([freq[i],Pyy[i]],i=1..1024)],style=LINE,
>       title="Power spectral density",
>       labels=["Frequency (Hz)"," "]);
```

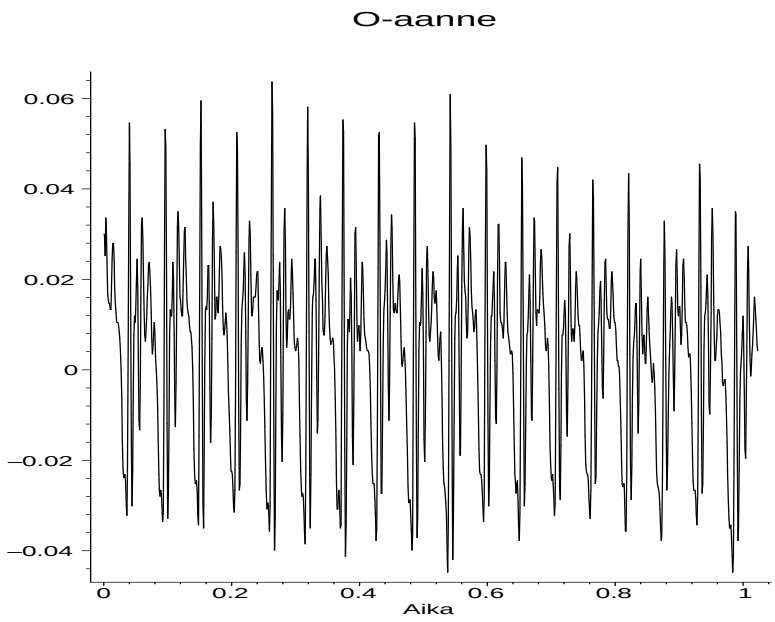



```
# FILE  aani_a.mpl  ends.
```

Tehdään sama uudestaan Maplesta käsin, mutta käyttäen lähes kokonaan MATLABin syntaksia. Kuva energiajakaumasta poikkeaa jonkin verran äsken saadusta:

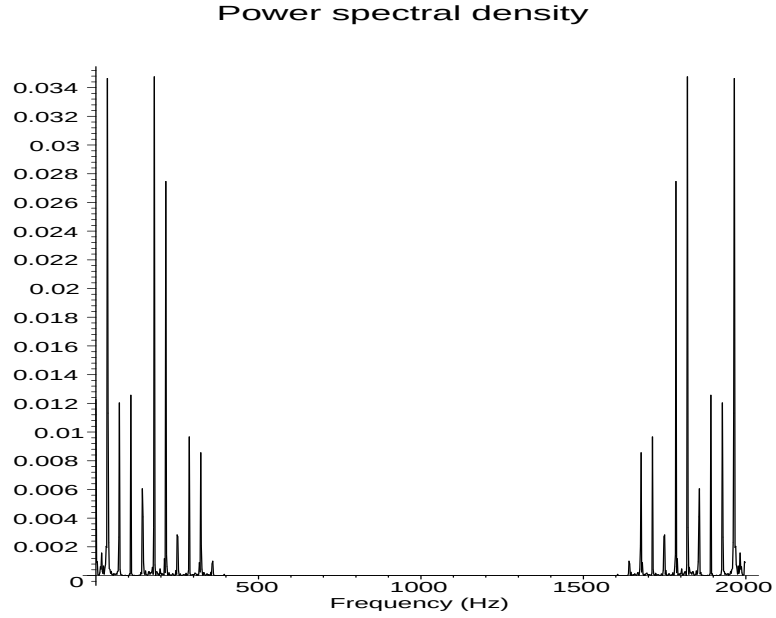
```
> interface(echo=2);
> read "aani_b.mpl";
# FILE  aani_b.mpl  begins.
> restart:
> with(Matlab):
```

```
> with(plots):
>
> evalM("load oaanne.dat");
> evalM("t=0:0.001:1.023");
> t:=convert(getvar("t"),array):
> data:=convert(getvar("oaanne"),array):
>
> pointplot([seq([t[1,i],op(data)[i]],i=1..op(2,size(t)))],style=LINE,
>           axes=FRAME, title="O-aanne", labels=["Aika"," "]);
```




```
>
> evalM("Y=fft(oaanne,1024)");
>
> evalM("Pyy=Y.*conj(Y)/1024");
> Pyy:=convert(getvar("Pyy"),array):
>
> evalM("f=4000*(0:1023)/2048");
> freq:=convert(getvar("f"),array):
>
> size(data);
bytes used=2053356, alloc=1507052, time=0.64
                                  [1, 1024]

> size(freq);
                                  [1, 1024]
```

```
> size(Pyy);
                                  [1, 1024]

>
> pointplot([seq([freq[1,i],Pyy[i]],i=1..1024)],style=LINE,
>       title="Power spectral density",
>       labels=["Frequency (Hz)"," "]);
```




```
# FILE  aani_b.mpl  ends.
```

8.8. Pns-sovitus. Polynomimallin sovitus pisteistöön on usein esiintyvä ongelma. Interaktiivisessa Maple-työskentelyssä sovitus voidaan tehdä esimerkiksi funktion `fit` avulla seuraavasti.

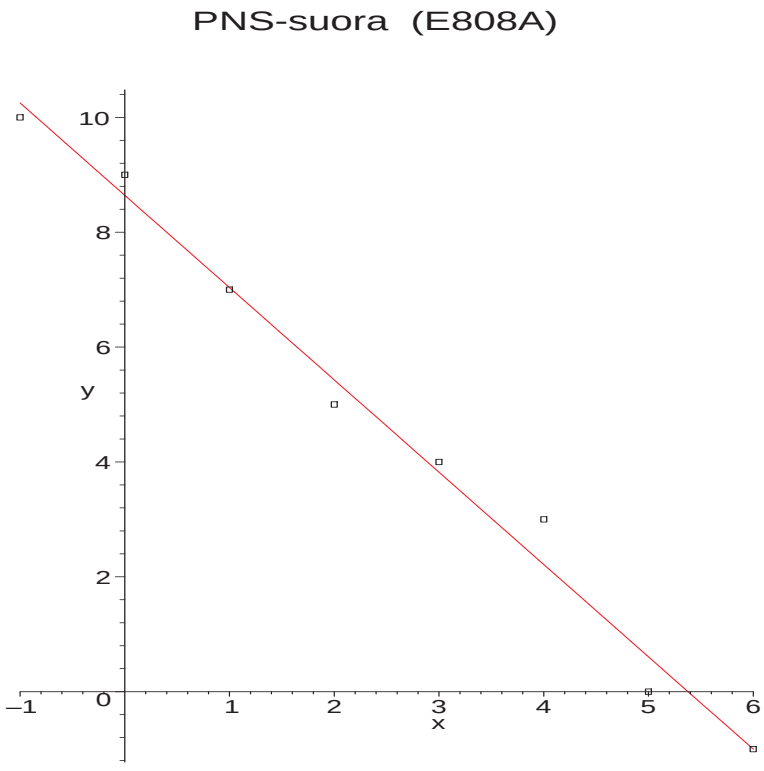
```
> dt:= [seq( sum(p*x^p, p=0..4), x=1..10)];
       dt := [10, 98, 426, 1252, 2930, 5910, 10738, 18056, 28602, 43210]

> with(stats);
[anova, describe, fit, importdata, random, statevalf, statplots, transform]

> a,b,c,d,e;
                                  a, b, c, d, e

> fit[leastsquare[[x,y], y=a*x^4+b*x^3+c*x^2+d*x+e,{a,b,c,d,e}]]
```

```
([[$1..10],dt]);
```

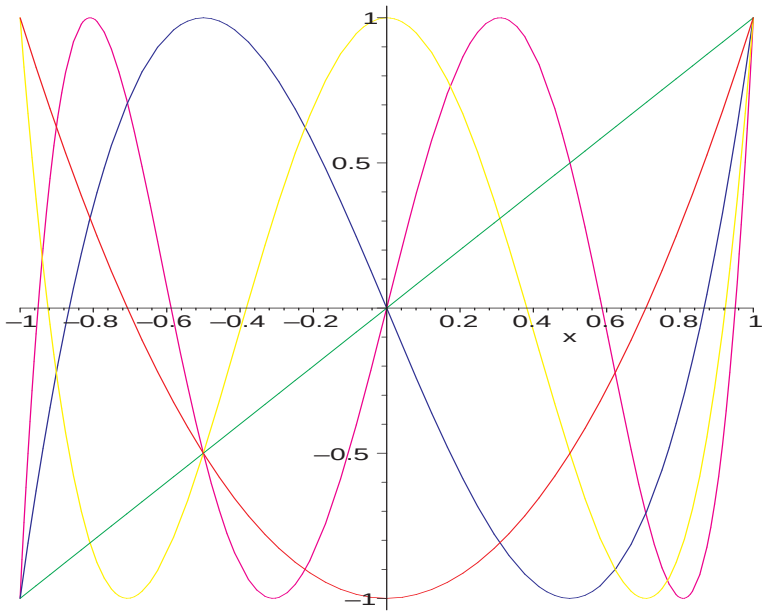
$$y = 4\,x^4 + 3\,x^3 + 2\,x^2 + x$$

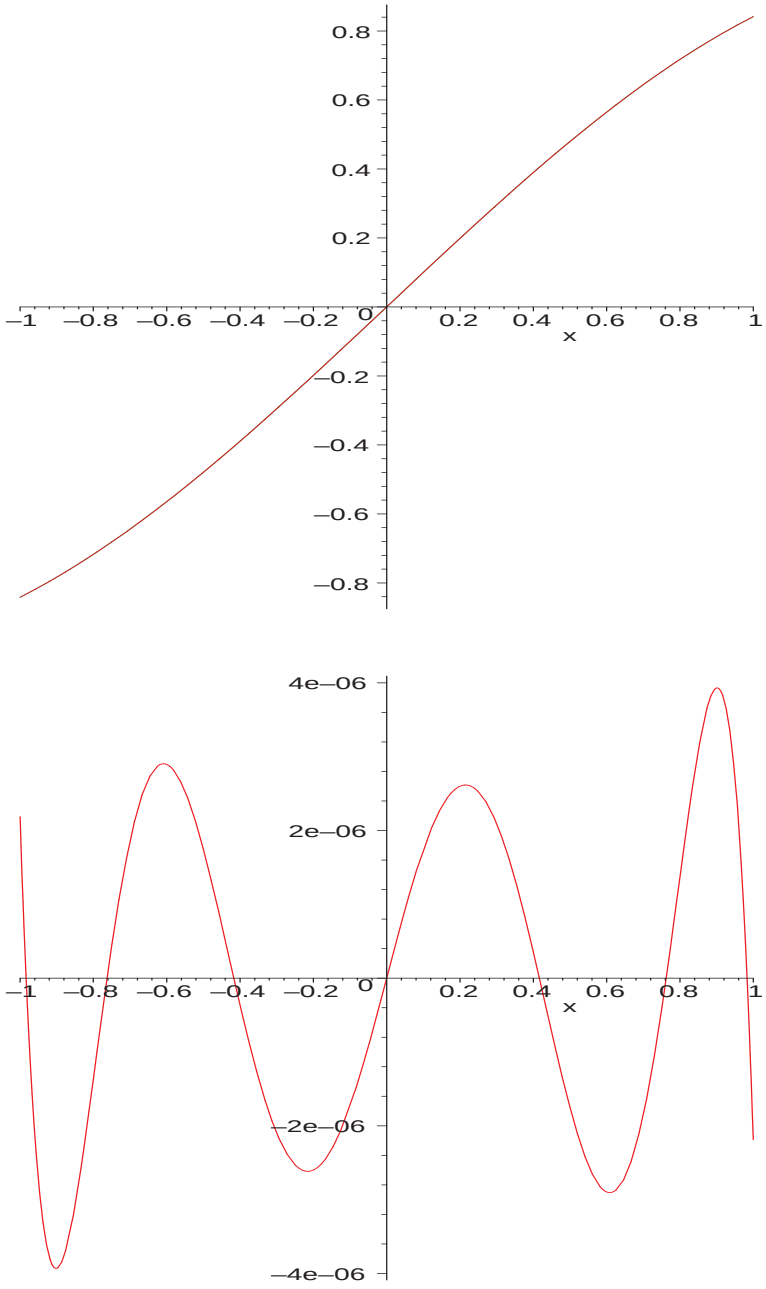
Näin ollen saatiin odotettu tulos. Alla vielä kaksi samaan aihepiiriin liittyvää sovitustehtävää tiedostoina.

```
# FILE  e808a.mpl  begins.
with(stats):
with(plots):
xy := [[-1,10],[0,9],[1,7],[2,5],[3,4],[4,3],[5,0],[6,-1]];
x:='x':  y:='y':
lineq := fit[leastsquare[ [x,y] ]]([map(a->a[1],xy),
                                    map(a->a[2],xy)]);
dots := pointplot(xy,symbol=BOX):
gr := plot(rhs(lineq), x=-1..6):
print(`The least squares line: `,lineq);
fig808a:=display([gr, dots], labels = ["x","y"],
                             title = " PNS-suora  (E808A)"):
fig808a;
# FILE  e808a.mpl  ends.
```



```
# FILE  e808b.mpl  begins.
x:='x':  y:='y':
with(stats):
with(orthopoly):
plot({seq(T(n,x),n=1..5)}, x=-1..1);
Xvalues:=[seq(k/10, k=-10..10)]:
Yvalues:=[seq(evalf(sin(k/10)), k=-10..10)]:

appr_eq := fit[leastsquare[ [x,y], y=sum(a[n]*T(n,x), n=1..5)]]
              ([Xvalues,Yvalues]);
fig808b := plot({sin(x), rhs(appr_eq)}, x=-1..1):
fig808b;
plot(sin(x) - rhs(appr_eq), x=-1..1);
# FILE  e808b.mpl  ends.
```

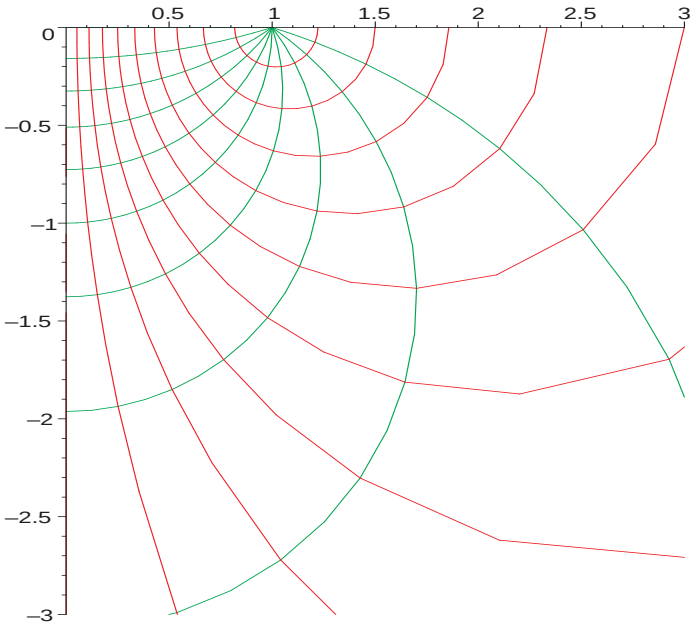


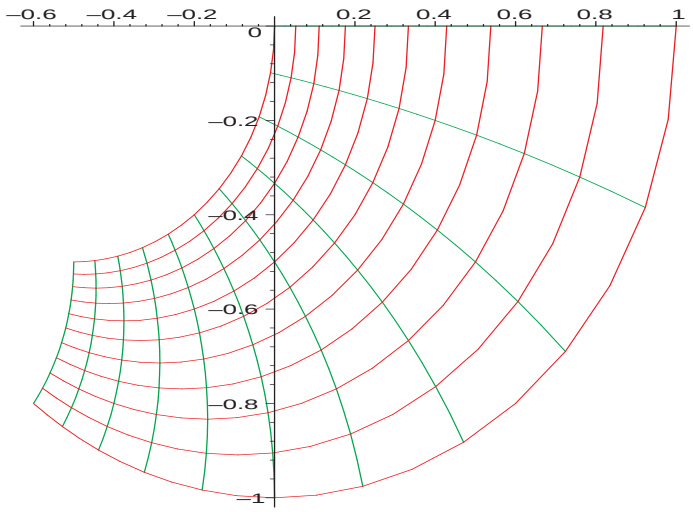


8.9. Konformikuvaukset. Komennolla `with(plots)` luetaan työtilaan mm. funktio `conformal`, jonka avulla voidaan piirtää koordinaatistoverkon kuvautuminen konformikuvauksessa. Voidaan asettaa

```
# FILE  e809.mpl  begins.
with(plots):
```

```
g:= z->(1-z)/(1+z);
fig809a:= conformal(g,0..1+2*I):
fig809a;
fig809b:= conformal(g@( Re*(exp@(I*Im)) ), 0..1+Pi*I,
                    0..3-3*I):  # view adjusted by hand
fig809b;
# FILE  e809.mpl  ends.
```





8.10. Käänteisfunktion arvon laskeminen. Seuraava ohjelma laskee funktion käänteisfunktion arvon annetussa pisteessä.

```
# FILE  invfnc.mpl  begins.

#  invfnc(f0,a0,b0,y) gives an inverse x of f0 at y,
#     for which a0<x<b0:

invfnc:=proc(f0,a0,b0,y)
local f,a,b,c,fa,fb,fc, k,tst,MaxIter,NonBracket;
   if type(evalf(y),numeric) then
      f:= f0-y;
      a:= evalf(a0);
      b:= evalf(b0);
      if b<a then c:=b; b:=a; b:=c fi;
      k:= 0;
      fa:= evalf(f(a));
      fb:= evalf(f(b));
      tst:= false;
      MaxIter:= 45;             #  2^{-45} \approx 3e-14
      if fa*fb>0 then do
         print("NonBracketing error in invfnc, a = ",a,
               ", b = ",b);
         RETURN(NonBracket)
         od
      fi;
      while tst=false and k<MaxIter do
         c:= (a+b)/2.;
         fc:= evalf(f(c));
         if fc=0 then a:=c; b:=c; tst:=true fi;
         if fb*fc>0 then b:=c; fb:=fc else a:=c; fa:=fc fi;
         k:= k+1
      od;
      RETURN(evalf(c))
   else 'invfnc'(args)   # To enable plotting normally
   fi
end;


# invfnc2 does the same, using the built-in fsolve;
#    the extra argument controlling the precision:
```

```
invfnc2:=proc(f0,a0,b0,y,d)
   if nargs>4 then Digits:=d fi;
   if type(evalf(y),numeric) then fsolve(f0(x)=y,x,a0..b0)
   else 'invfnc2'(args)
   fi
end;

# FILE  invfnc.mpl  ends.
```

Lasketaan esimerkiksi polynomin $x^2 - 2$ nollakohta välillä $(0, 2)$.

```
> read "invfnc.mpl":
> invfnc(x->x^2-2,0,2,0);
                                  1.414213563

> invfnc2(x->x^2-2,0,2,0);
                                  1.414213562
```

Tässä tapauksessa valmis `fsolve` oli tarkempi, kuten voidaan osoittaa.

8.11. Lineaarialgebraa. Komennolla `with(linalg)` saadaan käyttöön mm. Gramin–Schmidtin ortogonalisointi, jota käytetään seuraavasti.

```
> with(linalg,GramSchmidt):   # tai yleisemmin with(linalg):
> GramSchmidt({vector([3,4,2]), vector([2,5,2]), vector([1,2,6])});

                                -32  25  -2    -24  -24  84
                {[3, 4, 2], [---, --, --], [---, ---, --]}
                                 29   29  29    19   19   19
```

8.12. Kuvan tallettaminen PostScript-tiedostoon. Maplen kuvat voidaan tallettaa esimerkiksi seuraavasti:

```
> plot(sin@(17*tan),-1..1);
                           (Kuva leikattu pois.)
> kuva:=%:    #  Huom. kaksoispiste tassa!
> plotsetup(ps, plotoutput="kuva.ps",
            plotoptions="portrait,noborder");
> kuva;       #  Huom. puolipiste tassa!
```

Tämä tekee PostScript-tiedoston kuva.ps oletushakemistoon. Nyt kuitenkin myös kaikki seuraavat plottaukset ohjautuvat tuohon samaan tiedostoon (eivätkä näy ruudulla millään tavoin). Tilanteen palauttamiseksi normaaliksi voidaan antaa komento

```
> plotsetup(default);
```

jonka pitäisi palauttaa oletusarvot. Joissakin Maplen versioissa tämä ei toimi aivan oikein; komennolla `plotsetup();` nähdään, mitkä asetukset ovat voimassa, ja niitä voi tarvittaessa säätää lisää "käsin".

9 Tavalliset differentiaaliyhtälöt

Differentiaaliyhtälöitä esiintyy monenlaisia luonnontieteellisiä ongelmia mallinnettaessa:

- mekaniikka (esim. liikeyhtälöt);
- kemialliset reaktiot;
- populaatioiden kehitys (esim. flunssaepidemian leviäminen).

Lyhenteitä: DY = differentiaaliyhtälö,
ODY = osittaisdifferentiaaliyhtälö,
AAT = alkuarvotehtävä,
RAT = reuna-arvotehtävä.

9.1. Malliongelma. Etsi jatkuvasti derivoituva funktio $y(t)$, joka toteuttaa AAT:n

$$y'(t) = f(t, y(t)) \,; \quad y(t_0) = y_0 \,, \tag{9.2}$$

missä $f\colon \mathbb{R}^2 \to \mathbb{R}$ on jatkuva.

Ratkaisu. $y(t) = y_0 + \int_{t_0}^{t} f(s, y(s))\, ds$.

Koska ratkaisussa esiintyvää määrättyä integraalia ei yleensä voida ilmaista analyyttisellä lausekkeella edes kaikille säännöllisille funktioille f, on selvää, että numeerisia menetelmiä tarvitaan DY:iden käytännön sovelluksissa.

9.3. Esimerkki. Newtonin jäähtymislain mukaan kappaleen lämpötila $T(t)$ toteuttaa DY:n (missä T_∞ = ympäristön lämpötila)

$$T'(t) = -k(T(t) - T_\infty) \,.$$

Nyt

$$\text{DY} \;\Leftrightarrow\; \frac{T'(t)}{T(t) - T_\infty} = -k \;\Leftrightarrow\; \log|T(t) - T_\infty| = -kt + C_1$$

$$\Leftrightarrow\; T(t) = T_\infty \pm e^{C_1} e^{-kt} \;\Leftrightarrow\; T(t) = T_\infty + C_2 e^{-kt} \,.$$

AAT: $T(t_0) = T_0 \;\Rightarrow\; C_2 = (T_0 - T_\infty)e^{kt_0}$. Siis

$$T(t) = T_\infty + (T_0 - T_\infty)e^{-k(t-t_0)} \,.$$

Maplen avulla ratkaisu voidaan tehdä esim. seuraavalla kahdella tavalla:

(a) Analyyttinen ratkaisu:

```
> dsolve({diff(y(t),t) = -k*(y(t) - 20), y(0) = 95}, y(t));

                        y(t) = 20 + 75 exp(-k t)
```

(b) Numeerinen ratkaisu:

```
>  y:='y': t:='t':
>  dsolve( {diff(y(t),t) = sin(t)*y(t), y(0) = 1}, y(t),
            numeric, startinit = true );
>  plots[odeplot](%, 0..20);
```

Tämä tapaus voidaan tosin ratkaista myös eksaktisti:

```
>  dsolve( {diff(y(t),t) = sin(t)*y(t), y(0) = 1}, y(t));

                                 exp(-cos(t))
                        y(t) = ------------------
                               cosh(1) - sinh(1)
```

AAT:n (9.2) ohella myös yleisempi tehtävä, missä $y = (y_1, \ldots, y_n)$ ja $f = (f_1, \ldots, f_n)$, on tärkeä, sillä korkeamman kertaluvun differentiaaliyhtälöt palautuvat tähän: Tarkastellaan yhtälöä

$$(*) \qquad y^{(n)} = f(t, y, y', y'', \ldots, y^{(n-1)})\,.$$

Jos tässä merkitään $y_1(t) = y(t)$ ja vastaavasti $y_2 = y'$, $\ldots$, $y_n = y^{(n-1)}$, niin $(*)$ voidaan kirjoittaa muotoon

$$\begin{cases} y_1' = y_2 \\ y_2' = y_3 \\ \vdots \\ y_{n-1}' = y_n \\ y_n' = f(t, y_1, y_2, \ldots, y_n) \end{cases} \qquad \text{eli} \quad \overline{y}' = \overline{f}(t, \overline{y})\,, \quad \text{missä } \overline{y} = (y_1, \ldots, y_n).$$

Differentiaaliyhtälöiden teorian päätehtäviä ovat olemassaolo- ja yksikäsitteisyyskysymysten selvittäminen sekä yhtälötyyppien luokittelu. Mm. toisen kertaluvun vakiokertoimisen DY:n

$$y'' + ay' + by = 0$$

ratkaiseminen on esitetty [LP]:ssä. MATLAB-ratkaisu oli esillä kohdassa 1.5.

Ennen numeerisen ratkaisun aloittamista tulisi varmistua, että ratkaisu on olemassa ja yksikäsitteinen. Vaikka näin tehtäisiin, voi algoritmin numeerisessa toiminnassa silti ilmetä vaikeuksia mm. seuraavista syistä:

- diskretointiaskeleen pituuden valinta voi olla vaikeaa;
- epästabiilit probleemat (pieni muutos AAT:ssä $\Rightarrow$ iso muutos ratkaisussa).

9.4. Esimerkki. Kettujen ja jänisten lukumäärät $k(t)$ ja $j(t)$ riippuvat toisistaan yhtälöryhmän

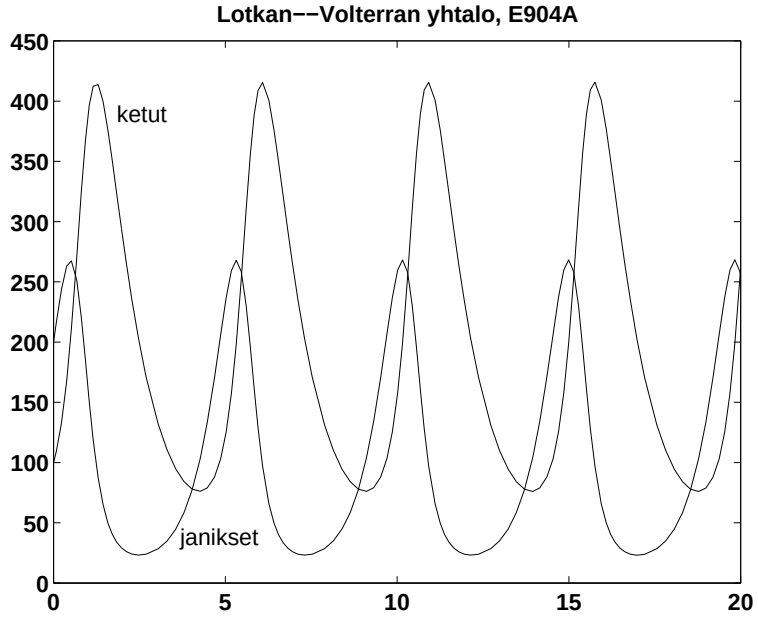
$$\begin{cases} j'(t) = 2j(t) - \alpha j(t)k(t) \\ k'(t) = -k(t) + \alpha j(t)k(t) \end{cases}$$

mukaisesti. Tämä on esimerkki ns. Lotkan–Volterran differentiaaliyhtälöistä. Numeerinen ratkaisu voidaan tehdä mm. seuraavin eri tavoin.

(a) Luodaan MATLABissa am. tiedostot e904dy.m ja e904a.m, ja annetaan käsky `e904a`.

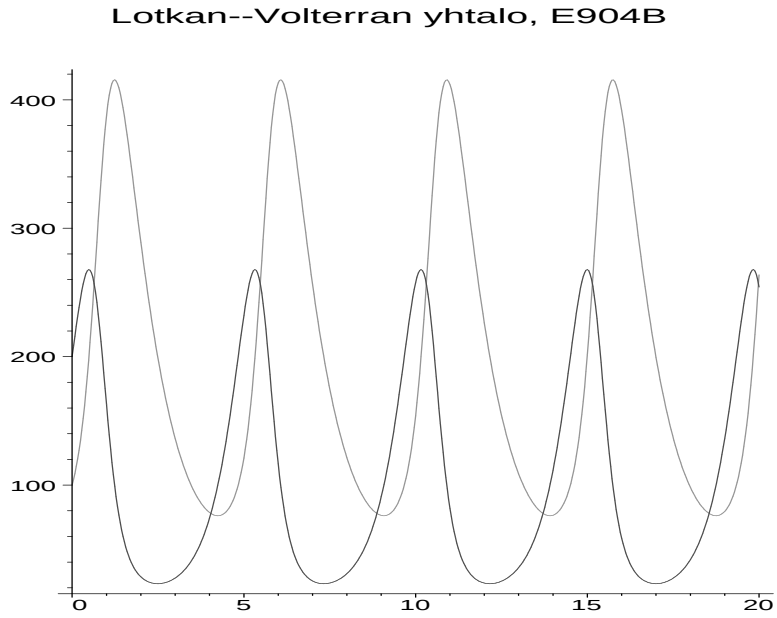
```
% FILE  e904dy.m  begins.
% dj/dt = a(1)*j(t) + a(3)*j(t)*k(t);
% dk/dt = a(2)*k(t) + a(4)*j(t)*k(t);
function z = e904dy(t,y)
j=y(1); k = y(2);
z= zeros(2,1);  a= zeros(1,4);
a(1) =2; a(2) = -1;
a(3) = -0.01; a(4) = 0.01;
z= [(a(1)*j+ a(3)*j*k), (a(2)*k + a(4)*j*k)]';
% FILE  e904dy.m  ends.
```

```
% FILE  e904a.m  begins.
[t,y] = ode23('e904dy', [0 20], [200 100]');
plot(t,y), title('Lotkan--Volterran yhtalo, E904A');
gtext('ketut'); gtext('janikset');
% FILE  e904a.m  ends.
```

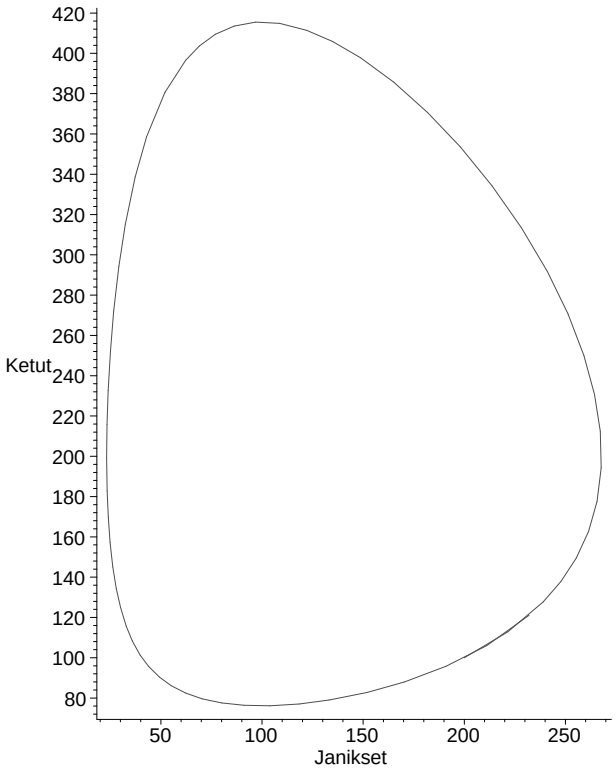


(b) Maple-ratkaisu puolestaan saadaan lukemalla sisään tiedosto e904b.mpl, jonka sisältö näytetään alla kaiuttamalla se kuvaruudulle.

```
> interface(echo=2);
> read "e904b.mpl";
# FILE  e904b.mpl  begins.
> sol:= dsolve( {diff(j(t),t) = 2*j(t) - 0.01*j(t)*k(t),
>                diff(k(t),t) =  -k(t) + 0.01*j(t)*k(t),
>                j(0) = 200, k(0) = 100},
>               [j(t), k(t)], numeric, startinit = true );

                    sol := proc(rkf45_x)  ...  end

> sol(0); sol(1);
                   [t = 0, j(t) = 200., k(t) = 100.]

      [t = 1, j(t) = 162.6842642344886, k(t) = 387.6534376198975]

> with(plots):
> odeplot(sol, [[t,j(t)], [t,k(t)]], 0..20, numpoints = 500,
>         title = "Lotkan--Volterran yhtalo, E904B");
```




```
> J:= t -> rhs( op(2,sol(t)) ):
> K:= t -> rhs( op(3,sol(t)) ):
> plot([J,K,0..5],                    # Ks.  ?plot[parametric]
>      labels = ["Janikset","Ketut"]);
```



```
# FILE  e904b.mpl  ends.
```

Alempi kuva esittää kettujen ja jänisten määrien keskinäistä riippuvuutta.

9.5. AAT:n (9.2) diskretointi. Halutaan löytää (9.2):n ratkaisu arvolla $t = T$. Valitaan $t_0 < t_1 < t_2 < \cdots < t_n < t_{n+1} = T$ ja etsitään ratkaisun arvo y_j näissä pisteissä. Yleisesti y_j riippuu arvoista $y_{j-1}, \ldots, y_{j-k}$. Jos $k = 1$, on kyseessä *yksiaskel*-menetelmä, jos $k > 1$, *moniaskel*-menetelmä.

Yksiaskelmenetelmä voidaan esittää joko muodossa

$$y_{n+1} = y_n + \Phi_1(t_n, y_n, h)\,, \tag{9.6}$$

jolloin kyseessä on *eksplisiittinen* menetelmä, tai muodossa

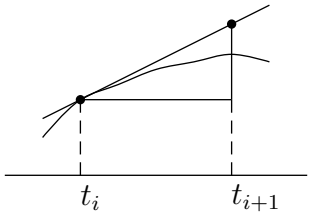
$$y_{n+1} = y_n + \Phi_2(t_n, y_n, y_{n+1}, h)\,, \tag{9.7}$$

jolloin menetelmä on *implisiittinen*. Tässä Φ_j on ns. *lisäysfunktio*.

Numeerisia ratkaisumenetelmiä (9.2):lle:

Eulerin menetelmä. Tasavälinen pisteistö: $h = t_{i+1} - t_i$,

$$y_{i+1} = y_i + hf(t_i, y_i)\,. \tag{9.8}$$



Runge–Kutta -menetelmät.

(a) Modifioitu Euler:

$$y_{i+1} = y_i + hf(t_i + \frac{h}{2}, y_i + \frac{h}{2}f(t_i, y_i))\,.$$

(b) Rungen–Kuttan 4. asteen menetelmä:

$$k_1 = f(t_i, y_i)\,, \quad k_2 = f(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_1)\,,$$

$$k_3 = f(t_i + \frac{h}{2}, y_i + \frac{h}{2}k_2)\,, \quad k_4 = f(t_{i+1}, y_i + hk_3)\,;$$

$$y_{i+1} = y_i + \frac{1}{6}h(k_1 + 2k_2 + 2k_3 + k_4)\,.$$

Käytännössä yo. menetelmiin on usein liitetty *askelpituuden adaptiivinen säätö* laskennan aikana kerätyn informaation avulla. Esim. NR:n ODEINT perustuu 4. asteen Runge–Kutta -menetelmään adaptiivisella askelpituuden säädöllä.

MATLABissa on ODE23 ja ODE45, jotka samaten ratkaisevat differentiaaliyhtälöryhmiä Runge–Kutta -menetelmällä. Maplen `dsolve`:ssa on oletusarvona Runge–Kutta -menetelmä (mutta erinäisiä vaihtoehtojakin on tarjolla).

9.9. Esimerkki. Eulerin menetelmä Maplessa.

```
# FILE  e909.mpl  begins.
  f:= (t,y) -> 1+y^2:
  eta:= 0:
  a:= 0.: b:= evalf(Pi/4):
  n:= 4:
  h:= (b-a)/n:  t:= a:  y:= eta:

  printf("\n    n = %d\n", n);
  printf("    h = %10.8f\n\n", h);
  printf("          t             y\n");
  printf("    %10.8f   %10.8f\n",t,y);

  for i to n do
     y:= y + h*f(t,y);
     t:= t + h;
     printf("    %10.8f   %10.8f\n",t,y)
  od:
# FILE  e909.mpl  ends.
```

Komennon `read "e909.mpl"` tuloste on seuraava:

```
    n = 4
    h =  .19634954

         t            y
    0.00000000   0.00000000
     .19634954    .19634954
     .39269908    .40026897
     .58904862    .62807671
     .78539816    .90188228
```

9.10. Esimerkki. (1) Epästabiili ongelma [KMN, s. 282]: $y'(t) = -10(t-1)y(t)$. Ratkaisu on $y = c\exp(-5(t-1)^2)$. Tarkastellaan kahta AAT:ää:

$$\begin{aligned} y_0 = 0.01 \;&\Rightarrow\; y(1) \approx 1.5\,; \\ y_0 = 0.1 \;&\Rightarrow\; y(1) \approx 15\,. \end{aligned}$$

(2) DY:n $y'(t) = y(t) - 100e^{-100t}$ ratkaisu on $y(t) = Ce^t + \frac{100}{101}e^{-100t}$. Alkuehdolla $y(0) = \frac{100}{101}$ on ratkaisu $y_0(t) = \frac{100}{101}e^{-100t}$. Alkuehdolla $y(0) = \frac{100}{101} + \delta$ $(\delta > 0)$ on ratkaisu $y_\delta(t) = \delta e^t + \frac{100}{101}e^{-100t}$. Nyt $|y_0(t) - y_\delta(t)| = \delta e^t$, jolloin ratkaisut loittonevat eksponentiaalisesti, kun $t \to \infty$. Näin ollen tehtävä on *häiriöaltis*.

(3) AAT: $y'(t) = e^{-t} - y\,;\ y(0) = y_0$. Ratkaisu on $y(t) = (t + y_0)e^{-t}$. Alkuehdolla $y(0) = y_0 + \delta$ ratkaisu $y_\delta(t) = (t + y_0 + \delta)e^{-t}$. Siis

$$|y(t) - y_\delta(t)| = |\delta|e^{-t} \to 0\,, \quad \text{kun } t \to \infty\,.$$

Tämä tehtävä on stabiili.

Jos em. numeerisia ratkaisumenetelmiä sovelletaan epästabiiliin ongelmaan, voidaan saada epäluotettavia tuloksia. Seuraava differentiaaliyhtälöiden kurssilla esitettävä lause antaa yhden riittävän ehdon ratkaisun olemassaololle ja yksikäsitteisyydelle. Lisäksi nähdään, että lauseen oletuksin AAT:n (9.2) ratkaiseminen on tietyssä mielessä stabiili ongelma.

9.11. Lause. Olkoon f jatkuva nauhassa $S = \{\,(t,y) : a \le t \le b,\ y \in \mathbb{R}\,\}$, ja olkoon olemassa vakio L s.e.

$$|f(t,y_1) - f(t,y_2)| \le L|y_1 - y_2|$$

aina, kun $t \in [a,b]$ ja $y_1, y_2 \in \mathbb{R}$. [Viimeksi mainittu ehto toteutuu esim., jos $\frac{\partial f}{\partial y}$ on olemassa ja rajoitettu nauhassa S.] Tällöin: jos $(t_0, y_0) \in S$, on AAT:llä (9.2) yksikäsitteinen ratkaisu y välillä $[a,b]$. Lisäksi: jos y_δ on alkuehtoa $y(t_0) = y_0 + \delta$ vastaava (9.2):n ratkaisu, on kaikille $t \in [a,b]$ voimassa epäyhtälö

$$|y(t) - y_\delta(t)| \le |\delta|e^{L|t-t_0|}\,.$$

9.12. Differenssimenetelmä. Tarkastelemme seuraavanlaista reunaarvotehtävää (RAT):

(9.13) $\quad x''(t) = p(t)x'(t) + q(t)x(t) + r(t) \quad (a < t < b);\ x(a) = \alpha,\ x(b) = \beta.$

Merkitään $h = (b-a)/N$ ja $t_j = a + jh$, kun $j = 0, 1, \ldots, N$. Derivaatoille käytetään approksimaatioita

$$\text{(9.14)} \qquad \begin{cases} x'(t_j) = \dfrac{x(t_{j+1}) - x(t_{j-1})}{2h} + O(h^2) \\ \\ x''(t_j) = \dfrac{x(t_{j+1}) - 2x(t_j) + x(t_{j-1})}{h^2} + O(h) \end{cases} .$$

Perustelemme x'':n lausekkeen (x':n lauseke samoin): Taylorin kaavasta saadaan

$$x(t+h) = x(t) + hx'(t) + \frac{h^2}{2}x''(t) + O(h^3)$$

$$x(t-h) = x(t) - hx'(t) + \frac{h^2}{2}x''(t) + O(h^3)$$

ja laskemalla puolittain yhteen

$$x(t+h) + x(t-h) = 2x(t) + h^2x''(t) + O(h^3)\,,$$

mistä

$$x''(t) = \frac{x(t+h) + x(t-h) - 2x(t)}{h^2} + O(h)\,.$$

Differenssimenetelmä RAT:n (9.13) ratkaisemiseksi:

(1) *Diskretointi.* Merkitään $x_j = x(t_j)$, jolloin saadaan

$$\text{(9.15)} \qquad x_{j+1} - 2x_j + x_{j-1} = p(t_j)\frac{h}{2}(x_{j+1} - x_{j-1}) + h^2(q(t_j)x_j + r(t_j)) \quad (j = 1, \ldots, N-1)\,.$$

(2) *Lineaaristen yhtälöryhmien muodostus* muuttujille $x_1, \ldots, x_N$. Ottamalla huomioon (9.13):sta $x_0 = \alpha$, $x_N = \beta$ saadaan eo. yhtälöt muotoon

$$\begin{bmatrix} a_1 & c_1 & & & 0 \\ b_2 & a_2 & c_2 & & \\ & \ddots & \ddots & \ddots & \\ & & b_{n-2} & a_{n-2} & c_{n-2} \\ 0 & & & b_{n-1} & a_{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-2} \\ x_{n-1} \end{bmatrix} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{n-2} \\ w_{n-1} \end{bmatrix},$$

missä kertoimet a_j, b_j, c_j saadaan (9.15):stä.

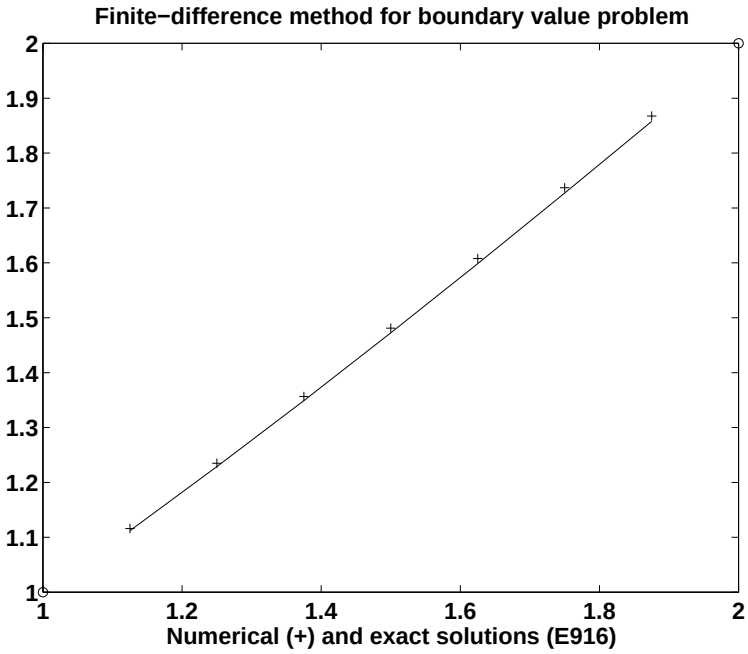
Tridiagonaalisten yhtälöryhmien ratkaisun kompleksisuus on $O(n)$.

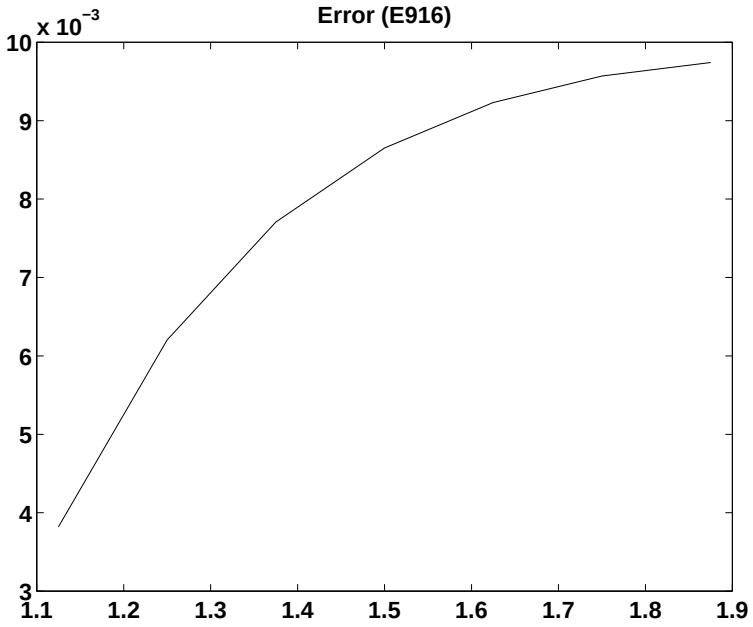
Algoritmeja differenssimenetelmille on mm. teoksissa [NR], [M, ch. 9].

9.16. Esimerkki. RAT $y'' = -\frac{2}{x}y' + \frac{2}{x^2}y + \frac{\sin(\log(x))}{x^2}$ $(1 \le x \le 2)$; $y(1) = 1$, $y(2) = 2$. MATLAB-koodi alla.

```
% FILE  e916.m  begins. Matlab 3.5, 4.0, and 5.3
% Solution of boundary value problem by finite difference method:
%    y(a)=bval1, y(b) = bval2, y" = p(t)y' + q(t)y + r(t)
% Put t(j) = a + j*h, j = (b-a)/(n+1), j = 0,1,..,n+1,
% FD approximation gives:
% y(t(j-1)) - 2y(t(j)) + y(t(j+1)) =
%    (h/2)(y(t(j+1)) -y(t(j-1)))p(t(j))+
%     h*h*q(t(j)) y(t(j)) + h*h*r(t(j)) or
% y(t(j-1))(1 +(h/2)p(t(j))) - (2+ h*h*q(t(j))) y(t(j))
%     + y(t(j+1))(1 -(h/2)p(t(j))) = h*h*r(t(j)), j = 1,,.n
% y(t(0)) = bval1, y(t(n+1)) = bval2
% Compare with exact solution given below
clear all;  clf;
a = 1; b = 2; h = (b-a)/8;  bval1 =1; bval2 = 2;
tt= a:h:b; t = tt(2:length(tt)-1);
n = length(t); one = ones(size(1:n));
% There are n interior points and n unknowns
p =-2*one./t; q = 2*one./(t.^2); r = sin(log(t))./(t.^2);
diagon = -(2*one +h*h*q);
tmp = (one- 0.5*h*p);         superd = tmp(1:n-1);
tmp = (one+ 0.5*h*(p));       subdia = tmp(2:n);
mtx = diag(diagon) + diag(superd,1) + diag(subdia,-1);
rhs =h*h*(r');
rhs(1,1) = rhs(1,1) -bval1*(1+(-2/(a+h))*0.5*h);
rhs(n,1) = rhs(n,1) -bval2*(1-(-2/(b-h))*0.5*h);
sol = mtx\rhs;
c2 = (1/70)*(8-12*sin(log(2)) - 4*cos(log(2)));
exact = (1.1-c2)*t +c2*one./(t.^2) - 0.3*sin(log(t)) - 0.1*cos(log(t));
plot(t, sol,'k+',a,bval1,'ko', b,bval2,'ko',t, exact),
xlabel('Numerical (+) and exact solutions (E916)'),
title('Finite-difference method for boundary value problem'),pause
print -deps fig916a
plot(t,sol-exact'), title('Error (E916)'), pause
print -deps fig916b
info(:,1) = sol; info(:,2) = exact';
delete e916.txt
diary e916.txt
disp('FILE  e916.txt  begins. ')
disp('Coefficient matrix ='), disp(mtx)
disp('Right hand side ='), disp(rhs)
disp('   t-values  numerical   exact     error    (E916)');
```

```
disp([ (a+h:h:b-h)' info, sol-exact' ])
disp('FILE  e916.txt  ends.')
diary off
% FILE  e916.m  ends.
```





```
FILE  e916.txt  begins.
Coefficient matrix =
   -2.0247    1.1111         0         0         0         0         0
    0.9000   -2.0200    1.1000         0         0         0         0
         0    0.9091   -2.0165    1.0909         0         0         0
         0         0    0.9167   -2.0139    1.0833         0         0
         0         0         0    0.9231   -2.0118    1.0769         0
         0         0         0         0    0.9286   -2.0102    1.0714
         0         0         0         0         0    0.9333   -2.0089

Right hand side =
   -0.8874
    0.0022
    0.0026
    0.0027
    0.0028
    0.0027
   -2.1307

   t-values  numerical   exact     error    (E916)
    1.1250    1.1160    1.1122    0.0038
    1.2500    1.2349    1.2287    0.0062
    1.3750    1.3567    1.3490    0.0077
    1.5000    1.4811    1.4724    0.0087
    1.6250    1.6079    1.5986    0.0092
    1.7500    1.7368    1.7272    0.0096
    1.8750    1.8676    1.8578    0.0097

FILE  e916.txt  ends.
```

10 Osittaisdifferentiaaliyhtälöt

10.1. Johdanto. Esimerkiksi

$$\frac{\partial^3 u}{\partial x^3} + (\frac{\partial u}{\partial t})^2 = \frac{\partial^2 u}{\partial x^2}$$

ja

$$\frac{\partial^2 u}{\partial t^2} = 2\frac{\partial^2 u}{\partial x \partial t} + u$$

(merk. myös $u_x = \frac{\partial u}{\partial x}$, $u_{xy} = \frac{\partial^2 u}{\partial x \partial y}$ jne.) ovat vastaavasti 3. ja 2. kertaluvun ODY:jä, $u = u(x,t)$.

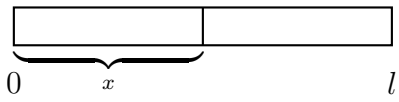
Tärkeitä "prototyyppejä" ovat $(u\colon D \to \mathbb{R})$

(1) $$\frac{\partial u}{\partial t} = \alpha^2 \frac{\partial^2 u}{\partial x^2}\,, \quad \textit{lämpöyhtälö,}$$

(2) $$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}\,, \quad \textit{aaltoyhtälö,}$$

(3) $$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0\,, \quad \textit{Laplacen yhtälö,}$$

(4) $$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = p(x,y)\,; \quad u \,|\, \partial D = g\,, \quad \textit{Poissonin yhtälö.}$$



(1):n ratkaisu $u(x,t)$ kuvaa (esim.) lämpötilaa homogeenisen sauvan kohdassa x ajanhetkellä t $(0 < x < l)$; α^2 on kerroin.

(2) esiintyy akustisten, vesi- ja sähkömagneettisten aaltojen tutkimuksessa (myös värähtelevä kieli).

(3) esiintyy sähkö- ja magneettikenttien potentiaalien tutkimuksessa, mistä toinen nimitys *potentiaaliyhtälö*.

Kuten tavallisten DY:den tapauksessa eivät ODY:den (1)–(4) ratkaisut yleensä ole yksikäsitteisiä. Tarvittaessa yhtälöihin (1)–(4) liitetään erilaisia täydentäviä ehtoja (esim. reunaehtoja), jotka takaavat ratkaisujen yksikäsitteisyyden.

Yhtälön (1) tapauksessa reunaehdot voivat esim. olla (i) lämpötilajakauma hetkellä $t = 0$ ts. $u(x,0) = f(x)$ ja lisäksi (ii) $u(0,t) = u(l,t) = 0 \quad \forall\ t \geq 0$ tai $u_x(0,t) = u_x(l,t) = 0$.

Yhtälön (2) tapauksessa reunaehto voi olla, kun tarkastellaan värähtelevää kieltä, (i) kielen asema kun $t = 0$, (ii) kielen nopeus hetkellä $t = 0$, (iii) päätepisteet paikallaan. Ts. $u(x,0) = f(x)$, $u_t(x,0) = g(x)$, $u(0,t) = u(l,t) = 0$.

Yhtälön (3) tapauksessa ei ole aikaparametria t. Luonnollinen lisäehto (3):een on esim. se, että annetaan u:n arvot alueen reunalla. Näin täydennetty (3) on kuuluisa "Dirichlet'n probleema".

Ratkaisun yksikäsitteisyys/olemassaolo voi olla selvää esim. fysikaalisin perustein.

ODY-teorian klassisena lähtökohtana on luokittelu yhtälöiden (1)–(3) mukaisiin prototyyppeihin ja kunkin luokan ominaispiirteiden selvittely. Täsmällisemmin ODY $A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} = f(x, y, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, u)$ on *elliptinen, parabolinen* t. *hyperbolinen* sen mukaan, onko $B^2 - 4AC < 0$, $= 0$ vai > 0. Tämä kolmijako muodostaa toisen kertaluvun ODY-teorian perustan.

10.2. Lämpöyhtälö. Lämpöyhtälön 10.1 (1) ratkaisuilla on *superpositio-ominaisuus*: Jos $u_k(x,t)$ toteuttaa yhtälön

$$\frac{\partial u}{\partial t} = \alpha^2 \frac{\partial^2 u}{\partial x^2}\,, \tag{10.3}$$

myös lineaarikombinaatio $\sum_{j=1}^{n} c_j u_j(x,t)$ toteuttaa (10.3):n. Lisäksi jos kukin u_k toteuttaa reunaehdot $u_k(0,t) = u_k(l,t) = 0 \quad \forall\, t$, niin myös lineaarikombinaatio $\sum c_j u_j$ toteuttaa samat ehdot.

Lämpöyhtälöön liittyvä reuna-arvotehtävä (RAT):

$$\frac{\partial u}{\partial t} = \alpha^2 \frac{\partial^2 u}{\partial x^2}\,; \qquad u(x,0) = f(x) \quad (0 < x < l,\ f \in C[0,l]), \tag{10.4}$$

$$u(0,t) = u(l,t) = 0\,.$$

Sovelletaan kaksivaiheista ajattelutapaa:

(A) Etsitään mahdollisimman monta RAT:n

$$\frac{\partial u}{\partial t} = \alpha^2 \frac{\partial^2 u}{\partial x^2}\,; \quad u(0,t) = u(l,t) = 0$$

ratkaisua $u_1, u_2, \ldots$.

(B) Yritetään löytää (10.4):n ratkaisu lineaarikombinaationa kohdan (A) funktioista u_k.

Vaiheen (A) toteutus: Etsitään ratkaisua $u(x,t)$ muodossa $u(x,t) = X(x)T(t)$. Nyt

$$\frac{\partial u}{\partial t} = X(x)T'(t), \quad \frac{\partial^2 u}{\partial x^2} = X''(x)T(t)\,,$$

joten ODY $u_t = \alpha^2 u_{xx}$ toteutuu jos

$$XT' = \alpha^2 X''T\,. \tag{10.5}$$

Jakamalla $\alpha^2 XT$:llä saadaan

$$\frac{X''}{X} = \frac{T'}{\alpha^2 T}\,. \tag{10.6}$$

Todetaan, että (10.6):n vasen puoli riippuu vain x:stä, oikea vain t:stä. Siis on olemassa vakio λ s.e.

$$\frac{X''}{X} = -\lambda \quad \text{ja} \quad \frac{T'}{\alpha^2 T} = -\lambda\,. \tag{10.7}$$

Reunaehdot $0 = u(0,t) = X(0)T(t),\ \ 0 = u(l,t) = X(l)T(t)$ antavat $X(0) = 0 = X(l)$.

Johtopäätös: $u(x,t) = X(x)T(t)$ toteuttaa (10.6):n vain, jos

$$X'' + \lambda X = 0;\ X(0) = 0 = X(l) \tag{10.8}$$

ja

$$T' + \lambda\alpha^2 T = 0\,. \tag{10.9}$$

Tässä vaiheessa λ oli mielivaltainen. Kuten 6.30 (1):ssä, voidaan osoittaa, että RAT (10.8):lla on ei-triviaali ratkaisu vain, jos

$$\lambda = \lambda_n = \frac{n^2\pi^2}{l^2} \quad (n = 1, 2, \ldots),$$

ja tällöin

$$X = X_n(x) = \sin\frac{n\pi x}{l}\,.$$

Yhtälö (10.9) puolestaan antaa

$$T(t) = T_n(t) = e^{-\alpha^2 n^2\pi^2 t/l^2}\,.$$

Myös $c_1 X_n(x)$ ja $c_2 T_n(t)$ ovat ratkaisuja. RAT:llä (10.4) on siis ei-triviaalit ratkaisut

$$u_n(x,t) = \sin\frac{n\pi x}{l} e^{-\alpha^2 n^2\pi^2 t/l^2} \quad (n = 1, 2, \ldots)\,. \tag{10.10}$$

Vaiheen (B) toteutus: Erikoistapaus missä f on äärellinen trigonometrinen summa:

$$f(x) = \sum_{n=1}^{N} c_n \sin\frac{n\pi x}{l}$$

($N < \infty$). Tällöin

$$u(x,t) = \sum_{n=1}^{N} c_n \sin\frac{n\pi x}{l} e^{-\alpha^2 n^2\pi^2 t/l^2} \tag{10.11}$$

on RAT:n (10.4) ratkaisu superpositioperiaatteen nojalla, sillä (a) (10.10):n nojalla u_n on ratkaisu, (b) (a):n nojalla u on ratk. ja (c) u toteuttaa RAT:n, erityisesti

$$u(x,0) = \sum_{n=1}^{N} c_n \sin \frac{n\pi x}{l} = f(x)\,.$$

Yleinen tapaus: Koska $f \in C[0,l]$, on sillä Fourier-sarja

$$f(x) = \sum_{n=0}^{\infty} c_n \sin \frac{n\pi x}{2}\,; \quad c_n = \frac{2}{l}\int_0^l f(x) \sin \frac{n\pi x}{l}\,dx\,. \tag{10.12}$$

Itse asiassa (10.12) on "f:n pariton laajennus" ts. $f(x) = -f(-x)$, jolloin Fourier-sarjan cos-termejä ei ole. Määritellään RAT:n (10.4) *formaali ratkaisu*

$$u(x,t) = \sum_{n=1}^{\infty} c_n \sin \frac{\pi n x}{l} e^{-\alpha^2 n^2 \pi^2 t/l^2}\,. \tag{10.13}$$

Huomautus. Jotta nähtäisiin, että formaali ratkaisu (10.13) on "todellinen ratkaisu", pitäisi osoittaa, että sarja (10.13) suppenee ja että sillä on olemassa (10.4):ssa esiintyvät osittaisderivaatat ja että (10.4) toteutuu. Tämä päättely kuitenkin sivuutetaan.

10.14. Esimerkki. Hetkellä $t = 0$ ohuen kuparisauvan ($\alpha^2 = 1.14$) lämpötila $u(x,0) = 2\sin(3\pi x) + 5\sin(8\pi x)$, kun $0 \le x \le 1$, ja pituus on 1. Etsi lämpötila $u(x,t)$ arvoilla $t > 0$.

Ratkaisu.

$$\text{RAT:}\ \ \frac{\partial u}{\partial t} = 1.14\frac{\partial^2 u}{\partial x^2};\ \ \begin{cases} u(x,0) = 2\sin(3\pi x) + 5\sin(8\pi x) \\ u(0,t) = u(1,t) = 0 \end{cases}$$

$$\Rightarrow\ u(x,t) = 2\sin(3\pi x)e^{-9(1.14)\pi^2 t} + 5(\sin(8\pi x))e^{-64(1.14)\pi^2 t}\,. \tag{10.11}$$

10.15. Esimerkki. 10 cm:n pituinen alumiinisauva $\alpha^2 = 0.86$ lämmitetään 100°C:n lämpötilaan. Hetkellä $t = 0$ sauvan päät upotetaan 0°C:n jääkylpyyn ja pidetään tässä lämpötilassa. Sivupintojen kautta ei lämpöä poistu. Etsi sauvan lämpötilajakauma hetkellä t (= RAT:n formaali ratkaisu).

Ratkaisu.

$$\text{RAT:}\ \ \frac{\partial u}{\partial t} = 0.86\frac{\partial^2 u}{\partial x^2}\,;\ \ \begin{cases} u(x,0) = 100 \quad (0 < x < 10) \\ u(0,t) = u(10,t) = 0 \end{cases}\,. \tag{10.16}$$

(10.16):n formaali ratkaisu

$$u(x,t) = \sum_{n=1}^{\infty} c_n \sin\tfrac{n\pi x}{10} \exp(-0.86\tfrac{n^2\pi^2 t}{100})\,;$$
$$c_n = \tfrac{1}{5}\int_0^{10} 100 \sin\tfrac{n\pi x}{10}\,dx = \tfrac{200}{n\pi}(1-\cos(n\pi))\,.$$

Nyt $c_{2n} = 0$ ja $c_{2n+1} = \frac{400}{(2n+1)\pi}$, kun $n = 0, 1, 2, \ldots$, joten

$$u(x,t) = \frac{400}{\pi}\sum_{n=0}^{\infty} \frac{\sin(2n+1)\frac{\pi x}{10}}{(2n+1)} \exp\Big(-0.86(2n+1)^2\frac{\pi^2 t}{100}\Big)\,.$$

Em. menetelmää, jossa ODY:lle haetaan muotoa $u(x,t) = X(x)T(t)$ olevaa ratkaisua, sanotaan *separointimenetelmäksi*. Se soveltuu moniin yo. tyyppiä oleviin tehtäviin. Seuraavaksi perehdymme sen käyttöön Laplacen yhtälön tapauksessa.

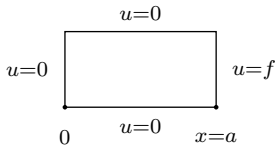
10.17. Dirichlet'n ongelma. Annettu alue $D \subset \mathbb{R}^2$ ja $f \in C(\partial D)$. Etsitään funktiota $u\colon D\cup\partial D \to \mathbb{R}$, $u \in C(D\cup\partial D)\cap C^2(D)$, s.e. $u(x,y) = f(x,y)$ kaikilla $(x,y) \in \partial D$ ja

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \quad u = u(x,y)\,. \tag{10.18}$$

ODY-teorian yleisten tulosten nojalla on olemassa ratkaisu, jos D on monikulmio.

Sovellamme tätä yleistä tulosta erikoistapaukseen, missä D on suorakulmio ja

$$u(x,0) = 0\,,\ u(x,b) = 0\,,\ u(0,y) = 0\,,\ u(a,y) = f(a,y)\,. \tag{10.19}$$



Ratkaisu. Jotta reuna-arvot olisivat jatkuvia, on syytä olettaa, että $f(a,0) = f(a,b) = 0$. Kaksi vaihetta:

Vaihe 1. Etsitään funktiot $u_n(x,y) = X_n(x)Y_n(y)$ s.e. (10.18) toteutuu ja $u_n(x,0) = 0$, $u_n(x,b) = 0$, $u_n(0,y) = 0$.

Vaihe 2. Etsitään c_n s.e. $\sum_{n=1}^{\infty} c_n u_n(x,y) = u(x,y)$ toteuttaa ehdon $u(a,y) = f(a,y)$. Merkitään myös $f(y) = f(a,y)$.

Vaiheen 1 toteutus. Etsitään muotoa $u(x,y) = X(x)Y(y)$ oleva (10.18):n ratkaisu u. Nyt $u_{xx} = X''Y$ ja $u_{yy} = XY''$, joten u toteuttaa Laplacen yhtälön, jos $X''Y + XY'' = 0$ eli

$$\frac{Y''}{Y} = -\frac{X''}{X}\,. \tag{10.20}$$

Tällöin (10.20):n vasen (oikea) puoli riippuu vain y:stä (x:stä). Kuten (10.7):ssa nähdään, että

$$Y''/Y = -X''/X = -\lambda$$

jollekin vakiolle λ. Reunaehdot $0 = u(x,0) = X(x)Y(0),\ 0 = u(x,b) = X(x)Y(b),$ $0 = u(0,y) = X(0)Y(y)$ antavat $Y(0) = 0$, $Y(b) = 0$, $X(0) = 0$. Olemme nähneet, että $u(x,y) = XY$ toteuttaa (1):n vain, jos

$$Y'' + \lambda Y = 0;\quad Y(0) = 0,\ Y(b) = 0 \tag{10.21}$$

ja

$$X'' - \lambda X = 0;\quad X(0) = 0\,. \tag{10.22}$$

Tässä vaiheessa λ on jokin vakio. Kuitenkin RAT (10.21):llä on ei-triviaali ratkaisu $Y(y)$ vain, jos $\lambda = \lambda_n = n^2\pi^2/b^2$ $(n = 1, 2, \ldots)$; vrt. 6.30 (1). Tällöin

$$Y(y) = Y_n(y) = \sin(n\pi y/b)\,.$$

Kun $\lambda = n^2\pi^2/b^2$, (10.22) antaa $X(x) = c_1 \cosh \frac{n\pi x}{b} + c_2 \sinh \frac{n\pi x}{b}$ joillekin c_1, c_2, ja vaatimus $X(0) = 0$ antaa $c_1 = 0$.

Johtopäätös:

$$u_n(x,y) = \sinh\frac{n\pi x}{b}\sin\frac{n\pi y}{b} \qquad (n = 1, 2, \ldots) \tag{10.23}$$

on (10.18):n ratkaisu.

Vaiheen (2) toteutus. Funktio

$$u(x,y) = \sum_{n=1}^{\infty} c_n \sinh\frac{n\pi x}{b}\sin\frac{n\pi y}{b} \tag{10.24}$$

on (10.18):n *formaali ratkaisu* kaikille c_i (formaalin ratkaisun määritelmä). Nyt

$$u(a,y) = \sum_{n=1}^{\infty} c_n \sinh\frac{n\pi a}{b}\sin\frac{n\pi y}{b}$$

ja (10.19):n mukaan $u(a,y) = f(y)$. Siis

$$f(y) = \sum_{n=1}^{\infty} c_n \sinh\frac{n\pi a}{b}\sin\frac{n\pi y}{b} \qquad (0 < y < b)\,,$$

joten Fourier-sarjan yksikäsitteisyydestä seuraa, että

$$c_n = \frac{2}{b \sinh \frac{n\pi a}{b}} \int_0^b f(y) \sin \frac{n\pi y}{b}\, dy \quad (n = 1, 2, \ldots)\,. \tag{10.25}$$

Todetaan: Välttämätön ehto sille, että (10.24) on RAT:n (10.19) ratkaisu, on (10.25). (Riittävyys sivuutetaan.)

10.26. Huomautus. Tavallisten DY:iden teoriassa on monia yhtälöluokkia, joille on eksplisiittinen ratkaisukaava, kuten lineaariset 1. kertaluvun DY:t tai 2. kertaluvun vakiokertoimiset yhtälöt. ODY-teoriassa tällaisia "hyviä tapauksia" on valitettavan vähän. Edellä esitetty Fourier-sarjoihin ja muuttujien separointiin perustuva menetelmä antaa em. tilanteissa eksplisiittisen ratkaisun.

10.27. ODY:n numeriikasta. Numeerinen ODY-teoria on tietokoneiden myötä kehittynyt laajaksi kokeellisen matematiikan alueeksi. Monien sovellustensa vuoksi se lienee numeriikan tärkeimpiä osa-alueita nykyään. Aktiivisen tutkimuksen kohteena on mm. FEM (finite element) -menetelmä ja ns. multigrid-menetelmä.

Algoritmeja löytyy mm. NR:stä ja Mathewsin kirjasta [M]. Tarkoitus on tämän luvun loppuosassa esitellä numeerisia tekniikoita lähinnä Mathewsin esityksen mukaisesti. Tilanteen yksinkertaistamiseksi joudutaan rajoittumaan pääasiassa tapaukseen, jossa alue D on $\mathbb{R}^2$:n suorakulmio.

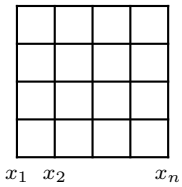
10.28. Aaltoyhtälön numeerinen ratkaisu. Tarkastelemme uudelleen aaltoyhtälöä, joka edellä jo ratkaistiin Fourier-menetelmällä. Nyt menetelmänä on 9.12:n kaltainen differenssimenetelmä.

Nyt tehtävänä on ratkaista RAT

$$\begin{cases} u_{tt}(x,t) = c^2 u_{xx}(x,t) \quad (0 < x < a,\ 0 < t < b) \\ u(0,t) = 0 \quad \text{ja} \quad u(a,t) = 0 \quad \forall\ t \in (0,b) \\ u(x,0) = f(x) \quad \forall\ x \in (0,a) \\ u_t(x,0) = g(x) \quad \forall\ x \in (0,a) \end{cases}. \tag{10.29}$$

Merkitään $R = \{(x,t) : 0 \le x \le a,\ 0 \le t \le b\}$.

Differenssiyhtälöiden johto. Jaetaan R $(n-1)(m-1)$:een samankokoiseen suorakulmioon, jakovälit x-akselilla $\Delta x = h$ ja t-akselilla $\Delta t = k$. Ratkaisun todellinen arvo verkkopisteissä on $u(x_i, t_j)$ ja likiarvo u_{ij}.



Käytetään kaavaa (9.14), jolloin

$$u_{tt}(x,t) = \frac{u(x,t+k) - 2u(x,t) + u(x,t-k)}{k^2} + O(k)$$

ja

$$u_{xx}(x,t) = \frac{u(x+h,t) - 2u(x,t) + u(x-h,t)}{h^2} + O(h)\,.$$

Merkitään $x_{j+1} = x_j + h$ ja $t_{i+1} = t_i + k$ sekä korvataan $u(x_i, t_j)$ u_{ij}:llä, jolloin (10.29):n nojalla

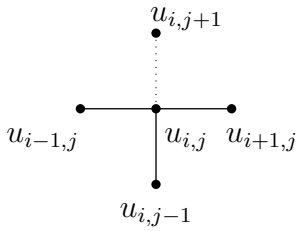
$$\frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{k^2} = c^2 \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2}\,. \tag{10.30}$$

Sijoitetaan (10.30):een $r = ck/h$, jolloin saamme

$$u_{i,j+1} - 2u_{i,j} + u_{i,j-1} = r^2(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) \tag{10.31}$$

ja edelleen

$$\begin{aligned} u_{i,j+1} &= (2 - 2r^2)u_{i,j} \\ &+ r^2(u_{i+1,j} + u_{i-1,j}) - u_{i,j-1} \quad (i = 2, 3, \ldots, n-1)\,. \end{aligned} \tag{10.32}$$



Differenssimenetelmää kutsutaan *stabiiliksi*, jos vaiheessa j tehty virhe vaimenee laskennan edetessä. Välttämätön ehto stabiiliudelle on, että $r = ck/h \leq 1$.

Alkuarvot. Rivin $j = 3$ arvojen laskemiseksi kaavasta (10.32) tarvitaan rivit $j = 2$ ja $j = 1$. Rivi $j = 1$ tunnetaan RAT:n (10.29) nojalla. Nyt

$$u(x_i, k) = u(x_i, 0) + u_t(x_i, 0)k + O(k^2)\,. \tag{10.33}$$

Käytetään merkintöjä $u(x_i, 0) = f(x_i) = f_i$, $u_t(x_i, 0) = g(x_i) = g_i$ (10.33):ssa, jolloin saadaan toisen rivin alkioille

$$u_{i,2} = f_i + kg_i \quad (i = 2, 3, \ldots, n-1)\,. \tag{10.34}$$

Jos on olemassa f'', niin $u_{xx}(x, 0) = f''$ ja

$$\begin{aligned} u_{tt}(x_i, 0) &= c^2 u_{xx}(x_i, 0) \\ &= c^2 f_{xx}(x_i) = c^2 \frac{f_{i+1} - 2f_i + f_{i-1}}{h^2} + O(h^2)\,. \end{aligned} \tag{10.35}$$

Taylorin kaava antaa approksimaation

$$u(x, k) = u(x, 0) + u_t(x, 0)k + \frac{u_{tt}(x, 0)k^2}{2} + O(k^3)\,. \tag{10.36}$$

Nyt (10.34)–(10.36) yhdessä antavat

$$\begin{aligned} u(x_i, k) = f_i + kg_i + \frac{c^2k^2}{2h^2}(f_{i+1} - 2f_i + f_{i-1}) \\ + O(h^2)O(k^2) + O(k^3)\,. \end{aligned} \tag{10.37}$$

Asettamalla $r = ck/h$ saadaan lopputulokseksi

$$u_{i,2} = (1 - r^2)f_i + kg_i + \frac{r^2}{2}(f_{i+1} + f_{i-1}) \quad (i = 2, 3, \ldots, n-1)\,. \tag{10.38}$$

10.39. d'Alembertin ratkaisu. Ranskalainen d'Alembert (1717–1783) huomasi, että

$$u(x, t) = F(x + ct) + G(x - ct) \tag{10.40}$$

toteuttaa aaltoyhtälön (10.29) välillä $0 < x < a$, jos on olemassa F', F'', G', G'' ja $F(-z) = -F(z)$, $G(-z) = -G(z)$, $G(z + 2a) = G(z)$, $F(z + 2a) = F(z)$ $\forall z$.

Todistus:

$$\left.\begin{aligned} u_{tt} &= c^2F''(x + ct) + c^2G''(x - ct) \\ u_{xx} &= F''(x + ct) + G''(x - ct) \end{aligned}\right\} \Rightarrow \text{(1) pätee.}$$

RAT (10.29) $\Rightarrow$ $F(x) = G(x) = f(x)/2$. □

10.41. Lause. Oletetaan että $u_{i,1}(x_i, 0)$, $u_{i,2}(x_i, k)$ $(i = 1, 2, \ldots, n)$ ovat (10.29):n eksakteja ratkaisuja. Jos asetetaan $k = h/c$, on kaavassa 10.28 (4) $r = 1$ ja edelleen

(10.42) $$u_{i,j+1} = u_{i+1,j} + u_{i-1,j} - u_{i,j-1}\,.$$

Lisäksi (10.42) antaa *eksaktin* ratkaisun.

Todistus. Kun $ck = h$, pätee $x_i - ct_j = (i-1)h - c(j-1)k = (i-1)h - (j-1)h = (i-j)h$, ja vastaavasti $x_i + ct_j = (i+j-2)h$, jolloin (10.40) antaa

$$u_{i,j} = F((i-j)h) + G((i+j-2)h) \quad (i = 1, 2, \ldots, n,\ j = 1, 2, \ldots, m)$$

ja edelleen

$$u_{i+1,j} + u_{i-1,j} - u_{i,j-1} = F((i+1-j)h) + F((i-1-j)h)$$

$$-F((i-(j-1))h) + G((i+1+j-2)h) + G((i-1+j-2)h) - G((i+j-1-2)h)$$
$$= F((i-(j+1)h) + G((i+j+1-2)h) = u_{i,j+1}\,,$$

kun $i = 1, \ldots, n$ ja $j = 1, 2, \ldots, m$.

10.43. Esimerkki. Värähtelevän kielen aaltoyhtälön ratkaisu finite-difference-menetelmällä (ks. Mathews [M, p. 505]).

$$\begin{aligned}
&u_{tt}(x,t) = 4u_{xx}(x,t) \quad (0 < x < 1,\ 0 < t < 0.5),\\
&u(0,t) = 0\ u(1,t) = 0 \quad (0 \le t \le 0.5),\\
&u(x,0) = f(x) = \sin(\pi x) + \sin(2\pi x) \quad (0 \le x \le 1),\\
&u_t(x,0) = g(x) = 0 \quad (0 \le x \le 1)\,.
\end{aligned}$$

Valitaan $h = 0.1$, $k = 0.05$. Koska $c = 2$, niin $r = ck/h = 1$, ja koska $g(x) = 0$ ja $r = 1$, niin (10.38) antaa

$$u_{i,2} = (f_{i-1} + f_{i+1})/2 \quad (i = 2, \ldots, 9)\,.$$

Asetetaan $r = 1$ kaavassa (10.32), jolloin saadaan

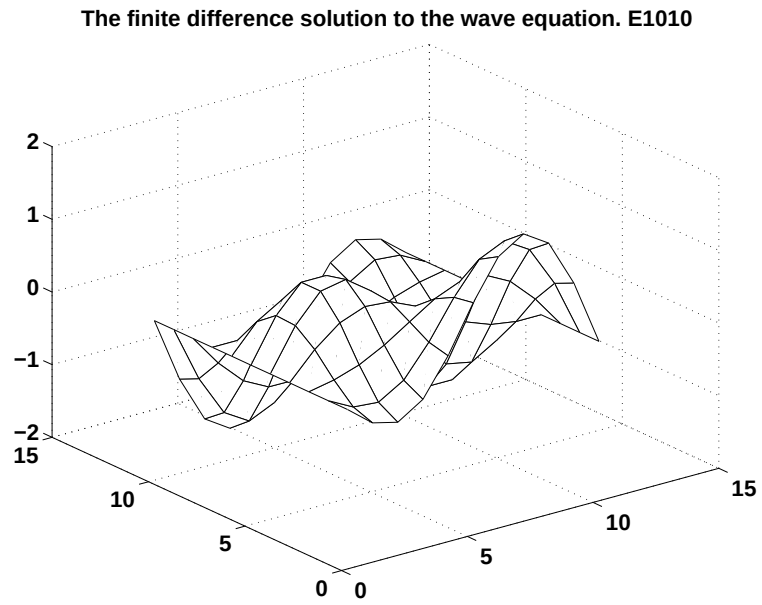
$$u_{i,j+1} = u_{i+1,j} + u_{i-1,j} - u_{i,j-1}\,.$$

MATLAB-ratkaisu alla.

```
% FILE  e1010.m  begins.
echo off;
% Simplified version (finedif.m not needed) of a10_1.m:
% Algorithm 10.1 (Finite-Difference Solution
```

```
%                 for the Wave Equation).
% Section  10.1,  Hyperbolic Equations, Page 507
% in: NUMERICAL METHODS by John H. Mathews
echo on; clf; format short; hold off; clear
% HYPERBOLIC PDE's.
% Finite difference solution for the wave equation
%                          2
%          u  (x,t)   =  c  u  (x,t)     with
%           tt               xx

%{\tt\%   u(0,t) = 0     and   u(a,t) = 0      for 0 \py t \py b.}

%   u(x,0) = f(x)  and  u (x,0) = g(x)   for  0 < x < a.
%                        t

% A numerical approximation is computed over the rectangle

%{\tt \%         0 \py x \py a ,   0 \py t \py b.}

pause  % Press any key to continue.

clf;
% Store f(x) and g(x) in the M-files f.m and g.m respectively.
% function z = f(x)
% z = sin(pi*x) + sin(2*pi*x);

% function z = g(x)
% z = 0;
delete e1010.txt
delete f.m
clear f
diary f.m; disp('function z = f(x)');...
           disp('z = sin(pi*x) + sin(2*pi*x);');...
diary off;

delete g.m
clear g
diary g.m; disp('function z = g(x)');...
           disp('z = 0;');...
diary off;
```

```
% Remark. f.m  g.m finedif.m are used for Algorithm 10.1
f(0); g(0);

pause  % Press any key to continue.
clf;
%  Place the endpoint of [0,a] in  a.
%  Place the endpoint of [0,b] in  b.
%  For the wave equation, enter the constant  c.
%  Over [0,a] enter the number of grid points  n.
%  Over [0,b] enter the number of grid points  m.
a  =  1;   b  =  0.5;
c  =  2;    Un  =  11;
m  =  11;   n = 11;
U = zeros(m,n);
h = 1/(n-1);
x = 0:h:1;
U(1,:) = f(x);
U(2,2:n-1) =0.5*(f(x(1:n-2)) + f(x(3:n)));
for j = 3:m
    U(j,2:n-1) = U(j-1,3:n)+U(j-1,1:n-2)-U(j-2,2:n-1);
end;
pause  % Press any key to see the solution.
clf;
Z = fliplr(U);
mesh(Z);
Mx1='The finite difference solution to the wave equation. E1010';
title(Mx1);
shg; pause % Press any key to continue.
clf,echo off,diary e1010.txt
disp(' '),disp(Mx1),disp(' '),disp(U)
diary off,echo on
% FILE  e1010.m  ends.
```



10.44. Laplacen yhtälön numeerinen ratkaisu. Laplacen yhtälön

$$\nabla^2 u = u_{xx} + u_{yy} = 0$$

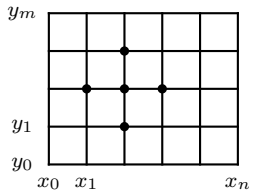
diskretointiin käytämme approksimointia

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2)\,,$$

joten

(10.45)
$$\nabla^2 u(x,y) = \frac{u(x+h,y) + u(x-h,y) + u(x,y+h) + u(x,y-h) - 4u(x,y)}{h^2} + O(h^2)\,.$$

Jaetaan suorakulmio $R = \{(x,y) : 0 \le x \le a,\ 0 \le y \le b,\ b/a = m/n\}$ $n \times m$ neliöön, joista kunkin sivunpituus on h $(a = nh,\ b = mh)$.



Merkitään

$$x_i = ih \quad (i = 0, \ldots, n)\,,$$
$$y_j = jh \quad (j = 0, \ldots, m)\,.$$

Yhtälöstä (10.45) saadaan

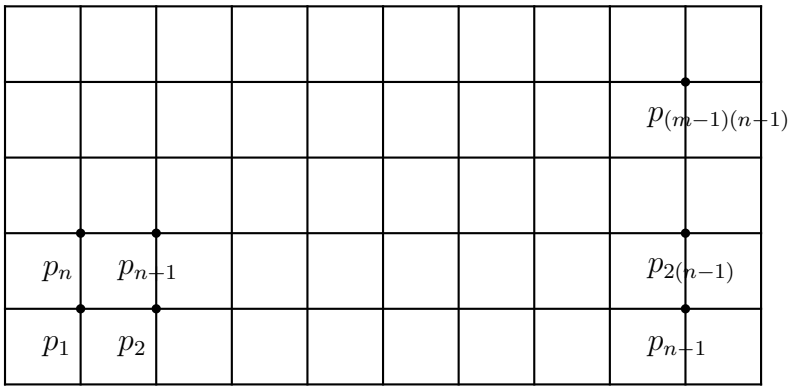
$$\nabla^2 u_{i,j} = \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} = 0\,. \tag{10.46}$$

Kaava (10.46) on nimeltään *viiden pisteen differenssikaava* Laplacen yhtälölle. (Vrt. *harmonisten funktioiden* keskiarvo-ominaisuus.)

Oletamme, että arvot $u_{i,j}$ tunnetaan reunalla (kun $i = 0$ tai n tai $j = 0$ tai m), kuten on laita esim. Dirichlet'n ongelmassa.

Verkon sisäpisteitä ($1 \le i \le n-1,\ 1 \le j \le m-1$) on $(n-1) \times (m-1)$ kpl ja tuntemattomia $u_{i,j}$ samoin.

Muodostetaan tuntemattomista vektori $p = (p_1, \ldots, p_{(n-1)\times(m-1)})$ seuraavasti:

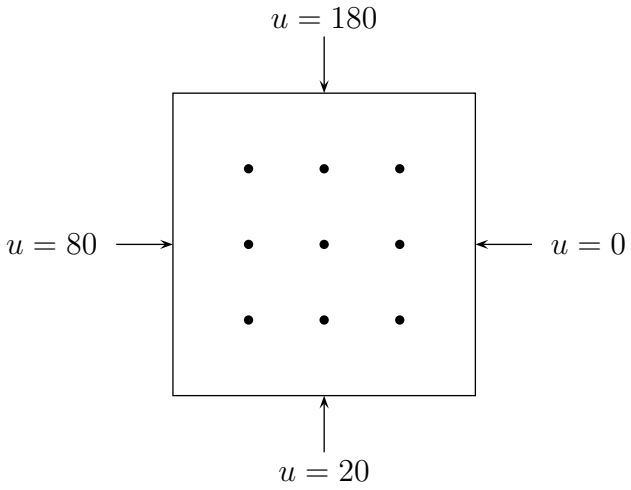


Yhtälöt kirjoitetaan p_i:ille indeksin i mukaisessa järjestyksessä käyttäen kaavaa (10.46) ja ottaen huomioon reuna-arvot.

10.47. Esimerkki. Etsi Dirichlet'n probleeman $\nabla^2 u = 0$,

$$u(x,0) = 20,\ u(x,4) = 180 \quad (0 < x < 4),$$
$$u(0,y) = 80,\ u(4,y) = 0 \quad (0 < y < 4),$$

ratkaisu alueen $R = \{(x,y) : 0 \le x \le 4,\ 0 \le y \le 4\}$ pisteissä (i,j), missä $i = 1,2,3$ ja $j = 1,2,3$.



1. yhtälö on $80 + p_2 + 20 + p_4 - 4p_1 = 0$.
2. yhtälö on $p_1 + p_3 + 20 + p_5 - 4p_2 = 0$.

Saadaan yhtälöryhmä (Mathews s. 524)

$$
\begin{aligned}
-4p_1 + p_2 + p_4 &= -100\\
p_1 - 4p_2 + p_3 + p_5 &= -20\\
p_2 - 4p_3 + p_6 &= -20\\
p_1 - 4p_4 + p_5 + p_7 &= -80\\
p_2 + p_4 - 4p_5 + p_6 + p_8 &= 0\\
p_3 + p_5 - 4p_6 + p_9 &= 0\\
p_4 - 4p_7 + p_8 &= -260\\
p_5 + p_7 - 4p_8 + p_9 &= -180\\
p_6 + p_8 - 4p_9 &= -180
\end{aligned}
$$

MATLAB-ratkaisu:

```
% FILE  e1012.m  begins.
% GENERATES: e1012.txt
% Dirichlet problem in R = {(x,y): 0 < x < 4, 0 < y < 4}:
%   u    + u      = 0
%    xx      yy
%
%   u(x,0) = 20, u(x,4) =180    0 < x < 4
%   u(0,y) = 80, u(4,y) = 0     0 < y < 4
```

```
a = -4*diag(ones(1,9))+ diag(ones(1,8),1) + diag(ones(1,8),-1);
a = a + diag(ones(1,6),3) + diag(ones(1,6),-3);
% This is a "rough approximation" of a. Now must correct some
% terms to zero:
a(3,4) =0; a(4,3)=0; a(6,7) =0; a(7,6) =0;
rhs(1) = -100; rhs(2) = -20;rhs(3) = -20;rhs(4) = -80; rhs(5) = 0;
rhs(6) = 0; rhs(7) = -260; rhs(8) = -180; rhs(9) = -180;
a, pause;
rhs', pause;
delete e1012.txt;
echo off
diary e1012.txt;
disp('FILE  e1012.txt  begins.');
disp('10.12. Esim. Dirichlet''n probleema');
eqn = [a  rhs'], pause;
p = a\(rhs');
disp('Solution:');
disp(p');
disp('FILE  e1012.txt  ends.');
diary off;
% FILE  e1012.m  ends.
```

```
FILE  e1012.txt  begins.
10.12. Esim. Dirichlet'n probleema

eqn =

    -4     1     0     1     0     0     0     0     0  -100
     1    -4     1     0     1     0     0     0     0   -20
     0     1    -4     0     0     1     0     0     0   -20
     1     0     0    -4     1     0     1     0     0   -80
     0     1     0     1    -4     1     0     1     0     0
     0     0     1     0     1    -4     0     0     1     0
     0     0     0     1     0     0    -4     1     0  -260
     0     0     0     0     1     0     1    -4     1  -180
     0     0     0     0     0     1     0     1    -4  -180

Solution:
  Columns 1 through 7

   55.7143   43.2143   27.1429   79.6429   70.0000   45.3571  112.8571
```

```
  Columns 8 through 9

  111.7857   84.2857

FILE  e1012.txt  ends.
```

10.48. Iteratiivisista menetelmistä. Viiden pisteen differenssimenetelmä Laplacen yhtälölle 10.44:n mukaisin jakopistein vaatii $(n-1)\times(m-1)$ tuntemattoman etsimistä lineaarisesta yhtälöryhmästä. Yhtälöryhmän matriisilla on $(n-1)^2\times(m-1)^2$ alkiota, joten tilantarve kasvaa neliöllisesti n:n ja m:n funktiona. Iteratiivisissa menetelmissä ideana on redusoida tilantarve niin, että käytetään tilaa vain $(n-1)\times(m-1)$ tuntemattomalle.

Tarkastelemme Dirichlet'n ongelmaa malliprobleemana; käytämme merkintöjä kuten 10.44:ssä. Reuna-arvot

$$\text{(10.49)}\qquad \begin{cases} u_{0,j} = u(x_0, y_j) & (1 \le j \le m-1) \\ u_{i,0} = u(x_i, y_0) & (1 \le i \le n-1) \\ u_{n,j} = u(x_n, y_j) & (1 \le j \le m-1) \\ u_{i,m} = u(x_i, y_m) & (1 \le i \le n-1) \end{cases}$$

tunnetaan, koska kyseessä on Dirichlet'n probleema. Laplacen yhtälön ratkaisut ovat harmonisia funktioita, joille pätee keskiarvo-ominaisuus. Käytämme viiden pisteen differenssikaavaa (10.46), keskiarvo-ominaisuuden diskreettiä analogiaa

$$\text{(10.50)}\qquad u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j} = 0\,.$$

Kirjoitetaan (10.50) ekvivalenttiin muotoon

$$\text{(10.51)}\quad u_{i,j} = u_{i,j} + r_{i,j} \quad\wedge\quad r_{i,j} = \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{4}\,,$$

kun $1 \le i \le n-1$ ja $1 \le j \le m-1$.

Olkoon K reuna-arvojen (10.49) keskiarvo. Iteratiivisen menetelmän yksi iteraatioaskel koostuu kaikkien $u_{i,j}$-arvojen läpikäynnistä kaavan (10.51) mukaisesti, alkuarvona K. Iteraatiota toistetaan, kunnes kaikki $|r_{i,j}|$:t ovat $< \varepsilon$, tai vaihtoehtoisesti enintään tietty määrä kertoja.

SOR-menetelmässä (*s*uccessive *o*ver-*r*elaxation) käytetään (10.51):n asemasta kaavaa

$$\text{(10.52)}\qquad u_{i,j} = u_{i,j} + w r_{i,j}\ , \quad \text{missä}$$

$$w = 4 \Big/ \left(2 + \sqrt{4 - \left(\cos\frac{\pi}{n} + \cos\frac{\pi}{m} \right)^2} \right).$$

10.53. Esimerkki. SOR-menetelmä Dirichlet'n ongelman $\nabla^2 u = 0$,

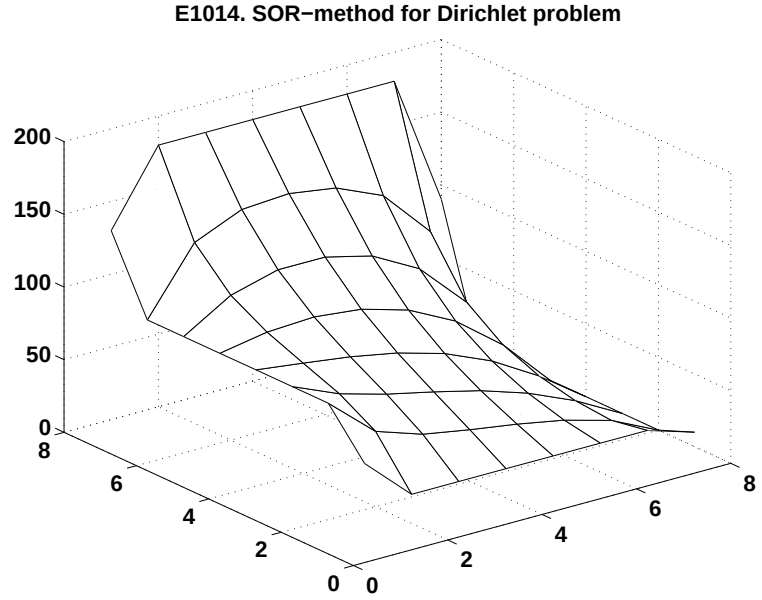
$$u(x,0) = 20,\ u(x,4) = 180 \quad (0 < x < 4),$$
$$u(0,y) = 80,\ u(4,y) = 0 \quad (0 < y < 4),$$

ratkaisun approksimoimiseksi.

Ks. Mathews s. 528 Ex. 10.7.

```
% FILE  e1014.m  begins.
% GENERATES: e1014.txt
% Dirichlet problem in R = {(x,y): 0 < x < 4, 0 < y < 4}:
%   u    + u      = 0
%    xx      yy
%
%   u(x,0) = 20, u(x,4) =180    0 < x < 4
%   u(0,y) = 80, u(4,y) = 0     0 < y < 4
% is solved using the SOR-method (Mathews, p. 528)
echo off;
n = 8; m = 8;
u = zeros(n,m);
u(1,:) =20*ones(1,n);
u(m,:) =180*ones(1,n);
u(2:m-1,1)=80*ones(m-2,1);
u(2:m-1,n)=0*ones(m-2,1);
u(1,1) =50; u(1,n) =10;
u(m,1) =130; u(m,n) =90;
K =(1/4)*(180 + 20 + 80 + 0);
u(2:n-1,2:m-1) =K*ones(n-2,m-2);
u, pause;
omega = 4/(2 + sqrt( 4 - (cos(pi/(n-1)) + cos(pi/(m-1)))^2 ));
for ite =1:40
for j =2:n-1
  for i=2:m-1
     r = (u(i+1,j) + u(i-1,j) + u(i,j+1) + u(i,j-1) - 4*u(i,j));
     u(i,j) = u(i,j) + 0.25*r*omega;
  end;
end;
end;
delete e1014.txt
diary e1014.txt;
disp('FILE  e1014.txt  begins.');
```

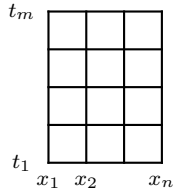
```
disp('10.14. Esim. Dirichlet''n probleema SOR-menetelmalla');
u, pause;
disp('FILE  e1014.txt  ends.');
diary off;
mesh(u), title('E1014. SOR-method for Dirichlet problem'),pause;
% FILE  e1014.m  ends.

FILE  e1014.txt  begins.
10.14. Esim. Dirichlet'n probleema SOR-menetelmalla

u =

  Columns 1 through 7

   50.0000   20.0000   20.0000   20.0000   20.0000   20.0000   20.0000
   80.0000   51.8284   41.0986   35.3809   30.4411   24.3705   15.4852
   80.0000   66.2149   57.1852   49.9840   42.0128   31.5557   17.5702
   80.0000   75.8460   71.4435   65.3569   56.0706   42.2694   23.2398
   80.0000   85.7257   87.3857   83.9294   74.6431   58.2117   33.1195
   80.0000   99.6712  108.4443  108.3320  100.3609   82.8148   51.0265
   80.0000  124.5148  138.3881  140.5934  135.6536  121.6600   88.1716
  130.0000  180.0000  180.0000  180.0000  180.0000  180.0000  180.0000

  Column 8

   10.0000
         0
         0
         0
         0
         0
         0
   90.0000

FILE  e1014.txt  ends.
```



10.54. Lämpöyhtälöistä. RAT

$$\text{(10.55)} \qquad \begin{cases} u_t(x,t) = c^2 u_{xx}(x,t) & (0 < x < a,\ 0 < t < b) \\ u(x,0) = f(x) & (t = 0,\ x \in (0,a)) \\ u(0,t) = g_1(t) = c_1 & (x = 0,\ t \in (0,b)) \\ u(a,t) = g_2(t) = c_2 & (x = a,\ t \in (0,b)) \end{cases}$$

kuvaa lämpötilan jakautumista (eristetyssä) sauvassa, jonka päät ovat vakiolämpötiloissa c_1 ja c_2 ja jonka lämpötilajakautuma hetkellä $t = 0$ on $f(x)$. Kohdassa 9.2 esitettiin (10.55):n Fourier-sarja-ratkaisu. Nyt esitämme numeerisen finite-difference-ratkaisun.



Käytetään approksimaatioita

$$\text{(10.56)} \qquad u_t(x,t) = \frac{u(x,t+k) - u(x,t)}{k} + O(k)\,,$$

$$\text{(10.57)} \qquad u_{xx}(x,t) = \frac{u(x-h,t) - 2u(x,t) + u(x+h,t)}{h^2} + O(h^2)\,,$$

$$x_{i+1} = x_i + h,\ t_{j+1} = t_j + k,\ u_{i,j} = u(x_i, t_j)\,.$$

Sijoitetaan (10.56) ja (10.57) (10.55):een:

$$\frac{u_{i,j+1} - u_{i,j}}{k} = c^2\,\frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2}\,. \tag{10.58}$$

Merkitsemällä $r = c^2k/h^2$ saadaan

$$u_{i,j+1} = (1-2r)u_{i,j} + r(u_{i-1,j} + u_{i+1,j})\,, \tag{10.59}$$

joten (10.59) antaa $u_{i,j+1}$:n, kun $u_{i-1,j}$, $u_{i,j}$ ja $u_{i+1,j}$ tunnetaan.

Stabiliteettivaatimus saa nyt muodon

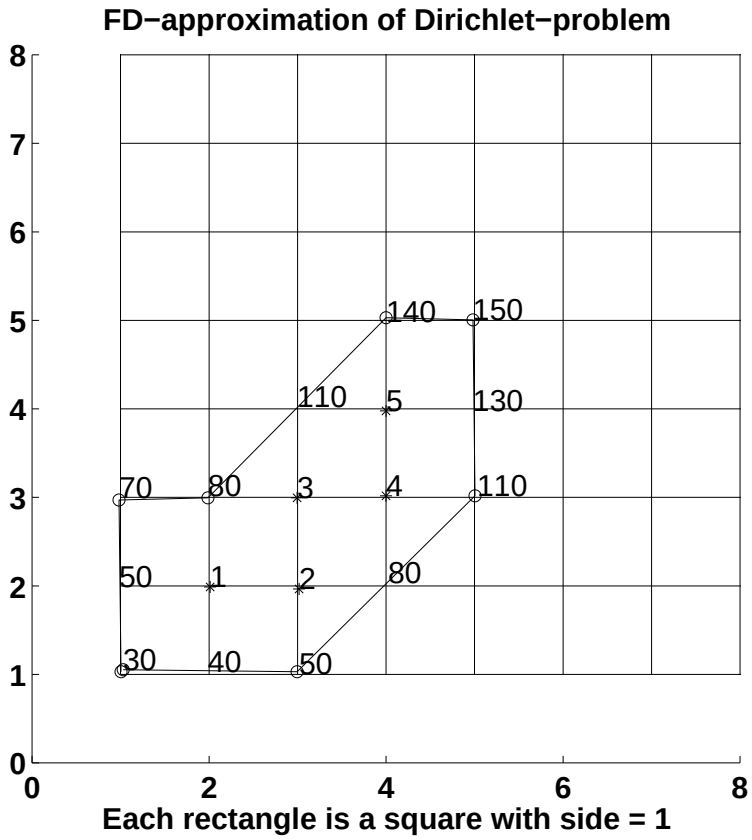
$$k \le h^2/(2c^2)\,.$$

10.60. Interaktiivinen Finite-difference-ohjelma MATLABilla. Alla interaktiivinen MATLAB-ohjelma finite-difference-menetelmää varten.

```
% FILE  fd2.m  begins.
% Design a polygonal region with vertices on a rectangular grid
% Each interior angle pi*c, c = 0.25, 0.5, 0.75
% Give temperatures on sides as Dirichlet boundary conditions
% and solve Laplace equation at interior grid points
% The last point has to be the same one as the first one
% (to make the region closed).
% MATLAB 4.0
clear; clf;
x= 1:10; y = 1:10;
axis('square');
axis([0 8 0 8 ]);
for m=1:10
    line([m m], [1 10]);
    line( [1 10] , [m m]);
end;
luku=input('Please enter number of corner points: ');
disp(['Please enter ' num2str(luku) ' points by clicking mouse:'])
[x,y] = ginput(luku);
hold on;
plot(x,y,'ko'), title('FD-approximation of Dirichlet-problem'),
xlabel('Each rectangle is a square with side = 1');
```

```
plot(x,y,'k');

n=input('Please enter the number of boundary temperatures : ');
for p=1:n
   t(p) =input('Please enter temperature: ');
%   gtext(num2str(t(p)));
end;
disp('Temperatures given: ');
disp(1:n)
disp(t)
ch =input('Enter 1 if you have corrections');
if (ch ==1)
n=input('Please enter the number of boundary temperatures : ');
for p=1:n
   t(p) =input('Please enter temperature: ');
%   gtext(num2str(t(p)));
end;
end
disp(['Now double click the mouse ' num2str(n) ' times '...
      'to give the places of the temperatures:']);
for p=1:n
  [x,y]=ginput(1);
  gtext(num2str(t(p)));
end;
disp('The temperature at a corner is the mean value of');
disp('the temperatures of the adjacent sides');
m=input('Please enter the number of unknowns: ');
disp(['Please enter ' num2str(m) ' points ' ...
      'by double clicking the mouse:'])
for p =1:m
      gtext(num2str(p));
    [x,y] =ginput(1);
    plot(x,y,'k*');
%    circle(x,y,0.03:0.02:0.1);
end;

a=-4*eye(m,m); b =zeros(m,1);
for j=1:m
   p=input(['Enter the number of the neighbors ' ...
            'of x(' num2str(j) ') :']);
   for r=1:p
```

```
      pp=input(['Enter the index of ' num2str(r) ':st neighbor'...
                ' of x(' num2str(j) ') :']);
      a(j,pp) =1;
   end;
   if (p < 4)
      disp(['s = sum of temperatures of the bdry values ' ...
            'of x(' num2str(j) ') :']);
      b(j) =-input('Enter s: ');
   end;
end;
disp([a b])
var= a\b;
var'
% FILE  fd2.m  ends.
```




```
>> disp(date)
29-Nov-1999
>> fd2
```

```
Please enter number of corner points: 8
Please enter 8 points by clicking mouse:
Please enter the number of boundary temperatures : 12
Please enter temperature: 30
Please enter temperature: 50
Please enter temperature: 70
Please enter temperature: 80
Please enter temperature: 110
Please enter temperature: 140
Please enter temperature: 150
Please enter temperature: 130
Please enter temperature: 110
Please enter temperature: 80
Please enter temperature: 50
Please enter temperature: 40
Temperatures given:
     1     2     3     4     5     6     7     8     9    10    11    12

    30    50    70    80   110   140   150   130   110    80    50    40

Enter 1 if you have corrections0
Now double click the mouse 12 times to give the places of the temperatures:
The temperature at a corner is the mean value of
the temperatures of the adjacent sides
Please enter the number of unknowns: 5
Please enter 5 points by double clicking the mouse:
Enter the number of the neighbors of x(1) :1
Enter the index of 1:st neighbor of x(1) :2
s = sum of temperatures of the bdry values of x(1) :
Enter s: 170
Enter the number of the neighbors of x(2) :2
Enter the index of 1:st neighbor of x(2) :1
Enter the index of 2:st neighbor of x(2) :3
s = sum of temperatures of the bdry values of x(2) :
Enter s: 130
Enter the number of the neighbors of x(3) :2
Enter the index of 1:st neighbor of x(3) :2
Enter the index of 2:st neighbor of x(3) :4
s = sum of temperatures of the bdry values of x(3) :
Enter s: 190
Enter the number of the neighbors of x(4) :2
Enter the index of 1:st neighbor of x(4) :3
Enter the index of 2:st neighbor of x(4) :5
```

```
s = sum of temperatures of the bdry values of x(4) :
Enter s: 190
Enter the number of the neighbors of x(5) :1
Enter the index of 1:st neighbor of x(5) :4
s = sum of temperatures of the bdry values of x(5) :
Enter s: 380
    -4     1     0     0     0  -170
     1    -4     1     0     0  -130
     0     1    -4     1     0  -190
     0     0     1    -4     1  -190
     0     0     0     1    -4  -380


ans =

    60    70    90   100   120

>> print -deps ../images/t5
```

Liite: Esimerkkiohjelmien Maple-versioita

Tässä liitteessä on seuraavien MATLAB-esimerkkiohjelmien Maple-vastineet: e102.m, e103.m, e104.m e105.m, e106.m, e107.m, e113.m, e118.m, e123.m, e205.m, e214.m, e302.m, e304.m, e306.m, e310.m, e315.m, e409.m, e419.m, e510.m, e519.m, e520.m, e626.m, e633.m, e818.m, e822.m, e916.m, e1010.m, e1012.m, e1014.m.

1.2. Lineaarisen yhtälöryhmän $Ax = b$ ratkaisu.

```
# FILE  e102  begins.
with(linalg):
A:=randmatrix(50,50):
b:=randvector(50):
x:=linsolve(A,b):
check:=norm(x,2);
evalf(check);
# FILE  e102  ends.
```

1.3. Minimin haku.

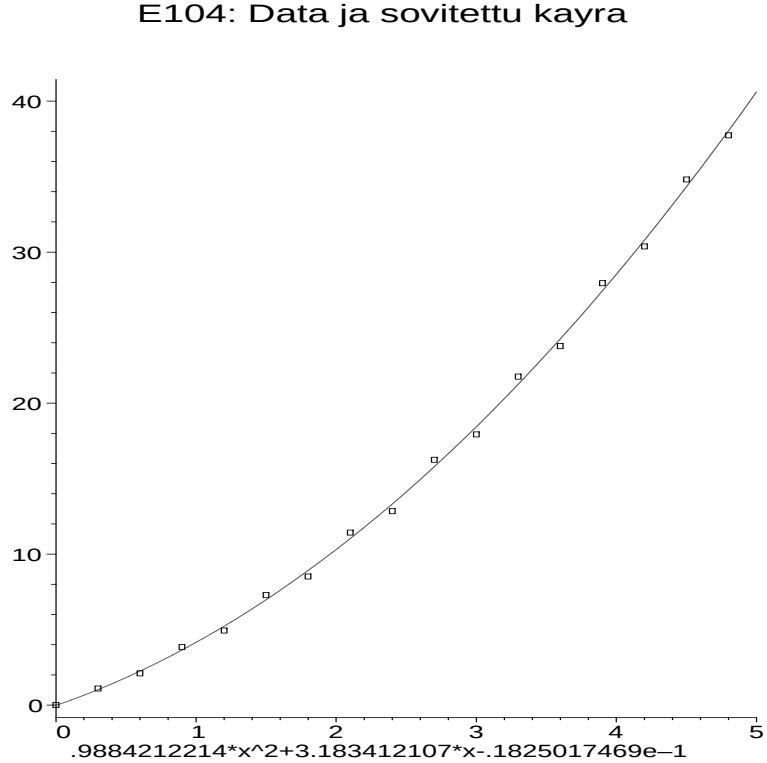
```
# FILE  e103  begins.
f[103]:= sin@tan + 1.2*cos;
plot(f[103], -1.7..1.7, title="Funktio esim. 1.3");
evalr(f[103](INTERVAL(-1.6..1.6)));
x:='x':
minimize(f[103](x), {x}, {x=-1.6..1.6});
readlib(extrema)(f[103](x), {},'s');
eqn:=diff(f[103](x),x);

min(op(map(f[103],[seq(fsolve(eqn,x = -1.6 + n*0.01),n=0..20)])));

# FILE  e103  ends.
```

1.4. Data-analyysi.

```
# FILE  e104  begins.
x:='x': y:='y':
a:='a': b:='b': c:='c':
with(stats):
with(plots):

X:= [seq(0.3*j,j=0..16)]:
```

```
Y:= map(x-> x^2+Pi*x+0.5*sin(10*x), X):
fit[leastsquare[ [x,y], y=a*x^2+b*x+c ]]([X,Y]);

p2:= rhs(%):
poly:= plot(p2,x=0..5):
points:= pointplot(zip( (x,y)->[x,y], X,Y), symbol=BOX):
display([poly, points], axes=FRAME,
                        labels = [sprintf('%a',p2), " "],
                        title = "E104: Data ja sovitettu kayra");
# FILE  e104  ends.
```

Polynomi on $.9884212214x^2 + 3.183412107x - .01825017469$.



1.5. Tavallisen differentiaaliyhtälön alkuarvotehtävän ratkaisu.

```
# FILE  e105  begins.
# Solves  y'' + a y' + b y = 0,  y(0) = 1, y'(0) = 0

x:='x': y:='y':
```

```
a:=-0.05: b:=0.15:
y0:=1.0:
dy0:=0.0:

dsolve({diff(y(x),x,x) + a*diff(y(x),x) + b*y(x) = 0,
        y(0) = y0, D(y)(0) = 0}, y(x));
# OR:
#dsolve({(D@@2)(y)(x) + a*D(y)(x) + b*y(x) = 0,
#        y(0) = y0, D(y)(0) = 0}, y(x));
plot(rhs(%), x=0..1, title="E105: Solution by dsolve");
# FILE  e105  ends.
```

1.6. Integraalin laskeminen.

```
# FILE  e106  begins.
f106:= sin^2 * (exp@sqrt);
a:= int(f106(x), x=0..Pi);
plot(f106, 0..Pi, title="Integral = ".(sprintf("%7.5f",evalf(a))));
# FILE  e106  ends.
```

1.7. Interpolointiongelma.

```
# FILE  e107  begins.
x:='x': y:='y': a:='a':
with(stats):
with(plots):
X:= [-1, 0, 1, 2]:
Y:= [0, 1, 0.5, 2]:
XY:= zip( (x,y)->[x,y], X,Y):
n:= nops(XY) - 1:           # = nops(X)-1 = nops(Y)-1

fit[leastsquare[ [x,y], y=sum(a[i]*x^i, i=0..n)]]( [X,Y] );
p:= unapply(rhs(%),x);
x[0]:= 1.5;
y[0]:= p(x[0]);

poly:= plot(p,-1..2):
datapoints:= pointplot(XY,symbol=DIAMOND):
interpoint:= pointplot([x[0],y[0]],symbol=BOX):
display([poly,datapoints,interpoint],
        title="E107: ".(sprintf("%a",'p'(x[0]))).
```

```
                        "=".(sprintf("%f",y[0])));
# FILE  e107  ends.
```

1.13. Esimerkki (Hypergeometrinen funktio). Seuraava ohjelma käyttää Maplen omaa funktiota hypergeometrisen funktion laskemiseen.

```
# FILE  e113  begins.
a:=0.5: b:=0.5: c:=1.0:
plot(hypergeom([a,b],[c],r), r=0..0.99,
     title="E113: F(a,b;c;r), a=0.5, b=0.5, c=1.0");
# FILE  e113.m  ends.
```

1.18. Ohjelma-epsilon.

```
# FILE  e118  begins.
r:= 'r':
test:= r -> arcsin(sin(r)) - r:
plot(test(r), r=0..1, title="E118: arcsin(sin(r)) - r");
# FILE  e118  ends.
```

1.23. Taylor-sarjan osasummat.

```
# FILE  e123  begins.
Digits:=16:
x:= -7.:
e:= exp(x):
t:='t':
for n from 2 by 2 to 38 do
    sums[n]:= eval( convert(taylor(exp(t),t=0,n+1),polynom), t=x)
od:
tbl:= [seq([2*n, sums[2*n], sums[2*n] - e], n=1..19)]:

dprintf:=proc()    # "Double Print Formatted"
         fprintf(args); printf(args[2..nargs]) end:

f:= fopen("e123.dat",WRITE,TEXT):
dprintf(f,"FILE  e123.dat  begins.\n\n"):
dprintf(f,"Terms  Partial sum       Error\n"):
for n to 19 do
    dprintf(f,"%2d %16.9f %16.5e\n", op(tbl[n]))
od:
```

```
dprintf(f,"\nThe answer is %10.9f\n",exp(x)):
dprintf(f,"FILE  e123.dat  ends.\n"):
fclose(f):

Digits:=10:
###writedata("e123.dat",tbl);
###writedata(terminal,tbl);
# FILE  e123  ends.
```

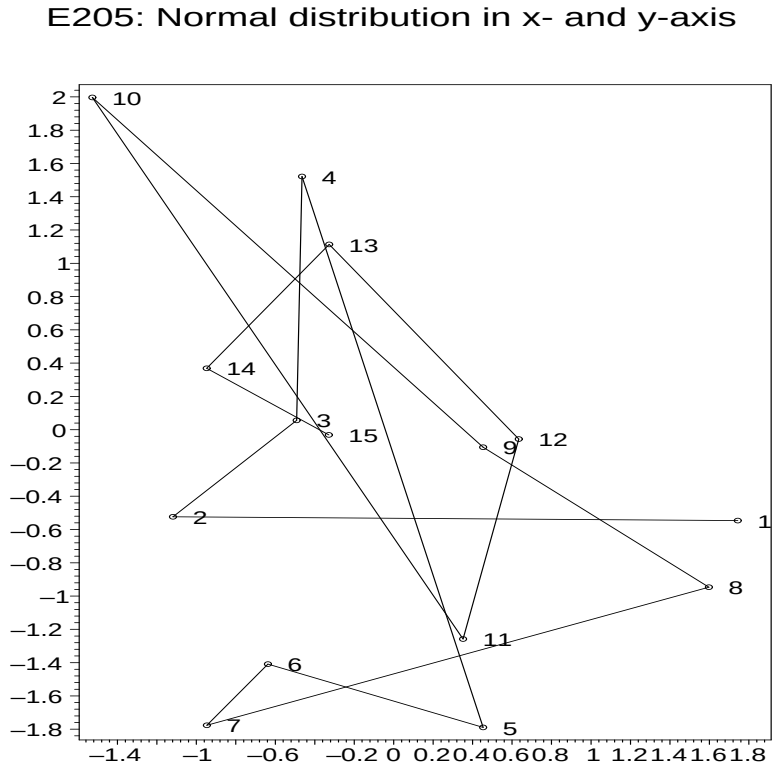
Ohjelma tuottaa seuraavan tiedoston:

```
FILE  e123.dat  begins.

Terms  Partial sum       Error
 2     18.500000000      1.84991e+01
 4     61.375000000      6.13741e+01
 6     84.718055556      8.47171e+01
 8     64.292881944      6.42920e+01
10     30.931765046      3.09309e+01
12     10.291680097      1.02908e+01
14      2.511955770      2.51104e+00
16       .469778134      4.68866e-01
18       .070092678      6.91808e-02
20       .009183674      8.27179e-03
22       .001729775      8.17893e-04
24       .000979884      6.80019e-05
26       .000916703      4.82104e-06
28       .000912177      2.94925e-07
30       .000911898      1.57279e-08
32       .000911883      7.37705e-10
34       .000911882      3.07055e-11
36       .000911882      1.17392e-12
38       .000911882      2.52973e-14

The answer is .000911882
FILE  e123.dat  ends.
```

2.5. Satunnaiskulku.

```
# FILE  e205  begins.
# with(stats):  with(random):
# ypt:= normald(n):
```

```
with(plots):
n:= 15:
ypt:= stats[random,normald](n):
xpt:= stats[random,normald](n):
xy:= [seq( [xpt[i], ypt[i]], i=1..n )]:

points:= pointplot(xy, symbol=CIRCLE):
lines:= pointplot(xy, connect=true):
texts:= textplot([seq( [xpt[i] + 0.1, ypt[i], `i`], i=1..n )],
                 align=RIGHT):
display([lines,points,texts],
        axes=BOXED,
        title="E205: Normal distribution in x- and y-axis");
# FILE  e205  ends.
```



2.14. Input.

```
# FILE  e214  begins.
x:= 'x':
```

```
###do
a:= readstat("Enter a real number a for a x + b : "):
b:= readstat("Enter a real number b for a x + b : "):

y:= x -> a*x + b:
plot(y, -3..3, title="Suora ".(sprintf("%a",'y'=a*x+b)));
plots[complexplot](sin(x + I*y(x)), x=-3..3,
                   title="E214: sin(z), kun ".
                   (sprintf("%a",Im('z')=a*Re('z')+b)),
                   labels=["Re(z)","Im(z)"]);
###od;
# FILE  e214  ends.
```

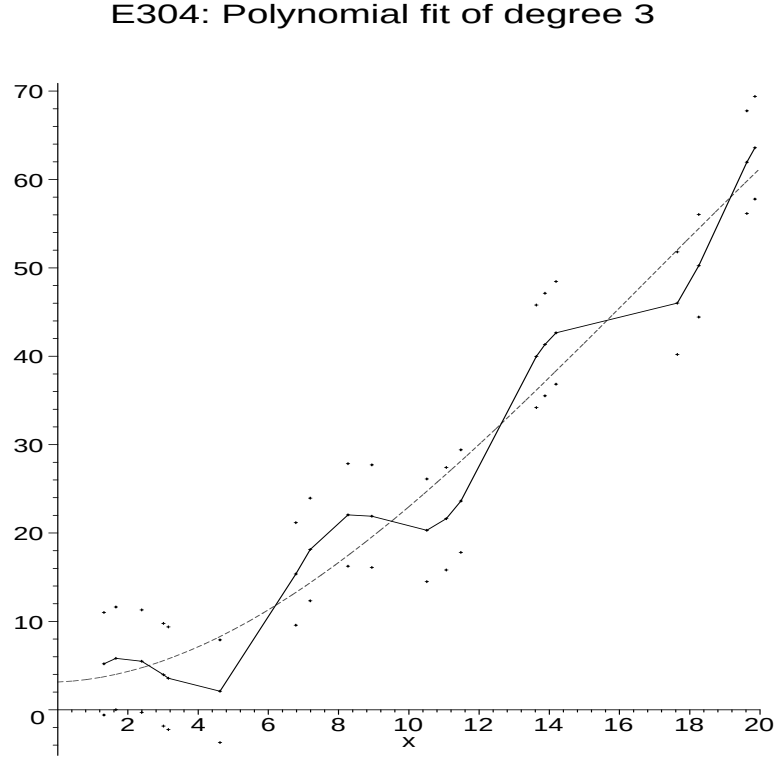
3.2. Kertoimien vaikutus juuriin.

```
# FILE  e302  begins.
# Study the Wilkinson polynomial
with(plots):
x:= 'x': k:= 'k':

p:= x -> mul(x-k, k=1..20):
for j from 0 to 10 do
   add_19:= -0.1*(j-5):
   points:= [solve(p(x) + add_19*x^19)]:
   plt[j]:= complexplot(points, style=POINT,
               title="E302: Wilkinson polynomial, a_19 = ".
               (sprintf("%a",-210+add_19))):
od:
display([seq(plt[j], j=0..10)], insequence=true);
# FILE  e302  ends.
```

3.4. Polynomisovitus.

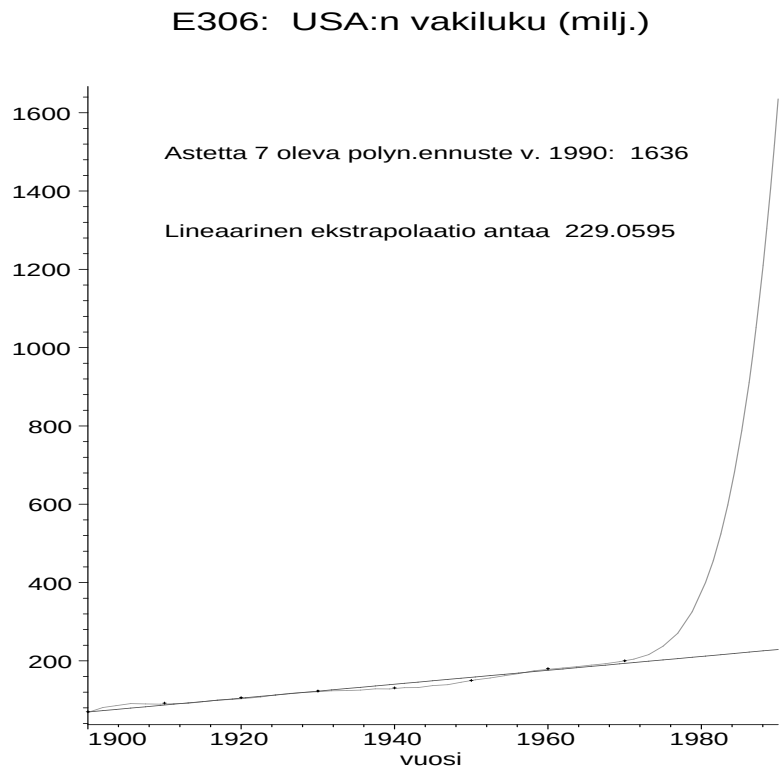
```
# FILE  e304  begins.
with(plots):
with(stats):

x:= 'x':
y:= x -> x*log(x) + 5*sin(x):
```

```
xpt:= 20*sort([ stats[random,uniform](20) ]):
ypt:= map(y,xpt):
xy:= zip((x,y)->[x,y], xpt, ypt):
err:= 0.3*describe[standarddeviation](ypt):
xyeu:= map((X,y)->[X[1],X[2]+y], xy, err):
xyel:= map((X,y)->[X[1],X[2]-y], xy, err):

a:= 'a': b:= 'b': c:= 'c': d:= 'd':
eqn:= fit[leastsquare[[x,y], y = a +b*x +c*x^2 +d*x^3, {a,b,c,d}]]
                    ([xpt,ypt]);

dataplot:= pointplot(xy,style=LINE):
polyplot:= plot(rhs(eqn), x=0..20, linestyle=3):
errplot:= pointplot([op(xy), op(xyeu), op(xyel)]):
display([dataplot,polyplot,errplot],
        title="E304: Polynomial fit of degree 3");
# FILE  e304  ends.
```



3.6. Esimerkki (Interpolaatio). Huomaa, että astetta 7 oleva polynomi on sama (tarkkaan ottaen tarkempi: itse asiassa täsmälleen oikea) kuin MATLABin antama tulos (josta Mathematica poikkesi melkoisesti).

```
# FILE  e306  begins.
with(stats):
with(plots):

year:= [seq(1900 + 10*i, i=0..7)]:
pop:= [70, 92, 106, 123, 131, 150, 180, 200]:
xy:= zip((x,y)->[x,y], year, pop):

x:= 'x':  a:= 'a':
eqn1:= fit[leastsquare[ [x,y] ]]([year,pop]):
eqn7:= fit[leastsquare[ [x,y], y = add(a[k]*x^k, k=0..7) ]]
                       ([year,pop]):

p17plot:= plot([rhs(eqn1), rhs(eqn7)], x=1900..1990,
               labels=["vuosi", ""]):
dataplot:= pointplot(xy):
txtplot:= textplot({[1910,1500,
                     "Astetta 7 oleva polyn.ennuste v. 1990:  ".
                     (sprintf("%.4g",subs(x=1990,rhs(eqn7))))],
                    [1910,1300,
                     "Lineaarinen ekstrapolaatio antaa  ".
                     (sprintf("%.4g",subs(x=1990,rhs(eqn1))))]},
                   align=RIGHT):

display([p17plot, dataplot, txtplot],
        title="E306:  USA:n vakiluku (milj.)");
# FILE  e306.m  ends.
```



3.10. Rungen ilmiö.

```
# FILE  e310  begins.
with(plots):
readlib(spline):

x:= 'x':
x1:= -5:  x2:= 5:
f:= x->1/(1+x^2):

for n from 3 to 14 do
  dx:= (x2-x1)/n:      # Use exact values: stats[fit] is sensitive
  xpt:= [seq(x1 + i*dx, i=0..n)]:
  ypt:= map(f,xpt):
  xy:= zip((x,y)->[x,y], xpt, ypt):

  a:= 'a':
  eqn:= stats[fit,leastsquare[ [x,y], y = add(a[k]*x^k, k=0..n) ]]
                ([xpt,ypt]):
```

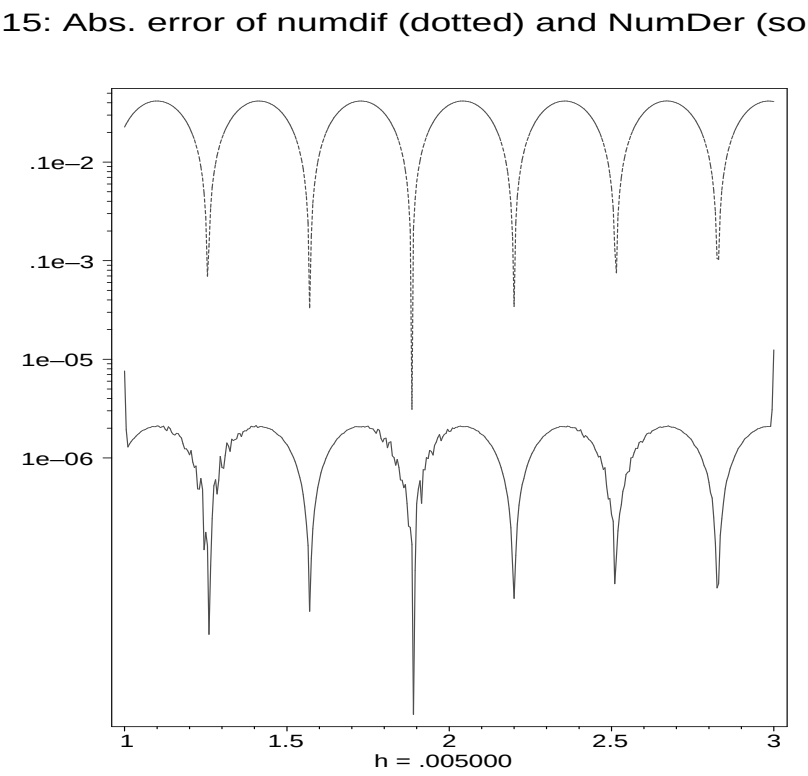
```
  spl:= spline(xpt,ypt,x):

  poly:= plot([rhs(eqn),f(x)], x=x1..x2, color=[red,blue]):
  spli:= plot([unapply(spl,x),f], x1..x2, color=[red,blue]):
  data:= pointplot(xy):
  text:= textplot(
         {[0,7.7,cat(sprintf("%d",n+1)," tukipistetta tasavalein")],
          [0,-3.5,""]},  # To adjust bottom of figure
          font=[HELVETICA,10]):

  ruframe[n]:= display([poly, data, text], title=
                 "E310a: Polynomiaproksimaatio 1/(1+x^2):lle: Rungen ilmio",
                       labels=["",""]):
  spframe[n]:= display([spli, data, text], title=
                       "E310b: Spliniaproksimaatio 1/(1+x^2):lle",
                       labels=["",""]):
od:
rufig:= display([seq(ruframe[n], n=3..14)],
                insequence=true, labels=["",""]):

spfig:= display([seq(spframe[n], n=3..14)],
                insequence=true, labels=["",""]):




#        PlotLabel->"E310a: Rungen ilmio, polynomiaproksimaatio",

#           PlotLabel->"E310b: spline-aproksimaatio",


#b:= animate(cos(t*x), x=0..Pi, t=0..1, color=blue,scaling=CONSTRAINED):
display(array(1..2,[rufig,spfig]));
# ,title=["E310: lausekkeen 1/(1+x^2) tasavalinen approksimointi",""]);
# FILE  e310  ends.
```

3.15. Numeerinen derivointi. Ohjelma e315 käyttää seuraavaa funktiota numeeriseen derivointiin:

```
# FILE  numder  begins.
```

```
# NumDer(y,h) gives a numerical approximation for the derivative
# of f for equally spaced data y:= [seq(f(x+j*h), j=1..m)], m >= 5.
# The 5-point-rule numerical differentiation coefficients,
# from Abramowitz-Stegun, 25.3.6, are used.  To compute
# dy0 = f'(x0) at a single point x0 set, for example, h:= 0.001,
# x:= [seq(x0+i*h, i=-2..2)]; y:= map(f,x); dy:= NumDer(y,h);
# dy0:= dy[3];

NumDer:=proc(flst::list,h)
local l,ta,tb,tc,td,te, ti,tp0,tp,tp2,tp3;
   l:=nops(flst);
   if l<5 then ERROR(
      `The first argument (list) must have at least 5 members!`) fi;
   ti:=[$1...l];
   tp0:=[1$l];
   tp:=[-2,-1,0$l-4,1,2];   # DOES NOT WORK for flist short ( <4 )
   tp2:=[4,1,0$l-4,1,4];
   tp3:=[-8,-1,0$l-4,1,8];
   ta:=(2*tp3 - 3*tp2 - tp + tp0)/12;
   tb:=(4*tp3 - 3*tp2 - 8*tp + 4*tp0)/6;
   tc:=(2*tp3 - 5*tp)/2;
   td:=(4*tp3 + 3*tp2 - 8*tp - 4*tp0)/6;
   te:=(2*tp3 + 3*tp2 - tp - tp0)/12;

   evalf((+zip(times,ta,map(proc(i,L) L[i] end, map(max,ti-tp-2*tp0,1),flst))
#,zip(times,ta,map(flst,map(max,ti-tp-h*tp0,1)));
#,zip(times,ta,map(i-> flst[i],map(max,ti-tp-h*tp0,1)));
#,zip(times,ta,map(proc(i) flst[i] end, map(max,ti-tp-h*tp0,1)));
#,seq(ta[j]*flst[j],j=map(max,ti-tp-h*tp0,1));
   -zip(times,tb,map((i,L)->L[i], map(max,ti-tp-tp0,1), flst))
   +zip(times,tc,map((i,L)-> L[i], map(max,ti-tp,1), flst))
   -zip(times,td,map((i,L)->L[i], map(max,ti-tp+tp0,1), flst))
   +zip(times,te,map((i,L)->L[i], map(max,ti-tp+2*tp0,1), flst)))/h)
end:

times:=(x,y)-> x*y:   # The auxilliary function used in NumDer.
# FILE  numder  ends.

# FILE  e315  begins.
with(plots):
read numder:
```

```
numdif:= (f,x,h) -> (f(x+h) - f(x-h))/(2*h):
e315f:= x -> sin(x) + cos(10*x):

h:= readstat("Enter h (e.g. = 0.02) : "):
x:= [seq(1 + h*r, r=0..floor(2/h))]:
z1:= zip( abs@((i,j) -> i-j),
          map2( numdif,e315f, x,h), map( D(e315f), x) ):
dy:= NumDer(map( e315f, x),h):
z2:= zip( abs@((i,j) -> i-j), dy, map( D(e315f), x) ):
y0:= min(op(z1),op(z2)):  y1:= max(op(z1),op(z2)):

xy1:= zip( (i,j) -> [i,j], x, z1):
xy2:= zip( (i,j) -> [i,j], x, z2):

c1:= logplot(xy1, linestyle=3):

#    PlotRange->{y0,y1},

c2:= logplot(xy2):
#    PlotRange->{y0,y1},

display([c1, c2], axes=BOXED, title=
    "E315: Abs. error of numdif (dotted) and NumDer (solid)",
    labels=["h = ".(sprintf("%f",h)),""]);
# FILE  e315  ends.
```



4.9. Gaussin eliminointi.

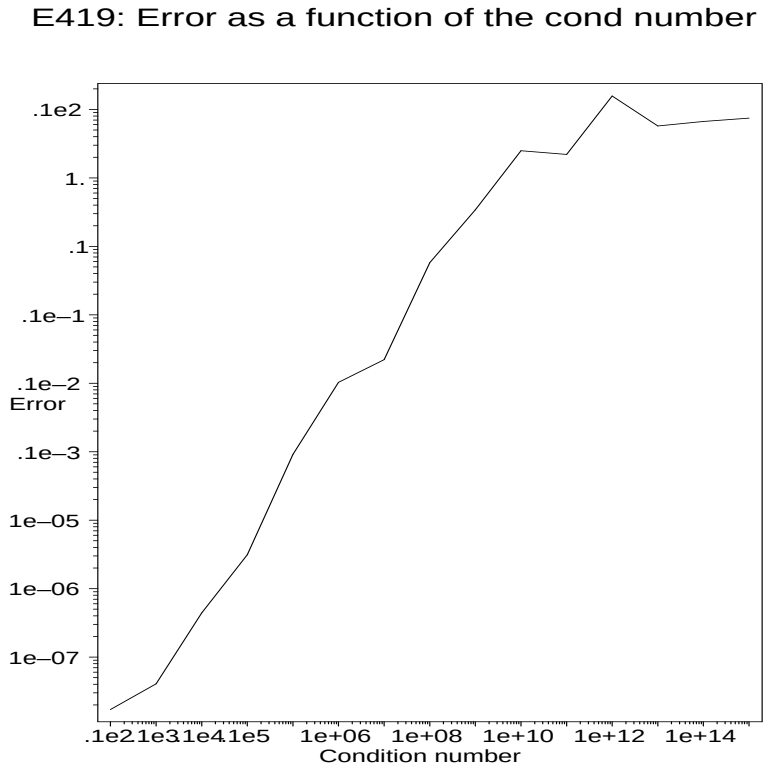
```
# FILE  e409  begins.
# Gauss elimination without pivoting.
with(linalg):

n:= readstat("Enter an integer n>=2 : "):
if n< 6 then small:=true else small:=false fi:

A:= 0.001*randmatrix(n,n,entries=rand(0..1000)):
x:= vector([seq(i,i=1..n)]):                  ###x:= randvector(n):
b:= A&*x:                ###b:=randvector(n,entries=rand(0..1000)):
C:= blockmatrix(1,2,[A,b]):   # augment?
if small then print(‘Extended matrix‘, ’C’=eval(C)); print(‘ ‘) fi:

for j to n-1 do
  for i from j+1 to n do
    if abs(C[j,j]) < 10^(-10) then
      print(‘Bad matrix, cond = ‘, cond(A,2));
```

```
      break
    fi;
    C:= addrow(C, j, i, -C[i,j]/C[j,j]);     # Add to row i
   if small then print(C); print(` `) fi
  od;
od;

sol:= vector(n):
sol[n]:= C[n,n+1]/C[n,n]:
for i from n-1 to 1 by -1 do
#  s:= 0;
#  for j from i+1 to n do
#    s:= s + C[i,j]*sol[j]
#  od;
#  sol[i]:= (C[i,n+1]-s)/C[i,i];
   j:= 'j';
   sol[i]:= (C[i,n+1] - sum(C[i,j]*sol[j], j=i+1..n))/C[i,i]
od:                                # pivot?
print(`Solution = `, sol);
print(`Should be = `, x);
print(`Error = `, norm(x-sol,2));

bb:= A&*sol:
print(`A x solution = `, evalm(bb));
print(`Should be = `, evalm(b));
print(`Residual = `, norm(b-bb,2));

print(`Difference from Maple solution = `,
      norm(linsolve(A,b) - sol));
# FILE  e409  ends.
```

4.25. Kuntoisuusluvun muuttaminen. Maplessa *on* valmiina kuntoisuusluku.

```
# FILE  e418  begins.
with(linalg):

n:= 5:
a:= 0.001*randmatrix(n,n,entries=rand(0..1000)):
a:= evalm(a):
c:= readstat("Enter a number c>1 : "):
```

```
s:= evalf(Svd(a,u,v)):
ss:= seq(s[1] - (j-1)*s[1]*(1 - 1./c)/(n-1), j=1..n):
aa:= u&*diag(ss)&*transpose(v):
aa:= evalm(aa):
print("The matrix"):
print(aa):
printf("\nhas condition number = %f", cond(aa,2)):
# FILE  e418  ends.
```



4.19. $\mathrm{cond}(a)$**:n vaikutus tarkkuuteen.**

```
# FILE  e419  begins.
with(linalg):
with(plots):

n:= 20:
a:= evalm(0.001*randmatrix(n,n,entries=rand(1...1000))):
s:= evalf(Svd(a,u,v)):
```

```
data:= NULL:
for p to 15 do
  c:= 10.0^p;
  ss:= seq(s[1] - (j-1)*s[1]*(1 - 1./c)/(n-1), j=1..n);
  aa:= evalm( u &* diag(ss) &* transpose(v) );
  x:= vector([1.$n]);
  b:= evalm(aa&*x);
  numsol:= linsolve(aa,b);
  d:= abs(norm(numsol-x,2));
  data:= data, [c,d]
od:

loglogplot([data], axes=BOXED,
           title="E419: Error as a function of the cond number",
           labels=["Condition number","Error"]);
# FILE  e419  ends.
```

5.10. Esimerkki (kiintopisteiteraatio).

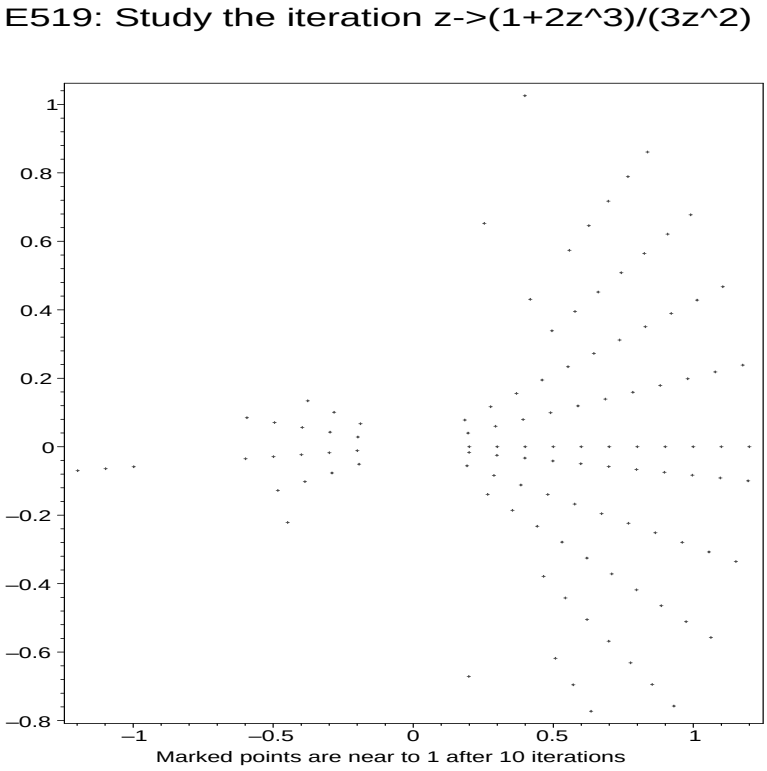
```
# FILE  e510  begins.
#  Fixed point iteration for the linear system
#  x = 0.2x   - 0.1y  + 0.2z + 0.8
#  y = -0.05x + 0.1y  + 0.3z - 0.5
#  z = -0.1x  - 0.05y + 0.2z - 0.1

with(linalg):

a:= matrix(3,3,[0.2, -0.1,  0.2,
                -0.05, 0.1,  0.3,
                -0.1, -0.05, 0.2]):
b:= vector([0.8, -0.5, -0.1]):
v:= vector([0.55, 0.1, 1.]):

F:= matrix(3,3, [8, 1, -2, 1, 18, -6, 2, 1, 16]):
G:= vector([8, -10, -2]):
u:= linsolve(F,G);
                                # For an unknown reason,
data:= [evalm(v)]:              # data:=evalm(v); here
for i to 10 do
  v:= evalm(a&*v + b);
  data:= [op(data),evalm(v)]  # and data:=data,evalm(v) here
```

```
od:                              # does not work!


# AGAIN, WITH DIFFERENT DATA REPRESENTATION:
v:= vector([0.55, 0.1, 1.]):
data:= op(convert(v,list)):
for i to 10 do
  v:= evalm(a&*v + b);
  data:= data,op(convert(v,list))
od:


f:= fopen("e510.dat",WRITE):
writeline(f, "FILE  e510.dat  begins."):
writeline(f, " x               y               z"):
writedata(f, matrix(11,3,[data])):
writeline(f):
unumer:= evalf(op(convert(u,list))):
writeline(f, "Maple solution to original system = ".
             (sprintf("[%.10f,%.10f,%.10f]", unumer))):
writeline(f, "Error = ".(sprintf("%e",
                         norm(vector([data[31..33]])-u,2)))):
writeline(f, "FILE  e510.dat  ends."):
fclose(f);

do
  line:=readline("e510.dat");    #TEST IF EXISTS !!!
  if line=0 then break fi;
  printf("%s\n",line)
od:
# FILE  e510  ends.
```

5.19. Huomautuksia (Newtonin menetelmä).

```
# FILE  e519  begins.
with(plots,complexplot):

#z:= 'z':
it:= z-> (1.+2.*z^3)/(3.*z^2):
arenear:= proc(i) if evalf(abs(z10[i]-1))<0.01 then true else false fi end:
w:= NULL:
```

```
for r from 0.1 by 0.1 to 1.2 do
  z:= [seq(r*exp(I*the), the=[seq(0.2*i, i=0..2*evalf(Pi)/0.2)] )];
  z10:= map(it@@10,z);
  ind:= [seq(i, i=1..nops(z))];
  w:= w, seq(z[i], i=select(arenear,ind));
od:

complexplot([w], style=POINT, axes=BOXED,
  title="E519: Study the iteration z->(1+2z^3)/(3z^2)",
  labels=["Marked points are near to 1 after 10 iterations",""],
  font=[HELVETICA,12]);

#xy={Re[#],Im[#]}& /@ w;     WAS NOT NEEDED !

# FILE  e519  ends.
```



5.20. Numeerinen Jacobin matriisi.

```
# FILE  e520  begins.
```

```
with(plots):
with(linalg):

numjaco:= proc(f,x)
local h,m,l,j,e;
  h:= 10.0^(-4);
  m:= [];
  l:= nops(convert(x,list));
  for j to l do
    e:= vector([0$l]);  e[j]:= 1;
    m:= [op(m), evalm((f(evalm(x+h*e)) - f(evalm(x-h*e)))/(2*h))]
  od:
  transpose(matrix(m))
end:

e520f:= x -> evalf(vector([3*x[1] - cos(x[2]*x[3]) - 0.5,
             x[1]^2 - 81*(x[2]+0.1)^2 + sin(x[3]) + 10.6,
             exp(-x[1]*x[2]) + 20*x[3] + (10*Pi - 3)/3]));

x0:= vector([3,4,5]);
data:=   [[0, op(convert(x0,list)), norm(e520f(x0),2)]]:
x1:= x0:
for j to 9 do
  x1:= evalm(x1 - linsolve(numjaco(e520f,x1), e520f(x1)));
  data:= [op(data),
          [j, op(convert(x1,list)), norm(e520f(x1),2)]]
od:

li:= pointplot3d(map(L->L[2..4], data), style=LINE, color=black):
pt:= pointplot3d(data[10][2..4], color=black, symbol=BOX):
display([li,pt], axes=BOXED, orientation=[-50,50],
        title="E520a: Newton iteration, end point marked",
        labels=["x1","x2","x3"],font=[HELVETICA,12]);

logplot(map(L->[L[1],L[5]], data),
   axes=BOXED,
   title="E520b: Function norm",
   labels=["Number of Newton iterations",""],font=[HELVETICA,12]);

f:= fopen("e520.dat", WRITE):
writeline(f, "FILE  e520.dat  begins."):
```

```
writeline(f, "i            x            y            z           norm(f)"):
writedata(f, data):
writeline(f, "FILE  e520.dat  ends."):
fclose(f):

do
  line:=readline("e520.dat");;    #TEST IF EXISTS !!!
  if line=0 then break fi;
  printf("%s\n",line)
od:
# FILE  e520  ends.
```

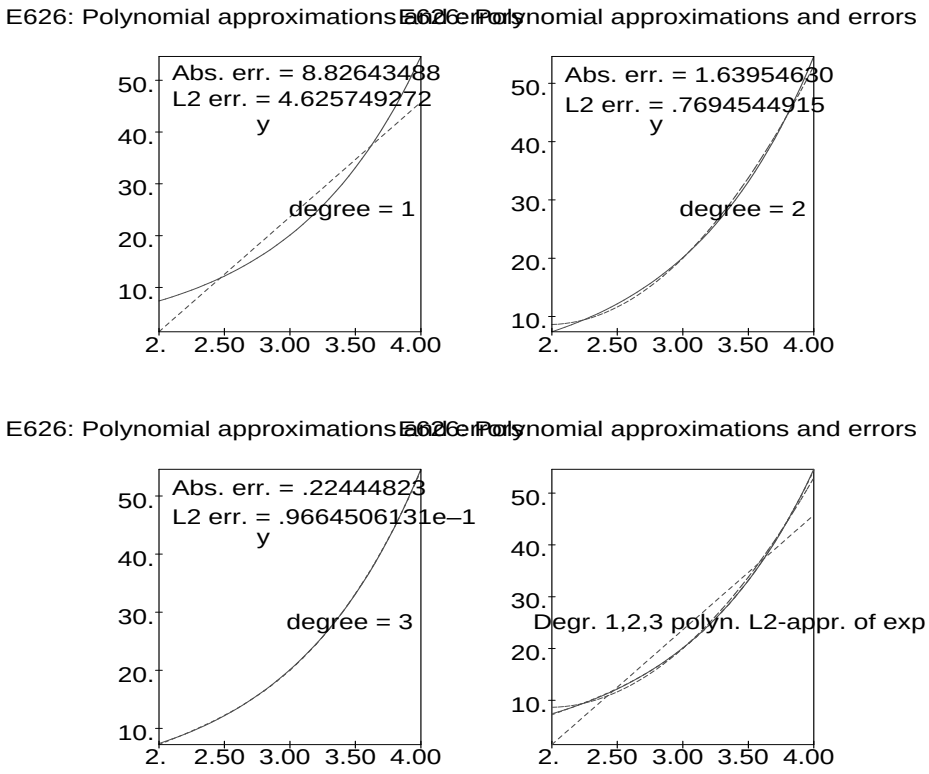
Ohjelma tuottaa seuraavan tulostiedoston:

```
FILE  e520.dat  begins.
i            x            y            z           norm(f)
0               3               4               5               1347.447633
1               9.950053969     2.038499585     -.473599085     262.8605363
2               .524524722      .742886910      -.4735987755    47.13315078
3               .4949074142     .3972533776     -.5143474504    9.675171919
4               .4971709115     .2771453778     -.5170791451    1.168496382
5               .4970474887     .2580112078     -.5175788059    .0296553901
6               .4970438097     .2574996297     -.5175919442    2.1190063450e-05
7               .497043807      .2574992636     -.5175919542    1.0000000000e-10
8               .497043807      .2574992636     -.5175919542    1.0000000000e-10
9               .497043807      .2574992636     -.5175919542    1.0000000000e-10
FILE  e520.dat  ends.
```

6.26. Huomautus (ortogonaaliset polynomit).

```
# FILE  e626  begins.
with(plots):
with(linalg,linsolve):

nmax:= 4:
g:= exp:
r1:= readstat("Enter r1 : "):
r2:= readstat("Enter r2 > r1 : "):

xi:= [seq(r1+(r2-r1)/40.0*i, i=0..40)]:
y1:= map(g,xi):
xy1:= zip( (i,j)->[i,j], xi,y1):
```

```
orig:= plot(xy1):

for n from 2 to nmax do
  a:= matrix(n,n, [seq(seq(
                  (r2^(2*n-k-j+1.) - r1^(2*n-k-j+1))/(2*n-k-j+1),
                  k=1..n), j=1..n)] ):
  b:= vector( [seq(int(g(x)*x^(n-j), x=r1..r2), j=1..n)] );
  c:= linsolve(a,b);

  y2:= map(x->(sum(c[k]*x^(n-k), k=1..n)), xi);
  virhe:= map(abs, zip( (x,y)->x-y, y2,y1));
  maxvirhe:= max(op(virhe));
  int2:= sqrt((r2 - r1)/40.0*
              (add(x^2,x=virhe) - (virhe[1]^2+virhe[-1]^2)/2));

  xy2:= zip( (i,j)->[i,j], xi,y2):
  appr[n]:= plot(xy2, linestyle=n):
  texta:= textplot([r1 + (r2-r1)/20, g(r2) - 3,   ### MUST BE ...
                  "Abs. err. = ".(sprintf("%a",maxvirhe))],
                                                    align=RIGHT):
  textl:= textplot([r1 + (r2-r1)/20, g(r2) - 8,   ### ... ADJUSTED
                  "L2 err. = ".(sprintf("%a",int2))],align=RIGHT):
  fig[n]:= display([orig, appr[n], texta, textl],
                  axes=BOXED,
                  labels=["degree = ".(sprintf("%a",n-1)),"y"])
od:

fig234:=display([orig, appr[2], appr[3], appr[4]],
                axes=BOXED,
                labels=["Degr. 1,2,3 polyn. L2-appr. of exp",""]):


display(array(1..2,1..2, [[fig[2],fig[3]], [fig[4],fig234]]),
        title="E626: Polynomial approximations and errors");
# FILE  e626  ends.
```



6.33. Esimerkki (Gramin-Schmidtin ortogonalisointi).

```
# FILE  e633  begins.
with(linalg):

vl2m:=proc(VL)   # convert from vector list to matrix
      convert(map(convert,VL,'list'),matrix)
end:

Gram:= proc(x)      #  x should be a list of vectors
local dim, y, n;
  dim:= nops(x);
  y:= [seq(0, n=1..dim)];
  y[1]:= x[1];
  for n from 2 to dim do
    y[n]:= evalm(x[n] -
           add( dotprod(x[n],y[k])*y[k]/norm(y[k],2)^2 , k=1..n-1))
  od;
  y
```

```
end:

dim:= 3:  #20:
x:= evalm(0.001*randmatrix(dim,dim,entries=rand(0..1000))):
x:= map(L->vector(L), convert(x,listlist)):
y:= Gram(x); yy:= GramSchmidt(x);
z:= Gram(y):
print("y := Gram(x); norm(y-Gram(y)) = ",  # CLAIM NOT RIGHT
      norm(vl2m(z) - vl2m(y),frobenius));
# FILE  e633  ends.
```

Vertailun vuoksi on käytetty myös Maplen paketissa `linalg` olevaa valmista ortogonalisointifunktiota `GramSchmidt`.

8.18. Esimerkki (Fourier-muunnos).

```
# FILE  e818  begins.
with(plots):
with(linalg):

data:=[]:
for n from 2 by 2 to 12 do
   w:=  exp( I*2*Pi/n);
   wc:= exp(-I*2*Pi/n);
   F:=  matrix(n,n, (p,q)->evalf(w ^((p-1)*(q-1))) );
   FC:= matrix(n,n, (p,q)->evalf(wc^((p-1)*(q-1))) );
   c:= norm(1/n*FC&*F - array(1..n,1..n,identity), frobenius);
   data:= [op(data), [n,c]]     # or band([1], n)
od:

logplot(data, title="E718:");
# FILE  e818  ends.
```

8.22. Esimerkki. Maplessa ei liene vastinetta MATLABin flops-muuttujalle, joten tehtyjen laskutoimitusten määrää ei saada selville.

```
# FILE  e822  begins.
with(plots):
with(linalg):

data:=[]:
```

```
for n from 2 by 10 to 60 do
  x:= evalm(0.001*randvector(n,entries=rand(0..1000))):
  w:= exp( I*2*Pi/n);
  F:= matrix(n,n, (p,q)->evalf(w^((p-1)*(q-1))) );
  y:= F&*x;

  m:= n/2;
  wm:= exp( I*2*Pi/m);
  Fm:= matrix(m,m, (p,q)->evalf(wm^((p-1)*(q-1))) );
  x1:= vector([seq( x[2*j-1], j=1..m )]);
  x2:= vector([seq( x[2*j  ], j=1..m )]);
  y1:= evalm(Fm&*x1);
  y2:= evalm(Fm&*x2);
  jm:= [$1..m];
  ym:= [op(map(j -> y1[j] + w^(j-1) * y2[j], jm)),   # Concatenate
        op(map(j -> y1[j] - w^(j-1) * y2[j], jm))];  #  two lists.

  data:= [op(data), [n, norm(y-ym,2)]]
od:

logplot(data, axes=BOXED,
        title="E722: n x n -matriisin FFT:n tarkkuus");
# FILE  e822  ends.
```

9.16. Esimerkki (Differenssimenetelmä).

```
# FILE  e916  begins.
with(linalg):
with(plots):

a:= 1.:
b:= 2.:
N:= 10:
h:= (b-a)/N:
bval1:= 1.:  bval2:= 2.:
p:= x -> -2./x:
q:= x -> 2./x^2:
r:= x -> sin(log(x))/x^2:

tt:= [seq(a + j*h, j=0..N)]:
t:= tt[2..-2]:
```

```
n:= nops(t):
diagon:= map( -(2. + h^2*q), t):
mtx:= diag(op(diagon)):
pp:= map(p, t):
for i to n-1 do
  mtx[i,i+1]:= 1 - 0.5*h*pp[i]
od:
for i to n-1 do
  mtx[i+1,i]:= 1.+0.5*h*pp[i+1]
od:

vcr:= map(h^2*r, t):   # The rhs of the lin. equation
vcr[1]:= vcr[1] - bval1*(1 + (-2/(a+h))*0.5*h):
vcr[n]:= vcr[n] - bval2*(1 - (-2/(b-h))*0.5*h):
sol:= linsolve(mtx, vcr):
xy:= zip( (x,y)->[x,y], t, convert(sol,list)):

c2:= (1/70.)*(8. - 12*sin(log(2.)) - 4*cos(log(2.))):
exact:= x ->
        (1.1-c2)*x + c2/(x*x) - 0.3*sin(log(x)) - 0.1*cos(log(x)):

fig:= plot(exact, a..b):
points:= plot(xy, style=POINT):
fig1:= display([fig, points], axes=BOXED, title=
               "E916a: Numerical (points) and exact solutions"):
fig1;

error:= zip( (x,y)->[x,y], t, convert(sol,list)-map(exact,t) ):
fig2:= plot(error, title="E916b: Error"):
fig2;

tbl:= concat(sol, vector(map(exact,t)), sol-vector(map(exact,t)));
f:= fopen("e916.dat", WRITE):
writeline(f, "FILE  e916.dat  begins."):
writeline(f, "numerical    exact        error"):
writedata(f, tbl):
writeline(f, "FILE  e916.dat  ends."):
fclose(f):

do
  line:=readline("e916.dat");;    #TEST IF EXISTS !!!
```

```
  if line=0 then break fi;
  printf("%s\n",line)
od:
# FILE  e916  ends.
```

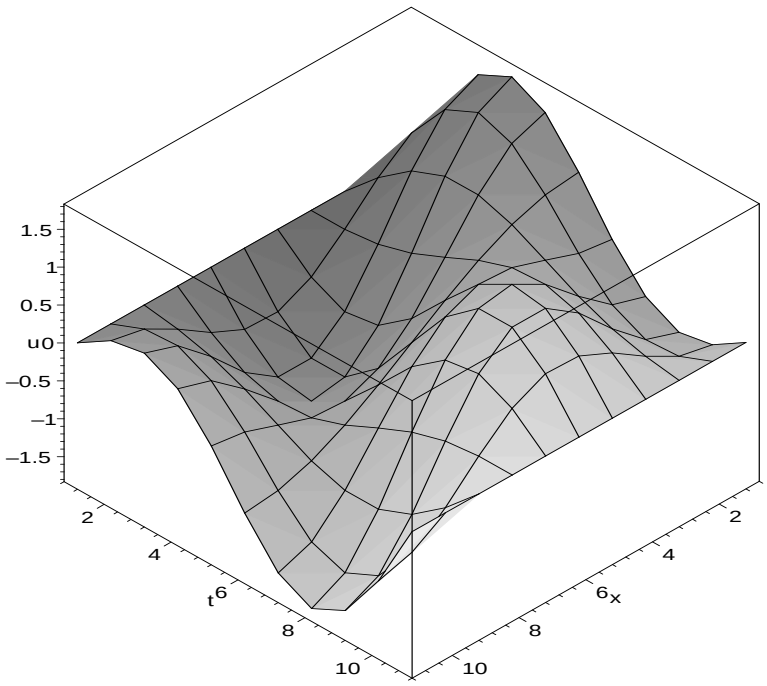
Ohjelma tuottaa seuraavan tiedoston:

```
FILE  e916.dat  begins.
numerical    exact        err
1.092600519     1.092629298     -2.8779e-05
1.187043127     1.187084842     -4.1715e-05
1.283336869     1.283382364     -4.5495e-05
1.381402046     1.381445951     -4.3905e-05
1.481120264     1.481159417     -3.9153e-05
1.582359898     1.582392461     -3.2563e-05
1.68498902      1.685013961     -2.4941e-05
1.788881747     1.788898534     -1.6787e-05
1.8939211       1.893929509     -8.409e-06
FILE  e916.dat  ends.
```

10.10. Esimerkki (aaltoyhtälön ratkaisu).

```
# FILE  e1010  begins.
with(plots):
with(linalg):

f:= x -> evalf( sin(Pi*x) + sin(2*Pi*x) ):
###g:= 0.;

a:= 1.: b:= 0.5:
c:= 2.: Un:= 11:
m:= 11: n:= 11:
###U:= matrix(m, n, 0);
h:= 1./(n-1):
xx:= [seq(j*h, j=0..n-1)]:
U[1]:= map(f,xx):
U[2]:= [0, op(map(.5*f, xx[1..n-2]) + map(.5*f, xx[3..n])), 0]:
for j from 3 to m do
  U[j]:= [0, op(U[j-1][3..n] + U[j-1][1..n-2] - U[j-2][2..n-1]), 0]
od:

matrixplot(array( [seq(U[j], j=1..m)] ),
```

```
          axes=BOXED, labels=["x","t","u"], font=[HELVETICA,12],
          title=
          "E1010: The finite diff. solution to the wave eq.");
# FILE  e1010.m  ends.
```



10.12. Esimerkki (Dirichletin ongelma).

```
# FILE  e1012  begins.
with(linalg):

a:= band([1,0,1,-4,1,0,1], 9):
a[3,4]:= 0: a[4,3]:= 0: a[6,7]:= 0: a[7,6]:= 0:
b:= vector([-100,-20,-20,-80,0,0,-260,-180,-180.]):  # dot!
sol:= linsolve(a, b);

eqn:= augment(a,b);
f:= fopen("e1012.dat", WRITE):
writeline(f, "FILE  e1012.dat  begins."):
writeline(f, "Dirichlet'n probleema, eqn ="):
```

```
writedata(f, eqn):
writeline(f, "sol ="):
writedata(f, sol):
writeline(f, "FILE  e1012.dat  ends."):
fclose(f):

do
  line:=readline("e1012.dat");    #TEST IF EXISTS !!!
  if line=0 then break fi;
  printf("%s\n",line)
od:
# FILE  e1012  ends.
```

Ohjelma tuottaa seuraavan tiedoston:

```
FILE  e1012.dat  begins.
Dirichlet'n probleema, eqn =
-4              1               0               1               0
1               -4              1               0               1
0               1               -4              0               0
1               0               0               -4              1
0               1               0               1               -4
0               0               1               0               1
0               0               0               1               0
0               0               0               0               1
0               0               0               0               0
sol =
55.71428574
43.21428575
27.14285717
79.64285718
70.00000009
45.35714290
112.8571429
111.7857144
84.28571434
FILE  e1012.dat  ends.
```

10.14. Esimerkki (SOR-menetelmä).

```
# FILE  e1014  begins.
with(linalg):
```

```
with(plots, matrixplot):

n:= 8:  m:= 8:
K:= (1/4)*(180. + 20. + 80. + 0.):
#u:= matrix(m,n,K):
u:= [[K$n]$m]:
u[1]:= [20.$n]:
u[m]:= [180.$n]:
for j from 2 to m-1 do u[j,1]:= 80. od:
for j from 2 to m-1 do u[j,n]:= 0. od:
u[1,1]:= 50.:   u[1,n]:= 10.:
u[m,1]:= 130.:  u[m,n]:= 90.:

omega:= evalf(4/(2 + sqrt(4+(cos(Pi/(n-1)) + cos(Pi/(m-1)))^2)));
for ite to 40 do
  for j from 2 to n-1 do
    for i from 2 to m-1 do
      r:= u[i+1,j] + u[i-1,j] + u[i,j+1] + u[i,j-1] - 4*u[i,j];
      u[i,j]:= u[i,j] + 0.25*r*omega
    od
  od
od:

f:= fopen("e1014.dat", WRITE):
writeline(f, "FILE  e1014.dat  begins."):
writeline(f, "Dirichlet'n probleema SOR-menetelmalla"):
writeline(f, "u ="):
writedata(f, u):
writeline(f, "FILE  e1014.dat  ends."):
fclose(f):

matrixplot(u, axes=BOXED, labels=["x","y","u"],
           title="E1014: SOR-method for Diriclet problem");
do
  line:=readline("e1014.dat");    #TEST IF EXISTS !!!
  if line=0 then break fi;
  printf("%s\n",line)
od:
# FILE  e1014  ends.
```

Viitteet

Esitiedoiksi:

[LP] A. Lahtinen ja E. Pehkonen: *Matematiikkaa soveltajille*, Kirjayhtymä, Helsinki, 1992.

Erinomainen taulukko- ja käsikirja:

[AS] M. Abramowitz and I. Stegun, editors: *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*, Dover, 1965. Tärkeimpiä käsikirjoja erikoisfunktioiden teoriassa ja sisältää laskettuja esimerkkejä myös numeriikan eri aloilta. Eniten myytyjä (ja siteerattuja) matematiikan kirjoja. Online versio:`http://www.math.sfu.ca/ cbm/aands/`

Eräitä perusteoksia

[A] H. Apiola: *Symbolista ja numeerista matematiikkaa Maple-ohjelmalla*, Otatieto, 1998.

[LiP] G. Lindfield and J. Penny: *Numerical methods using MATLAB, Second Ed.*, Prentice-Hall, Inc., New Jersey, 2000.

[M] J. Mathews: *Numerical Methods for Mathematics, Science, and Engineering, 2nd ed*, Prentice Hall, 1992; MATLAB-liite, saatavissa ftp:llä; Mathematica-liite. (Kolmas painos v. 1999 Mathews ja Howell.)

[Mol] C.B. Moler: *Numerical Computing with MATLAB*, SIAM 2004.
MATLABin kehittäjän korkealle arvostettu teos.

Hyödyllistä lisämateriaalia:

[D] H.F. Davis: *Fourier Series and Orthogonal Functions*, Dover, 1989.

[KMN] D. Kahaner, C. Moler, and S. Nash: *Numerical Methods and Software*, Prentice Hall, 1989. Mukana FORTRAN-disketti. Tunnettujen tekijöiden teos, jossa numeriikan perusalgoritmit esitellään luotettavasti. Ohjelmat saatavana osoitteesta `ftp.mathworks.com/pub/books` ; komenna `get kahaner.tar` .

[LI] E. Lindelöf: *Differentiaali- ja integraalilasku III*, Helsinki, 1940.

[Mpl/L] K. M. Heal et al.: *Maple V / Learning Guide*, Waterloo Maple Inc., Springer-Verlag New York etc., 1998 (Release 5), 1996 (Release 4).

[Mpl/P] M. B. Monagan et al.: *Maple V / Programming Guide*, Waterloo Maple Inc., Springer-Verlag New York etc., 1998 (Release 5), 1996 (Release 4).

[Sig] K. Sigmon: *Matlab Primer*, CRC Press, 1994.

[NR] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling: *Numerical Recipes in C, 2nd ed.*, Cambridge Univ. Press, 1992. Fyysikkojen kirjoittama valtavan suosion saavuttanut sovelletun numeriikan yleisesitys, jota on myös paljon kritisoitu. Huom: Myös Pascal-kieliset ja FORTRAN-versiot.

[NR1] –"–: *Numerical Recipes C-Example Book, 2nd ed.*

[NR2] –"–: *Numerical Recipes C-Diskette Version 2.0.*

[S1] G. Strang: *Introduction to Applied Mathematics*, Wadsworth, 1985.

[S2] G. Strang: *Introduction to Linear Algebra*, Wadsworth, 1993.

Käyttöoppaita, tieteellisen laskennan ja numeriikan kirjallisuutta:

[BaN] J. J. Barton and L. R, Nackman: *Scientific and Engineering C++*, Addison Wesley, 1993.

[BuF] C. Buzzi-Ferraris: *Scientific C++*, Addison–Wesley, 1993.

[C] D. M. Capper: *C++ for Scientists, Engineers, and Mathematicians*, Springer-Verlag, 1994.

[CV] T. F. Coleman and C. van Loan: *Handbook for Matrix Computations*, SIAM, Philadelphia, 1988. Sisältää johdatuksen numeeriseen lineaarialgebraan. Viitesanat Fortran 77, BLAS, LINPACK, MATLAB.

[Flo] B. H. Flowers: *An Introduction to Numerical Methods in C++*, Clarendon Press, 1995.

[G] R. Glassey: *Numerical Computation Using C*, Academic Press, 1993. Erinomainen johdatus numeerisen matematiikan tarvitsemiin C-kielen piirteisiin, ehkä alkeellisempi kuin [NR]. Ohjelmat ovat saatavissa anonymous ftp:llä internet-osoitteesta 129.79.147.6, jossa komennetaan `cd pub/glassey` , `get csrcrtg.txt` .

[GV] G. H. Golub and C. F. van Loan: *Matrix computations*, The John Hopkins University Press, Second ed., 1989. Numeerisen lineaarialgebran perusteoksia.

[HP] T. Hopkins and C. Phillips: *Numerical methods in practice: Using the NAG library*, Addison–Wesley, 1988. Sisältää johdatuksen FORTRAN-kielisen tunnetun ja laajan NAG-ohjelmakirjaston käyttöön.

[Ki] S. Kivelä: *Matlab-opas*, Otatieto, 1991 (myydään laskentakeskuksessa).

[KL] J. Korpela ja T. Larmela: *C-ohjelmointikieli, 3. painos*, Otadata, Espoo, 1988.

[La] P. Laukkanen: *Johdatusta Mathematica-ohjelmiston käyttöön*, Jyväskylän yliopisto, Matematiikan laitos, 1992.

[Ma] M. Marcus: *Matrices and Matlab*, Prentice Hall, 1993.

[Mae] R. Maeder: *Programming in Mathematica, 2nd ed.*, Addison–Wesley, 1991.

[MAPLE] `http://daisy.uwaterloo.ca/`

[MATu] `http://www.glue.umd.edu/~nsw/ench250/matlab.htm`

[MNV] M. Mäkelä, O. Nevanlinna ja J. Virkkunen: *Numeerinen matematiikka*, Gaudeamus, 1982. Monipuolinen numeriikan perusteos, sisältää myös hyvän kirjallisuusluettelon.

[Mo] C. Moler: *MATLAB User's Guide.*

[Mos] S. L. Moshier: *Methods and programs for mathematical functions*, Ellis Horwood Ltd, 1989.

[Mpl/H] D. Redfern: *The Maple Handbook / Maple V Release 4*, Springer-Verlag New York, Inc., 1997.

[MplV] `http://www.maplesoft.com/products.html`

[N] J. C. NASH: *Compact Numerical Methods for Computers*, Second ed., Adam Hilger, Bristol and New York, 1990. Programs in Turbo Pascal available on a diskette.

[Na] S. NAKAMURA: *Applied Numerical Methods in C*, Prentice Hall, 1993.

[PG] M. C. POTTER AND J. GOLDBERG: *Mathematical Methods*, Prentice Hall, 1987.

[R] J. R. RICE: *Numerical Methods, software, and analysis*, McGraw–Hill, 1983. Sisältää perusteellisen yleiskatsauksen numeriikkaan ja ohjelmakirjastoihin IMSL, ACM, PROTRAN. Suositeltava kokeellisen numeriikan yleisteos.

[Ru1] H. RUSKEEPÄÄ: *Mathematica-opas, versio 4*, Otatieto 593, 2000.

[Ru2] H. RUSKEEPÄÄ: *Mathematica-Navigator*, Academic Press, 1999.

[SB] J. STOER AND R. BULIRSCH: *Introduction to Numerical Analysis, 2nd ed.*, Springer-Verlag, 1991. Numeerisen analyysin kansainvälisesti tunnettu ensyklopedinen esittely.

[TC] BORLAND: *TURBO C (Versio 2.0)*.

[W] S. WOLFRAM: *Mathematica, Version .0*, Addison–Wesley, 1997.

Sovellusalojen kirjallisuutta:

[Be] C. P. BERNARDIN: *C/Math Toolchest for engineering and scientific applications*, Prentice Hall, 1993 (2 disks). Mukana lähdekoodina mm. kätevä piirto-ohjelma.

[K] J. KAUPPINEN: *Fourier-muunnokset ja niiden sovelluksia*, Turun yliopisto, Sovelletun fysiikan laitos, 1994.

[SK] R. J. STROEKER AND J. F. KAASHOEK: *Discovering Mathematics with Maple / An interactive exploration for mathematicians, engineers and econometricians*, Birkhäuser Basel, 1999.

Hakemisto