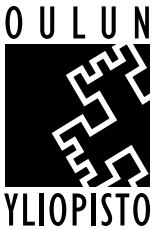


Säätötekniikan laboratorio
Infotech Oulu
ja
Prosessitekniikan osasto

MATLAB-ohjelman käyttö eräissä prosessiteknisissä laskuissa

Juha Jaako

Raportti B No 12, Syyskuu 1999

Sisällysluettelo

ISBN 951-42-5354-X SBN 951-42-7524-1 (PDF)
ISSN 1238-9404

Johdanto

Tämän kirjasen tarkoituksena on esittää, miten voidaan käyttää MATLAB[1] ohjelmointikieltä

- ohjelmoinnissa - mukana oleva koodi on testattua ja toimivaa sekä lisäksi koodi on täydellisenä esitetty;
- laskemaan prosessitekniikan laskuja ja erityisesti yksinkertaisia prosessiteknisiä optimointitehtäviä,
- tekemään graafisia esityksiä ja siirtämään ne tekstinkäsittelyohjelmaan.

Eräs kirjan painotuksista on se, miten ohjelmat tekijän mielestä pitäisi ulkoasultaan kirjoittaa.

Tarkoituksellisesti tässä kirjassa ei ole säätöteknisiä laskuja, koska on olemassa hyviä kirjoja tätä varten, esim. Bishop (1997) ja Ogata (1997).

Tekstin alkuosassa on kunkin luvun lopuksi esitetty luettelo käytetyistä MATLAB-funktioista. Kaikki tässä kirjassa oleva koodi on kenen tahansa käytettävissä. Jos kuitenkin käytät tässä kirjassa olevaa koodia, on lähde (siis tämä kirja) mainittava.

Käytetyt työkalut:

- Osa koodista on testattu Matlab-versiolla 5.3.0.10183 (R11), mutta pääosa koodista on tehty toimimaan versiossa 5.1.0.421. Kaikkien ohjelmien tulisi toimia ilman Toolboxeja mutta tätä ei ole varmistettu.
- Ohjelmaeditorina on ollut MicroEMACS 3.12.
- Tämä kirja on kirjoitettu Microsoft Word 97 SR-1 tekstinkäsittelyohjelmalla ja paperiversio on tulostettu käyttämällä Hewlett-Packard LaserJet 5M kirjoitinta ja Abode Acrobat versio[2] Adoben Distiller Assistant v3.01:llä.
- Käytetty kirjasin on Tahoma 10 pt, ohjelmakoodin esittämisessä on käytetty kirjasinta Lucida Console 8 pt ja matemaattiset kaavat on kirjoitettu Microsoft Equation Editor ver 3.0:lla. Kirjan valmistuksessa on käytetty seuraavia kuvaformaatteja: TIFF (-tiff), JPEG (-djpeg) ja Encapsulated Level 2 PostScript (-deps2).
- Käytetty kone oli seuraavanlainen: käyttöjärjestelmä Windows NT 4.0, prosessori Pentium 160 MHz, keskusmuisti 64 MB, levytilaa 3 GB, näyttö Panasonic PanaSync/Pro 5G, näyttösovitin Matrox MGA Millennium 2 MB ja koneessa on verkkoyhteys.

Lopuksi kiitos Esko Juusolle, koska hän minulle ensimmäisenä esitteli Matlab-ohjelman ja sen mahdollisuudet.

Oulussa 03/09/1999 Juha Jaako

---

1 ® The Mathworks, Inc. 1984-1999.

2 Adobe Acrobat versio on saatavissa osoitteesta http://ntsat.oulu.fi/jaako/matlpros.pdf

Yleistä MATLAB-ohjelmointikielestä

Matlab on interaktiivinen ohjelma, jota voidaan käyttää tekniseen numeeriseen laskentaan. Nimi Matlab tulee sanoista MATrix LABoratory. Matlabin perusdatatyyppi on matriisi, mikä antaa mahdollisuuden tehdä erittäin tehokasta ja - ikävä kyllä - myös erittäin vaikeaselkoista koodia. Matlab on integroitu systeemi, jossa on mm. grafiikka, makrokieli, IEEE-aritmetiikka, nopea tulkki[3] ja monia valmiiksi koodattuja komentoja.

Matlabin kehitys alkoi 1980-luvun alussa. Jo alusta alkaen siitä tuli tiettyjen teknisten ja luonnontieteellisten alueiden standardityökalu. Alussa käyttö keskittyi korkeakouluihin, mutta luonnollisesti korkeakouluopiskelijat opittuaan mielestään tehokkaan työkalun käytön alkoivat käyttää Matlabia myös korkeakoulusta päästyään. Korkeakouluissa Matlabia käytetään mm. matematiikan opetuksessa (sovellettu lineaarialgebra), säätötekniikan opetuksessa, simuloinnissa, tilastomatematiikassa ja digitaalisessa signaalinkäsittelyssä. Osasyynä Matlabin kehitykseen on ollut se, että ohjelma on saatavissa käytännössä kaikkiin tietokonejärjestelmiin.

Matlab ei olisi nykyisen kaltainen ja niin suosittu ilman ns. Toolboxeja. Toolbox on kokoelma m-tiedostoja (Matlab-kielellä koodattuja ohjelmia), jotka on erityisesti tehty jonkin tutkimusalueen ongelmien ratkaisemiseen. Tällä hetkellä ovat tarjolla seuraavat Toolboxit: µ-Analysis and Synthesis, Communications, Control System, Database, Financial Toolbox, Frequency Domain System Identification, Fuzzy Logic, Higher-Order Spectral Analysis, Image Processing, LMI Control, Mapping, Model Predictive Control, NAG, Neural Network, Optimization, Partial Differential Equation, QFT Control Design, Robust Control, Signal Processing, Spline, Statistics, Symbolic/Extended Symbolic Math, System Identification, Wavelet

(http://www-europe.mathworks.com/products/toolboxes.shtml).

Matlabiin liittyy läheisesti ohjelma nimeltään Simulink , jota käytetään graafisessa mallintamisessa ja simuloinnissa

(http://www-europe.mathworks.com/products/simulink/index.shtml).

Simulinkistä on tullut myös tiettyjen alojen standardityökalu.

Seuraavassa eräitä Matlab-linkkejä (tilanne 25.05.1999)

- http://www.mathworks.com/ ' Matlab kotisivu
- http://www-europe.mathworks.com/ ' Matlab kotisivu eurooppalaisille
- http://www-europe.mathworks.com/education/ ' Matlab opetuksessa
- http://www-europe.mathworks.com/education/basics.shtml ' Opetus & Matlab
- http://www-europe.mathworks.com/books/ ' Matlab kirjoja

[3] Vaikka Matlab onkin tulkkaava järjestelmä, on se kuitenkin suht' nopea. Lisäksi Matlabiin on tarjolla kääntäjä, joka kääntää Matlab-koodin C- tai C++-koodiksi.

1. Pienimmän neliösumman menetelmä - lineaarinen tapaus

Esimerkki otettu kirjasta Edgar & Himmelblau (1988, 55-56). Eräästä koesarjasta on saatu seuraavat tulokset

Taulukko 1.1: Koesarja

x	20	20	30	40	40	50	50	50	60	70
Y	73	78	85	90	91	87	86	91	75	65

Nyt x (reaktantin mooliosuus) on riippumaton muuttuja ja Y (saanto) on riippuva muuttuja. Halutaan sovittaa neliöllinen malli taulukon 1 tietojen avulla ja lisäksi halutaan määrittää se x:n arvo, jolla saavutetaan maksimisaanto. Sovitettava malli on muotoa

$$Y = b_0 + b_1 x + b_2 x^2 \tag{1.1}$$

Nyt mallissa (1.1) on 3 parametria ja taulukon 1 avulla voidaan muodostaa 10 (=p) yhtälöä; asia voidaan esittää seuraavasti matriisimuodossa

$$\mathrm{X\,b} = \mathrm{Y} \tag{1.2}$$

jossa $\mathrm{b} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix}$, $\mathrm{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_p \end{bmatrix}$ ja $\mathrm{X} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_p & x_p^2 \end{bmatrix}$.

Koska nyt on kyseessä ylimääritelty tapaus, ei (1.2):lla välttämättä ole lainkaan ratkaisua (Kivelä 1980, 133). Täten on perusteltua hakea sellaista ratkaisua, joka mahdollisimman tarkoin toteuttaa yhtälön (1.2). Tavallisimmin ongelma ratkaistaan pienimmän neliösumman menetelmällä. Koska nyt kyseessä on parametrien suhteen lineaarinen tapaus, saadaan parametrien b_0, b_1 ja b_2 arvot seuraavasti (Kivelä 1980, 133- 134; Edgar & Himmelblau 1988, 54) käytettäessä matriisimuotoa.

$$\mathrm{b} = \left(\mathrm{X}^T\mathrm{X}\right)^{-1}\mathrm{X}^T\mathrm{Y} \tag{1.3}$$

Nyt saadaan taulukon 1.1 arvojen perusteella

$$\mathrm{Y} = \begin{bmatrix} 73 & 78 & 85 & 90 & 91 & 87 & 86 & 91 & 75 & 65 \end{bmatrix}^T \tag{1.4}$$

$$\mathrm{X} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 20 & 20 & 30 & 40 & 40 & 50 & 50 & 50 & 60 & 70 \\ 400 & 400 & 900 & 1600 & 1600 & 2500 & 2500 & 2500 & 3600 & 4900 \end{bmatrix}^T \tag{1.5}$$

Kun sijoitetaan (1.4) ja (1.5) yhtälöön (1.3) saadaan monimutkainen matriisiyhtälö, jonka ratkaiseminen ei ole kovin helppoa, mutta käyttämällä MATLAB-ohjelman matriisinkäsittelykomentoja saadaan ratkaisu helposti; lisäksi MATLABissa voidaan ratkaisukaava (1.3) kirjoittaa suoraan matriisimuodossa. Ratkaisemalla MATLABilla saadaan

$$b = \begin{bmatrix} 35.6574 \\ 2.6314 \\ -0.0319 \end{bmatrix} \tag{1.5}$$

ja edelleen malliyhtälöksi

$$Y = 35.66 + 2.63\,x - 0.032\,x^2 \tag{1.6}$$

Seuraavassa on esitetty ongelman ratkaiseva MATLAB-koodi (EHals1.m).

```
% EHals1.m - Neliöllinen malli.
% 27.01.1999 (c) Juha Jaako
% Edgar & Himmelblau s. 55-56
% Tämä versio laskee vain malliparametrit!

clear all ;

% Data - pystyvektoreita!
x = [ 20 20 30 40 40 50 50 50 60 70 ]' ;
Y = [ 73 78 85 90 91 87 86 91 75 65 ]' ;
N = size (x, 1) ;                    % Vektorin pituus.

% Muodostetaan X-matriisi.
mX = zeros (N, 3) ;          % Alustetaan matriisi nollilla.
for i = 1 : N ,              % Täytetään matriisi rivi kerrallaan.
  mX(i,:) = [ 1 , x(i) , x(i)^2 ] ;
end

% Ratkaistaan malliparametrit.
b = (inv ((mX')*mX))*(mX')*Y       % Ratkaisukaava matriisimuodossa
```

Ohjelma tulostaa vastaukseksi

```
» EHals1
b =
   35.6574
    2.6314
   -0.0319
```

Käytetyssä lähteessä parametrien arvoiksi oli saatu (35.66, 2.63, -0.032). Edellä oleva koodi laskee pelkät parametrit; havainnollisempaa olisi laskea maksimisaanto, mallin hyvyys ja selitysaste sekä piirtää kuva, jossa ovat sekä koepisteet että malliyhtälö. Saatava kuva olisi myös pystyttävä siirtämään tekstinkäsittelyohjelmaan raportointia varten. Seuraavassa on edellinen koodi (EHals1.m) laajennettu laskemaan ja piirtämään kaikki edellä olevat (EHals.m).

```
% EHals.m - Neliöllinen malli.
% 27.01.1999 (c) Juha Jaako
% Edgar & Himmelblau s. 55-56

clear all ; hold on ;
```

```
% Data
x = [ 20 20 30 40 40 50 50 50 60 70 ]' ;
Y = [ 73 78 85 90 91 87 86 91 75 65 ]' ;
N = size (x, 1) ;                          % Vektorin pituus.
plot (x, Y, 'ko') ;                        % Piirretään koepisteet.

% Muodostetaan X-matriisi.
mX = zeros (N, 3) ;
for i = 1 : N,
  mX(i,:) = [ 1 , x(i) , x(i)^2 ] ;
end

% Ratkaistaan malliparametrit.
b = (inv ((mX')*mX))*(mX')*Y ;

% Piirretään mallin kuvaaja.
xx = min(x) : 0.1 : max(x) ;               % 0.1:n välein välille 20-70.
M = size (xx,2) ;
for i = 1 : M,
  yy(i) = b(1)+b(2)*xx(i)+b(3)*xx(i)^2 ;
end
plot (xx, yy, 'k-') ;                      % Piirretään malli kuvaan.

% Maksimi voidaan etsiä kahdella tavalla
% JOKO etsimällä maksimi edellä lasketuista vektoreista.
[I,J] = max (yy) ;                         % I : suurin arvo.
                                           % J : arvon indeksi.
disp (' ') ;
disp ( 'Maksimi vektoreista hakemalla:') ,
disp ( ['Suurin arvo (Y) : ' num2str(I) ]) ;
disp ( ['Pisteessä (x) : ' num2str(xx(J)) ]) ;
plot (xx(J), I, 'k+') ;

% TAI kayttamalla MATLABin fmin-funktiota (kts. liite 3!)
% Etsitaan maksimi (pannaan miinusmerkki eteen).
  % Ensin muodostetaan kohdefunktio.
b0s = num2str(b(1)) ;
b1s = num2str(b(2)) ;
b2s = num2str(b(3)) ;
funktio = [ '-((' b0s ')+(' b1s ')*x+(' b2s ')*x^2)' ] ;

  % Sitten kutsutaan MATLABin fmin-funktiota.
xmax = fmin (funktio, min(x), max(x) ) ;
ymax = b(1)+b(2)*xmax+b(3)*xmax^2 ;

disp (' ') ;
disp ( 'Maksimi fmin-funktiolla:') ,
disp ( ['Suurin arvo (Y) : ' num2str(ymax) ]) ;
disp ( ['Pisteessä (x) : ' num2str(xmax) ]) ;
plot (xmax, ymax, 'k*') ;

% Kuvan otsikot, akselit jne.. (käytetään kursiivia).
xlabel ('\itx') ; ylabel ('\itY') ;
  % _ : alaindeksi, ^ : yläindeksi
title ('Neliöllinen malli {\itY = {b_0}+{b_1 x}+{b_2}{x^2}}') ;

% Esitetään mallparametritkin kuvassa.
text (30, 72.5, ['\it{b_0} = ' b0s ] ) ;
text (30, 70.0, ['\it{b_1} = ' b1s ] ) ;
text (30, 67.5, ['\it{b_2} = ' b2s ] ) ;

% Lasketaan mallin selitysaste.
R2 = seliaste (Y, mX*b, N) ;
R2s = num2str (R2) ;
text (30, 64.0, ['\it{R^2} = ' R2s ] ) ;

hold off ;
```

```
% Tulostetaan kuva tiedostoon Encapsulated Postscript -muotoon[4].
print -tiff -deps2 c:\temp\EHals ;
```

Selitysasteen laskenta tehdään erillisen funktion avulla (MATLAB-ohjelmointikielessä funktiot on koodattava erilliseksi tiedostoksi) - huomaa funktion syntaksi.

```
% seliaste.m - Laskee selitysasteen
% (c) Juha Jaako 20.11.1998

% DO     - havaitut arvot
% DOkalc - mallin antamat arvot

function f = seliaste (DO, DOkalc, N)

% Laskettu selitysaste.
SST = 0 ;          % Kokonaisneliösumma
SSR = 0 ;          % Jäännösneliösumma
SSD = 0 ;          % Selitetty neliösumma

DOmean = mean (DO) ;
for i = 1 : N,
  SST = SST + (DO(i)-DOmean)^2 ;
  SSR = SSR + (DO(i)-DOkalc(i))^2 ;
end
SSD = SST - SSR ;
f = 1-(SSR/SST) ;
```

Koodi tulostaa seuraavaa

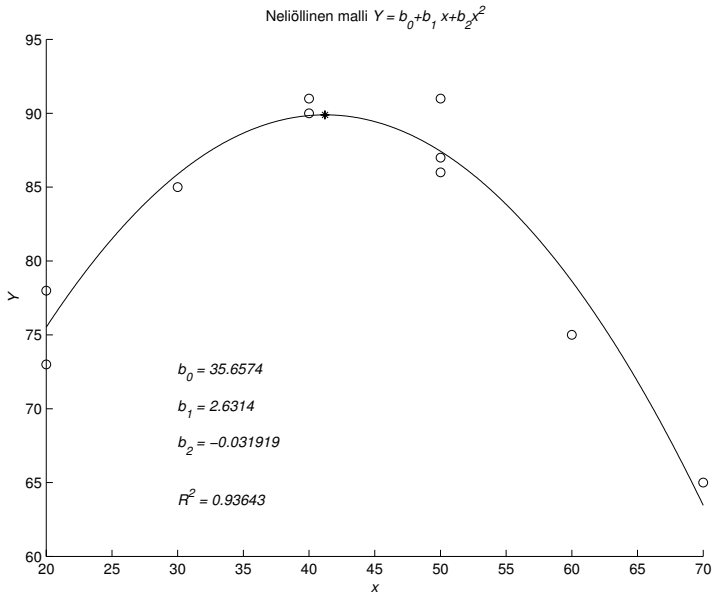
```
» ehals

Maksimi vektorista hakemalla:
Suurin arvo (Y) : 89.8933
Pisteessä (x) : 41.2

Maksimi fmin-funktiolla:
Suurin arvo (Y) : 89.8933
Pisteessä (x) : 41.22
```

sekä lisäksi saadaan kuva, johon tulostuvat myös tekstit. Kuvatekstit on osaksi kirjoitettu kursiivilla käyttäen (\it)-tarkenninta. Tekstin muotoiluun on tarjolla paljon erilaisia vaihtoehtoja[5].

---

[4] TIFF = *Tagged Image File Format* on eräs kuvantallennusformaatti.
DEPS2 = *Level 2 black and white Encapsulated PostScript* (EPS) on myös eräs kuvantallennusformaatti.

[5] kts. viite *Using MATLAB Graphics*.



Kuva 1.1. Mallipisteet, malliyhtälö ja malliparametrit.

Koodissa oleva komento

```
print -tiff -deps2 c:\temp\EHals ;
```

tulostaa kuvan tiedostoon PostScript-muotoon (-deps2) hakemistoon c:\temp ja lisäksi tiedostoon liitetään kuvan esikatselumahdollisuus (-tiff). Ilman (-tiff)-tarkenninta ei kuvaa voi nähdä tekstinkäsittelyohjelmassa; vain pelkkä laatikko näkyy[6]. Esimerkiksi Word-tekstinkäsittelyohjelmaan kuva liitetään valikkokomennoin Insert-Picture-From File...-Ehals. Lopuksi teksti on tulostettava PostScript-kirjoittimella.

Jos käytössä ei ole PostScript-kirjoitinta ja jos kuvan laadulle ei aseteta suuria vaatimuksia, niin voidaan kuva tulostaa JPEG[7]-muotoon seuraavalla komennolla

```
print -djpeg c:\temp\EHals ;
```

kuva saadaan Word-tekstinkäsittelyohjelmaan samoin komennoin kuin edellä.

---

6 kts. viite *Using MATLAB Graphics*, Chapter 7 Printing MATLAB Graphics.

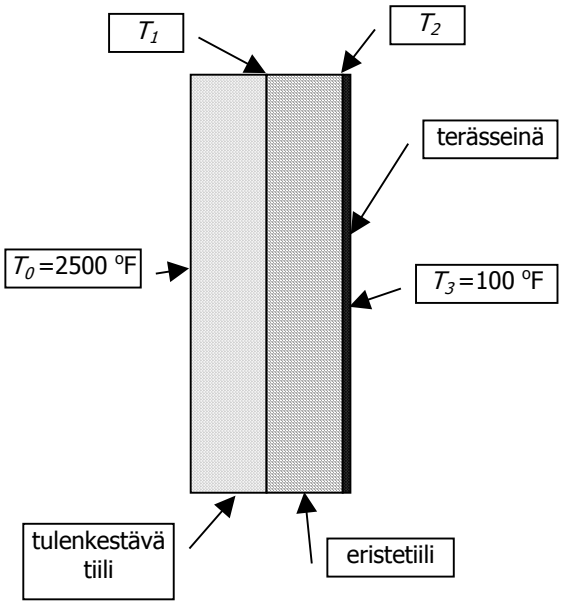
7 JPEG =*Joint Photographers Expert Group* on eräs kuvantallenusformaatti, joka on häviöllinen ja kompressoitu.

Taulukko 1.2: Käytetyt MATLAB-funktiot

funktio	kuvaus
'	laskee matriisin (tahi vektorin) transpoosin; erittäin kätevä komento
%	kaikki %-merkin jäljessä rivillä oleva teksti on kommenttia
;	jos halutaan, ettei komentorivi tulosta laskennan tulosta, niin rivin loppuun pannaan puolipiste; puolipistettä käytetään myös erottimena matriiseissa
clear all	poistaa kaikki muuttujat, funktiot ja MEX-linkit; varmistaa sen etteivät edelliset laskennat pääse sotkemaan meneillään olevaa laskentaa; varotoimenpide
disp	tulostaa taulukon ilman taulukon nimeä; jos argumentti on tekstiä, niin teksti tulostuu
fmin	minimoi yhden muuttujan funktion
for ... end	toistaa komentoja ennalta määrätyn määrän
hold off	lopettaa komennon hold on vaikutuksen
hold on	pitää jo piirretyn kuvan aktiivisena, joten kuvaan voidaan piirtää uusia elementtejä
inv	laskee käänteismatriisin
max	hakee suurimman arvon
mean	laskee keskiarvon
min	hakee pienimmän arvon
num2str	muuttaa luvun merkkijonoksi, tarvitaan tulostettaessa lukuja tekstin sekaan
plot	tulostaa x-y-kuvan, tarjolla runsaasti erilaisia viiva- ja pistevaihtoehtoja
print	tulostaa kuvan joko kirjoittimelle tai haluttuun tiedostoon; tarjolla on runsaasti erilaisia tiedostoformaatteja, mm. PostScript ja JPEG; käytetään, kun halutaan siirtää kuvia tekstinkäsittelyohjelmaan
size	laskee matriisin dimensiot
text	lisää halutun tekstin kuvaan haluttuun kohtaan
title	lisää kuvaan otsikkotekstin
xlabel	lisää kuvaan selittävän tekstin x-akselille
ylabel	lisää kuvaan selittävän tekstin y-akselille
zeros	muodostaa matriisin jonka kaikki alkiot ovat nollia; käytetään kun alustetaan taulukoita

2. Monikerrosseinämän mitoittaminen

Esimerkki otettu kirjasta Bird *et al.* (1960, 305). Seinämä koostuu kolmesta kerroksesta kuvan 2.1 mukaan: ensimmäinen kerros on tulenkestävää tiiltä, toinen kerros on eristetiiltä ja uloin kerros teräslevyä, jonka paksuus on 1/4 engl. tuumaa.



Kuva 2.1. Komposiittirakenteinen tuliseinä.

Lasketaan seinämän eri osille sellainen paksuus, että seinämän kokonaispaksuus on mahdollisimman pieni. Lämmönhukka seinämän läpi saa olla korkeintaan[8] 5000 $\frac{Btu}{ft^2\ h}$; oletetaan samalla, että seinämän osat ovat termisesti hyvässä kontaktissa.

[8] Säilytetään nyt käytettävät yksiköt arkaaisina, jotta voidaan havainnollistaa MATLABin funktioiden käyttöä.

Laskentaa varten on tarjolla seuraavat tiedot

Taulukko 2.1: Seinämän tiedot

Materiaali	Korkein käyttölämpötila	Lämmönjohtavuus $\left[\frac{Btu}{h\ ft\ ^oF}\right]$	
		100 °F:ssa	2000 °F:ssa
Tulenkestävä tiili	2600 °F	1.8	3.6
Eristetiili	2000 °F	0.9	1.8
Teräs	-	26.1	-

Taulukosta huomaamme, että tulenkestävän tiilikerroksen on oltava niin paksu, että $T_1 \leq 2000$ °F. Seinän mitoitus tapahtuu kaavojen (9.6-6)-(9.6-8) (Bird *et al.* 1960, 285) mukaan (tässä niitä on merkitty kaavoina (2.1)-(2.3))

$$-k^{01}\frac{dT^{01}}{dx} = q_0 \tag{2.1}$$

$$-k^{12}\frac{dT^{12}}{dx} = q_0 \tag{2.2}$$

$$-k^{23}\frac{dT^{23}}{dx} = q_0 \tag{2.3}$$

Integroimalla yhtälöt (2.1)-(2.3) saaadaan, kun k^{01}, k^{12} ja k^{23} ovat vakioita

$$T_0 - T_1 = -q_0\left(\frac{x_0 - x_1}{k^{01}}\right) \tag{2.4}$$

$$T_1 - T_2 = -q_0\left(\frac{x_1 - x_2}{k^{12}}\right) \tag{2.5}$$

$$T_2 - T_3 = -q_0\left(\frac{x_2 - x_3}{k^{23}}\right) \tag{2.6}$$

Tässä tehtävässä eivät k^{01} ja k^{12} olekaan vakioita, mutta k^{23} on vakio (kts. taulukko 2.1). Oletetaan nyt lineaarinen riippuvuus

$$k^{01}(T) = a_1 T + b_1 \tag{2.7}$$

jossa a_1 ja b_1 ovat sovitettavia parametreja. Sijoittamalla (2.7) kaavaan (2.1) saadaan

$$-\left(a_1 T^{01} + b_1\right)\frac{dT^{01}}{dx} = q_0 \tag{2.8}$$

Integroimalla (2.9)

$$-\int_{T_0}^{T_1}\left(a_1 T^{01} + b_1\right)dT^{01} = \int_{x_0}^{x_1} q_0\ dx \tag{2.9}$$

saadaan tulenkestävän tiiliseinämän leveydelle kaava (2.10), eristetiiliseinämän leveydelle kaava (2.11) ja terässeinälle kaava (2.12)

$$x_1 - x_0 = \frac{\frac{a_1}{2}\left(T_1^2 - T_0^2\right) + b_1\left(T_1 - T_0\right)}{-q_0} \tag{2.10}$$

$$x_2 - x_1 = \frac{\frac{a_2}{2}\left(T_2^2 - T_1^2\right) + b_2\left(T_2 - T_1\right)}{-q_0} \tag{2.11}$$

$$T_2 = T_3 - q_0\left(\frac{x_2 - x_3}{k^{23}}\right) \tag{2.12}$$

Nyt on vielä määritettävä parametreille a_i ja b_i arvot. Taulukon 2.1 avulla saadaan kaksi yhtälöparia

Yhtälömuodossa	Matriisimuodossa	Ratkaisu
$\begin{cases} 100\,a_1 + b_1 = 1.8 \\ 2000\,a_1 + b_1 = 3.6 \end{cases}$	$\begin{bmatrix} 100 & 1 \\ 2000 & 1 \end{bmatrix}\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} 1.8 \\ 3.6 \end{bmatrix}$	$\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} 100 & 1 \\ 2000 & 1 \end{bmatrix}^{-1}\begin{bmatrix} 1.8 \\ 3.6 \end{bmatrix} = \begin{bmatrix} 0.0009 \\ 1.7053 \end{bmatrix}$
$\begin{cases} 100\,a_2 + b_2 = 0.9 \\ 2000\,a_2 + b_2 = 1.8 \end{cases}$	$\begin{bmatrix} 100 & 1 \\ 2000 & 1 \end{bmatrix}\begin{bmatrix} a_2 \\ b_2 \end{bmatrix} = \begin{bmatrix} 0.9 \\ 1.8 \end{bmatrix}$	$\begin{bmatrix} a_2 \\ b_2 \end{bmatrix} = \begin{bmatrix} 100 & 1 \\ 2000 & 1 \end{bmatrix}^{-1}\begin{bmatrix} 0.9 \\ 1.8 \end{bmatrix} = \begin{bmatrix} 0.0005 \\ 0.8526 \end{bmatrix}$

Lineaarinen yhtälöryhmä voidaan ratkaista MATLAB-ohjelmalla seuraavasti. Syötetään kerroinmatriisit

```
» A = [ 100 1 ; 2000 1] ;
» b = [ 0.9 ; 1.8 ] ;
```

Ratkaisu saadaan joko komennolla

```
» inv(A)*b
ans =
    0.0005
    0.8526
```

tai (suositeltava tapa) jakamalla kenoviivalla

```
» A\b
ans =
    0.0005
    0.8526
```

Seuraavassa on ongelman ratkaisu koodattu MATLAB-ohjelmaksi (threwall.m); huomaa, että ohjelma laskee ongelman ratkaisun SI-yksikköinä. Muunnoksen yksikköjen välillä tapahtuvat erikseen kirjoitettujen funktioiden (F2K.m, Bfh2Wm.m ja BfhF2WmK.m) avulla.

Ohjelma tulostaa ratkaisuna

```
» threwall
T2 =
  313.1450   % Kelvin
x1mx0 =
    0.1169   % metreinä
    0.3837   % jalkoina
x2mx1 =
    0.1561   % metreinä
    0.5123   % jalkoina
```

jota voi verrata viitteen (Bird *et al.* 1960, 305) ratkaisuun (0.39 ft ja 0.51 ft). Ohjelmakoodi on seuraavanlainen

```
% threwall.m
% 12.01.1999 (c) Juha Jaako
% Lämmönsiirto kolmen seinämän läpi.
% Bird et al., s. 284-285 & 305

format short ;

% ____________________________Lasketaan terässeinä.

% Ensin muunnos SI-yksiköihin.
T3 = F2K(100) ;                 % oF -> K
q0 = Bfh2Wm(5000) ;             % Btu/(ft^2 h) -> W/m^2
x2mx3 = 0.0254*(-1/4) ;         % in -> m
k23 = BfhF2WmK(26.1) ;          % Btu/(ft h oF) -> W/(m K)

T2 = T3 - q0*(x2mx3/k23)        % Kaava (2.12)

% ____________________Lasketaan tulitiiliseinä.

% Lasketaan ensin datan avulla parametrit.
A = [ F2K(100)          1 ; ...
      F2K(2000)         1 ] ;
B = [ BfhF2WmK(1.8) ; BfhF2WmK(3.6) ] ;
data = A\B ;
a1 = data(1) ;
b1 = data(2) ;

T0 = F2K(2500) ;                % oF -> K
T1 = F2K(2000) ;                % oF -> K

temp1 = (a1/2)*(T1^2-T0^2) ;
temp2 = b1*(T1-T0) ;
x1mx0 = (temp1+temp2)/(-q0)     % Kaava (2.10)
disp(x1mx0/0.3048) ;            % m -> ft

% ___________________Lasketaan eristetiiliseinä.

% Lasketaan ensin datan avulla parametrit.
A = [ f2K(100)          1 ; ...
      f2K(2000)         1 ] ;
B = [ BfhF2WmK(0.9) ; BfhF2WmK(1.8) ] ;
data = A\B ;
a2 = data(1) ;
b2 = data(2) ;

temp1 = (a2/2)*(T2^2-T1^2) ;
temp2 = b2*(T2-T1) ;
x2mx1 = (temp1+temp2)/(-q0)     % Kaava (11)
disp(x2mx1/0.3048) ;            % m -> ft
```

Muunnosfunktiot SI-yksiköihin ovat seuraavanlaisia (huomaa, että kaikki funktiot on kirjoitettava *erillisiksi* tiedostoiksi)

```
% F2K.m
% 12.01.1999 (c) Juha Jaako
% Fahrenheit to Kelvin conversion.

function K = F2K(F)

K = (F-32)/(1.8) + 273.15 ;

% Funktion loppu


% BfH2Wm.m
% 12.01.1999 (c) Juha Jaako
% Btu/(ft^2 h) to W/m^2 conversion
% CRC 74th edition s. 1-26

function Wm = BfH2Wm(BfH)

Wm = 3.154591*BfH ;

% Funktion loppu


% BfHF2WmK.m
% 12.01.1999 (c) Juha Jaako
% Btu/(ft h oF) to W/(m K) conversion
% CRC 74th edition s. 1-26

function WmK = BfHF2WmK(BfHF)

WmK = 1.730735*BfHF ;
```

Taulukko 2.2: Käytetyt MATLAB-funktiot

funktio	kuvaus
\	yhtälöryhmön ratkaisu (» help slash)
format short	käytetään lyhyttä tulostusmuotoa

3. Kolmiulotteinen kuvaaja ja satunnaishaku

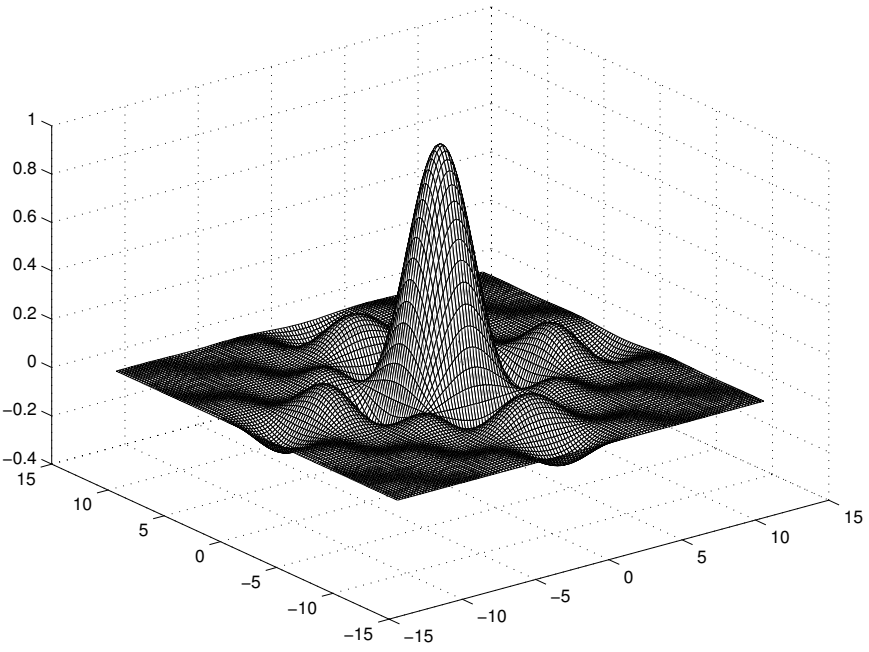
Esimerkki otettu Laineen kirjasta (1997, 186). Monesti graafisia esityksiä tehtäessä tulee tarve piirtää 3-ulotteisia kuvia. Seuraavassa piirretään funktion

$$z(x,y)=\frac{\sin x}{x}\frac{\sin y}{y} \qquad (3.1)$$

kuvaaja alueelle $-4\pi < x < 4\pi \quad ; \quad -4\pi < y < 4\pi$. Huomaa, että funktiolla on monta paikallista minimiä ja maksimia tällä alueella.

Kuva funktiosta 3-ulotteisena voidaan piirtää seuraavanlaisen koodinpätkän avulla.

```
% lainк187.m
% 25.11.1998 (c) Juha Jaako

% Funktion kuvaaja.
axis([-4*pi 4*pi -4*pi 4*pi -0.4 1.0]);
[x1, x2] = meshgrid(-4*pi : 0.2 : 4*pi, -4*pi : 0.2 : 4*pi);
A = (sin(x1)./x1).*(sin(x2)./x2);
mesh(x1, x2, A);
print -deps2 c:\temp\lainк187;
```



Kuva 3.1. Kolmiulotteisen kuvaajan piirtäminen.

Kuvaajan piirtämisessä ei tarvita kovin monta komentoa, mutta komennot ovat todella tehokkaita. Käydään seuraavassa ko. komennot läpi.

axis

axis-komennon avulla määritellään akseleiden skaalausta. Tässä tapauksessa x-akselin kattaa välin $-4\pi < x < 4\pi$, y-akseli välin $-4\pi < y < 4\pi$ ja z-akseli välin $-0.4 < z < 1$. Komento on

```
axis ( [-4*pi  4*pi  -4*pi  4*pi  -0.4  1.0] ) ;
```

meshgrid

meshgrid-komento muodostaa verkon laskentapisteille väleille $-4\pi < x < 4\pi$ ja $-4\pi < y < 4\pi$. Luku 0.2 ilmoittaa verkon jaotuksen. Mitä pienempää lukua käytetään sitä tarkempi kuvasta saadaan; samalla on kuitenkin huomattava, että mitä suurempi tarkkuus halutaan niin sitä hitaammin piirto tapahtuu ja sitä enemmän muistitilaa käytetään. HUOM! MATLABin yksinkertaisilla komennoilla on hyvin helppo muodostaa erittäin suuria matriiseja ja samalla voi täyttää koneen keskusmuistin. Komento on

```
[x1, x2] = meshgrid (-4*pi : 0.2 :  4*pi, -4*pi : 0.2 : 4*pi ) ;
```

/. ja .*

Seuraavaksi lasketaan funktion arvot meshgrid-komennon luomissa verkkopisteissä. Huomaa, että kertomerkkinä on .* ja jakomerkkinä ./ . Komento on

```
A = (sin(x1)./x1).*(sin(x2)./x2) ;
```

mesh

mesh-komennolla piirretään varsinainen kuva. Mesh-komento vaatii syöttötietoina meshgrid-komennon muodostamat matriisit sekä funktion arvot ko. pisteissä.

```
mesh (x1, x2, A) ;
```

* * *

Kuvasta 3 huomaamme, että funktiolla on alueella yksi globaali maksimi. Funktio (3.1) on muodoltaan hyvin hankala, joten käytetään maksimia haettaessakin eksoottista menetelmää.

Optimin haku voidaan tehdä satunnaishakuna (Laine 1997, 186) seuraavan koodin avulla.

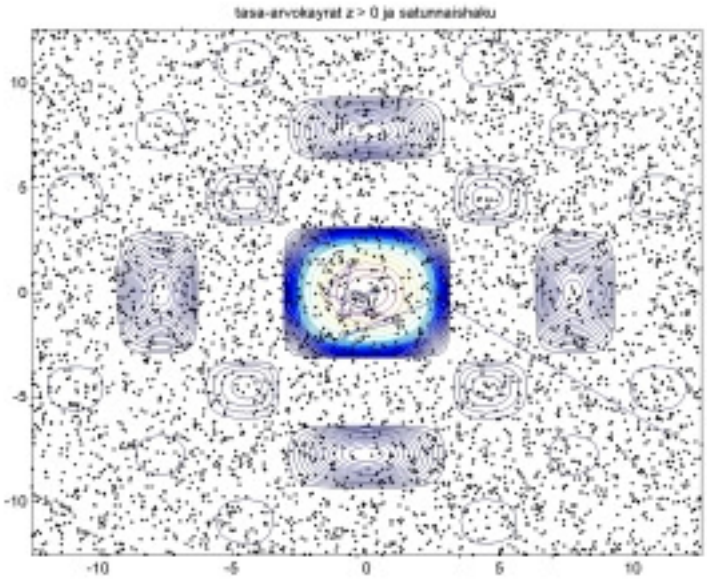
```
% laine187.m
% 29.01.1998 & 25.11.1998 & 26.11.1998
% (c) Juha Jaako
% Satunnaishaku.

% Funktion kuvaaja.
title ('tasa-arvokayrat z > 0 ja satunnaishaku') ;
axis ( [-4*pi  4*pi  -4*pi  4*pi ] ) ;
[x1, x2] = meshgrid (-4*pi : 0.1 :  4*pi, -4*pi : 0.1 : 4*pi ) ;
z1 = (sin(x1)./x1).*(sin(x2)./x2) ;
ta1 = 0.01 : 0.01 : 0.19 ;
ta2 = 0.2  : 0.02 : 0.28 ;
ta3 = 0.3  : 0.1  : 1.0 ;
ta = [ta1 ta2 ta3] ;
contour (x1, x2, z1, ta) ;
```

```
% Satunnaishaku.
hold on ;
axis ( [-4*pi  4*pi  -4*pi  4*pi] ) ;
m = REALMIN ;

% Lasketaan 5000 kierrosta.
for i = 1 : 5000 ,
  x = 4*pi*(2*rand-1) ;
  y = 4*pi*(2*rand-1) ;
  plot (x, y, 'k.') ;
  if (i == 1) ,
    oldx = x ;
    oldy = y ;
    plot (x, y, 'ko') ;
  end
  z = (sin(x)/x)*(sin(y)/y) ;
  if (z > m) ,
    m = z ;
    str = sprintf ('%d\t%8.3f\t%8.3f\t%8.3f', i, x, y, z) ;
    disp (str) ;
    plot (x, y, 'ko') ;
    line ([oldx x], [oldy y]) ;
    oldx = x ; oldy = y ;
  end
end

hold off ;

print -djpeg100 c:\temp\laine187 ;
```

Ohjelma tulostaa seuraavaa:

```
» laine187
2        2.685        -0.352        0.161
18      -0.855        -2.045        0.384
39       1.170        -1.385        0.558
65      -1.542        -0.042        0.648
90      -0.543         1.378        0.678
118     -0.986        -1.072        0.693
197      0.152        -0.885        0.871
463     -0.512        -0.100        0.955
528     -0.342        -0.099        0.979
2586     0.345         0.074        0.979
2985    -0.108        -0.224        0.990
```

Seuraavaan kuvaan on piirretty haku: haun tarkastamat pisteet on merkitty (.):lla, haun hyväksymät pisteet (o):lla (huomaa, että jokaisella ohjelman suorituskerralla tulee aina uudenlainen haku ja vastaava kuva). Tasa-arvokäyriä on piirretty vain arvoilla $z > 0$, jotta kaikki paikallisetkin maksimit voidaan nähdä. Kuten huomataan, satunnaislukugeneraattoria käyttämälläkin voidaan päätyä optimiin.



Kuva 3.2. Funktion $z(x,y)=\frac{\sin x}{x}\frac{\sin y}{y}$ tasa-arvokuvaajia ja maksimin haku satunnaishaulla; huomaa, että kuva JPEG-kuva, joten kuvan laatu on huonompi kuin edellä kuvassa 3.1.

Taulukko 3.1: Käytetyt MATLAB-funktiot

funktio	kuvaus
axis	akseleiden skaalaus
contour	piirretään tasa-arvokuvaaja
if ... end	totuusarvon testaus
line	piirretään viiva pisteiden välille
mesh	piirretään kolmiulotteinen kuva
meshgrid	laskentapisteiden verkon muodostus
pi	π (=3.141...)
rand	generoi tasaisesti jakautuneita satunnaislukuja
REALMIN	pienin positiivinen liukuluku MATLABissa
sin	sinifunktio
sprintf	kirjoitetaan luku formatoituna merkkijonoksi; kätevä tulostettaessa

4. Optimaalisen kierrätyssuhteen määrittäminen autokatalyyttiselle reaktiolle

Esimerkki otettu kirjasta Levenspiel (1972, 155-156). Tarkastellaan seuraavanlaista autokatalyyttista reaktiota. Aine A reagoi seuraavanlaisesti

$$A + R \rightarrow R + R \tag{4.1}$$

jossa $k = 1\dfrac{l}{mol\ \min}$. Tarkoituksena on käsitellä $F_{A0} = 1\dfrac{mol}{\min}$ syötettä, jossa on pelkkää A:ta. ($C_{A0} = 1\dfrac{mol}{l}$, $C_{R0} = 0$) siten, että konversio on 99%. Reaktio tapahtuu kierrätysreaktorissa. Tarkoituksena on etsiä kierrätyssuhde (R), joka minimoi tarvittavan reaktorin tilavuus; lisäksi määritetään reaktorin tilavuus. Lisäksi verrataan tätä laskettua reaktorin tilavuutta sekoitusreaktoriin (R = ∞) ja putkireaktoriin (R = 0).

Tilavuudelle on johdettu seuraavanlainen kaava (Levenspiel 1972, 155)

$$\frac{V}{F_{A0}} = (\mathrm{R}+1) \int_{X_{Ai}=\frac{\mathrm{R}X_{Af}}{\mathrm{R}+1}}^{X_{Af}} \frac{dX_A}{k\ C_{A0}^2\ X_A(1-X_A)} \tag{4.2}$$

Integroimalla ja yksinkertaistamalla saadaan

$$\frac{V}{F_{A0}} = \frac{(\mathrm{R}+1)}{k\ C_{A0}^2} \ln\left[\frac{1+\mathrm{R}(1-X_{Af})}{\mathrm{R}(1-X_{Af})}\right] \tag{4.3}$$

josta analyyttisesti johtamalla ja asettamalla $X_{Af} = 0.99$ saadaan yhtälö

$$\ln\frac{1+0.01\,\mathrm{R}}{0.01\,\mathrm{R}} = \frac{\mathrm{R}+1}{\mathrm{R}(1+0.01\,\mathrm{R})} \tag{4.4}$$

josta yrityksen ja erehdyksen menetelmällä saadaan R = 0.19. Nyt voidaan tilavuus laskea

$$V = 1.19 \ln\frac{1.0019}{0.0019} = 7.46\ l \tag{5.5}$$

Näin siis Levenspiel. Ongelmaa voidaan tarkastella myös optimointiongelmana. Muutetaan yhtälö (4.2) muotoon

$$\min\left[V(\mathrm{R}) = F_{A0}(\mathrm{R}+1) \int_{X_{Ai}=\frac{\mathrm{R}X_{Af}}{\mathrm{R}+1}}^{X_{Af}} \frac{dX_A}{k\ C_{A0}^2\ X_A(1-X_A)}\right] \tag{4.6}$$

eli saamme yhden muuttujan minimointitehtävän, jossa kohdefunktiona on (4.6). Ongelmaksi muodostuu kohdefunktiossa olevan integraalin laskenta. Yhtälössä (4.3) on esitetty analyyttinen ratkaisu, mutta aina analyyttinen ratkaisu ei ole mahdollinen.

Yleensä tietokoneella laskettaessa käytetään numeerista integrointia; tavallisimmin numeerinen integrointi tapahtuu Simpsonin säännön avulla (kts. Kreyszig 1993, 960-963).

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left(f_0 + 4f_1 + 2f_2 + 4f_3 + \cdots + 2f_{2m-2} + 4f_{2m-1} + f_{2m}\right) \qquad (4.7)$$

jossa $h = \frac{b-a}{2m}$, $f_j = f(x_j)$ ja $2m$ on välien lukumäärä.

Koodataan ratkaisu MATLAB-koodiksi. Nyt (4.6) voidaan ratkaista seuraavasti. Ensin koodataan pääohjelma (recycle.m); kohdefunktion (4.6) minimointi tapahtuu jo edellä esitetyllä fmin-funktiolla.

```
% recycle.m
% 15.12.1998 (c) Juha Jaako

% Palautussuhteen optimi : R = 0.1893 ;

% Reaktioparametrit.
FA0 = 1 ;               % [mol/min]
k = 1 ;                 % [l/mol*min]
CA0 = 1 ;               % [mol/l]
XAf = 0.99 ;            % [dimensioton]

param = [ FA0  k  CA0  XAf ] ;

disp (' -- Minimoimalla ja numeerisesti integroimalla') ;
R = fmin ('simpson', 0, 1, [], param)    % Palautussuhteen minimi.
V = simpson (R, param)                   % Minimitilavuus.

disp (' -- Levenspiel') ;
Ropt = fmin ('(x+1)*log((1+0.01*x)/(0.01*x))', 0.1, 0.9)
Vopt = (Ropt+1)*log((1+0.01*Ropt)/(0.01*Ropt))
```

Ohjelma tulostaa seuraavaa. Ensin tulostetaan vastaus, joka saadaan minimoinnin ja numeerisen integroinnin avulla ja lopuksi vastaus, jossa käytetään viitteestä (Levenspiel 1972) saatuja kaavoja.

```
» recycle
 -- Minimoimalla ja numeerisesti integroimalla
R =     0.1893
V =     7.4587
 -- Levenspiel
Ropt =     0.1893
Vopt =     7.4587
```

Lisäksi tarvitaan tietysti minimoitava kohdefunktio eli kaava (4.6) koodattuna (simpson.m). Funktion tarvitseman parametrit välitetään parametrilistan avulla.

```
% simpson.m
% 15.12.1998 (c) Juha Jaako
% Palautussuhteen optimointi.
% Kreyszig E, Advanced engineering mathematics. s. 960-963.
% Levenspiel O, Chemical reaction engineering. s. 155-156.
% Numeerinen integrointi - Simpsonin sääntö.

function f = simpson (R, param)

% Palautussuhteen optimi : R = 0.1893 ;
```

```
% Reaktioparametrit.
FA0 = param(1) ;        % [mol/min]
k = param(2) ;          % [l/mol*min]
CA0 = param(3) ;        % [mol/l]
XAf = param(4) ;        % [dimensioton]

% Parametrit
a = (R/(R+1))*XAf ;     % Integroinnin aloituspiste.
b = XAf ;               %       "          lopetuspiste.
m2 = 500 ;              % Välien lukumäärä. Oltava SUURI ja PARILLINEN.
h = (b-a)/m2 ;          % Välin pituus.

% Laskentapisteet numeerista integrointia varten.
f = zeros (m2+1, 1) ;
for i = 0 : m2 ,
  x = a + h*i ;
  f(i+1) = 1/(x*(1-x)) ;
end

% Lasketaan Simpsonin parametrit.
s0 = f(1) + f(m2+1) ;
s1 = 0 ;
for i = 1 : 2 : (m2-1) ,
  s1 = s1 + f(i+1) ;
end
s2 = 0 ;
for i = 2 : 2 : (m2-2) ,
  s2 = s2 + f(i+1) ;
end

% Numeerisen integraalin arvo.
J = (h/3)*(s0 + 4*s1 + 2*s2) ;

vakio = (FA0*(R+1))/(k*CA0^2) ;
V = vakio*J ;

f = V ;
```

Koodi on suoraan Kreyszigin kirjassa (Kreyszig 1993, 961) olevan algoritmin koodaus. Koodi näyttää toimivan hyvin, mutta siinä on eräs ongelma; laskentapisteitä tarvitaan runsaasti. Välien lukumäärä on koodissa komentona

```
m2 = 500 ;              % Välien lukumäärä. Oltava SUURI ja PARILLINEN.
```

tämä on huomattavan suuri arvo. Pienemmillä arvoilla integraalin arvoksi tuli epätarkka arvo. Näyttää siltä, että funktio (simpson.m) ei ole sovelias numeeriseen integrointiin. Yleensä on syytä käyttää testattuja, tehokkaita algoritmeja. MATLAB ohjelmistossa on onneksi valmis numeerisen integroinnin funktio nimeltään quad8, josta on olemassa myös yksinkertaisempi versio quad.

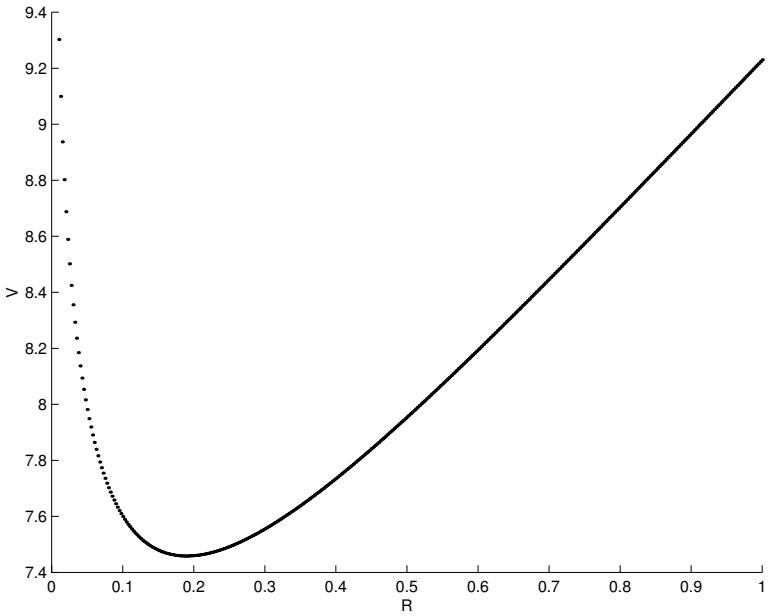
Nyt voimme koodata uuden pääohjelman (recycle1.m)

```
% recycle1.m
% 16.12.1998 (c) Juha Jaako

% Palautussuhteen optimi : R = 0.1893 ;

% Reaktioparametrit.
FA0 = 1 ;               % [mol/min]
k = 1 ;                 % [l/mol*min]
CA0 = 1 ;               % [mol/l]
XAf = 0.99 ;            % [dimensioton]

param = [ FA0  k  CA0  XAf ] ;

disp (' -- Minimoimalla ja numeerisesti integroimalla') ;
R = fmin ('simpson1', 0, 10, [], param)    % Palautussuhteen minimi.
V = simpson1 (R, param)                    % Minimitilavuus.

% Kuvaaja.
hold on ;
for R = 0.01 : 0.0025 : 1 ,
  plot (R, simpson1 (R, param), 'k.') ;
end
xlabel ('R') ; ylabel ('V') ;
hold off ;
print -deps2 -tiff c:\temp\recycle1 ;

disp (' -- Levenspiel') ;
Ropt = fmin ('(x+1)*log((1+0.01*x)/(0.01*x))', 0.1, 0.9)
Vopt = (Ropt+1)*log((1+0.01*Ropt)/(0.01*Ropt))
```

Koodi tulostaa seuraavan tekstin

```
» recycle1
 -- Minimoimalla ja numeerisesti integroimalla
R =    0.1893
V =    7.4587
 -- Levenspiel
Ropt =    0.1893
Vopt =    7.4587
```

ja kuvan 4.1



Kuva 4.1. Palautussuhde vs. tilavuus.

Lisäksi tarvitaan kohdefunktio (simpson1.m), jonka sisällä käytetään numeerista integraattoria

```
% simpson1.m - 16.12.1998 (c) Juha Jaako
% Palautussuhteen optimointi.
% Levenspiel O, Chemical reaction engineering. s. 155-156.
% Numeerinen integrointi - Matlabin quad8-funktio.

function f = simpson1 (R, param)

% Puretaan parametrivektori -> reaktioparametrit.
FA0 = param(1) ;        % [mol/min]
k   = param(2) ;        % [l/mol*min]
CA0 = param(3) ;        % [mol/l]
XAf = param(4) ;        % [dimensioton]

% Parametrit
a = (R/(R+1))*XAf ;   % Integroinnin aloituspiste.
b = XAf ;               %       "          lopetuspiste.
tol = 1e-3 ;            % Suhteellinen virhe.

% Numeerinen integrointi quad8-funktion avulla.
J = quad8 ('simpint', a, b, tol) ;        % Matlabin integraattori!!!

vakio = (FA0*(R+1))/(k*CA0^2) ;
V = vakio*J ;

f = V ;
```

ja integraattorin käyttämä funktio (simpint.m).

```
% simpint.m
% 16.12.1998 (c) Juha Jaako

% HUOM! x on vektori
%       f on vektori

function f = simpint (x)

f = 1./(x.*(1-x)) ;
```

huomaa, että simpint-funktio palauttaa vektorin.

Lopuksi voimme laskea vastaavan sekoitusreaktorin tilavuuden ($R = \infty$) seuraavasti (käytetään R:lle arvoa 1e10)[9].

```
» simpson1 (1e10, [ 1 1 1 0.99 ])
ans =  100.0000
```

eli saadaan sama vastaus kuin Levenspielkin on saanut. Kuvasta 5 näemme, että kun $R = 0$ (putkireaktori) niin $V = \infty$; tämä tarkoittaa sitä, ettei putkireaktorissa voida toteuttaa reaktiota (1). Sama voidaan todeta myös laskemalla seuraavaa (eli $R \rightarrow 0$).

```
» simpson (1e-2, [ 1 1 1 0.99 ])
ans =    9.3026
» simpson (1e-4, [ 1 1 1 0.99 ])
ans =   18.2042
» simpson (1e-6, [ 1 1 1 0.99 ])
ans =  678.2945
» simpson (1e-8, [ 1 1 1 0.99 ])
ans =  6.6678e+004
» simpson (1e-10, [ 1 1 1 0.99 ])
ans =  6.6667e+006
```

Taulukko 4.1: Käytetyt MATLAB-funktiot

funktio	kuvaus
quad8	numeerinen integraattori (» help quad8)
log	laskee luonnollisen logaritmin

[9] Huomaa, että tehtyjä funktioita voi kutsua suoraan MATLABin komentoriviltä.

5. Pienimmän neliösumman menetelmä - johdantoa

Datan mallitusongelma (malliparametrien sovittaminen annettuun dataan) tulee vastaan useasti prosessitekniikassa[10]. Jos malli on parametrien suhteen lineaarinen, voimme käyttää luvun 1 mukaista lähestymistapaa. Jos taas malli on parametrien suhteen epälineaarinen, ongelman ratkaisu on edellistä vaikeampaa.

Epälineaarisessa tapauksessa tavallisin lähestymistapa on erilaisten muunnosten käyttö; tavallisin tapaus on logaritmin otto. Esimerkki selventänee asiaa. Oletetaan seuraava malli

$$y(t) = a\ e^{-bt} \tag{5.1}$$

logaritmit ottamalla saadaan

$$\log[y(t)] = \log(a) - bt \tag{5.2}$$

ja merkitsemällä $c = \log(a)$ ja $Y = \log[y(t)]$ saadaan

$$Y = -bt + c \tag{5.3}$$

joka on nyt parametrien (b & c) suhteen lineaarinen. Ongelmana tässä lähestymistavassa on, että saadaan väärä vastaus (ks. esim. Jaako 1998, 6-12). Ennenkuin mennään epälineaariseen regressioanalyysiin, käydään läpi muutamia asioita.

Pienimmän neliösumman menetelmässä joudutaan minimoimaan seuraava funktio (kaksiparametrisessa tapauksessa)

$$\min\left\{ I(a,b) = \sum_{i=1}^{n} [f_i(a,b)]^2 \right\} \tag{5.4}$$

eli kohdefunktiona on neliösumma

$$I(a,b) = [f_1(a,b)]^2 + [f_2(a,b)]^2 + [f_3(a,b)]^2 + \ldots \tag{5.5}$$

Minimointitehtävä (5.4) voidaan ratkaista usealla eri optimointimenetelmällä[11]. Yhteistä näille menetelmille on, että tarvitaan minimointia varten aloituspiste, josta erilaisia algoritmeja käyttäen lähdetään hakemaan minimiä. Laskenta-algoritmia varten tarvitaan kohdefunktion gradienttivektori, toisista osittaisderivaatoista koostuva Hessin matriisi tai Hessin matriisia (tai sen käänteismatriisia) approksimoiva päivityskaava.

---

[10] Sopiva kirja aiheeseen tutustumiseen on Press *et al.* (1986, 498-577).

[11] Tässä käsitellään vain gradienttivektoria käyttäviä menetelmiä. On olemassa ns. suorahakumenetelmiä, jotka käyttävät vain funktion arvoja laskennassa, kts. esim. Reklaitis *et al.* (1983, 74-98). Matlabin fmins-funktio käyttää suorahakumenetelmää nimeltään Nelderin ja Meadin menetelmä.

Kaavan (5.4) tapauksessa saadaan

gradienttivektori[12] : $$G = \begin{pmatrix} \frac{\partial I}{\partial a} \\ \frac{\partial I}{\partial b} \end{pmatrix} \quad (5.6)$$

Hessin matriisi : $$J = \begin{pmatrix} \frac{\partial^2 I}{\partial a^2} & \frac{\partial^2 I}{\partial ab} \\ \frac{\partial^2 I}{\partial ba} & \frac{\partial^2 I}{\partial b^2} \end{pmatrix} \quad (5.7)$$

Tavallisimmat gradienttivektoria käyttävät optimointimenetelmät ovat (kts. esim. Reklaitis *et al.* 1983, 98-126)

Jyrkin lasku[13] (SD): $a_{i+1} = a_i - \lambda_{\min} G_i$ (5.8)

Newton (N): $a_{i+1} = a_i - J_i^{-1} G_i$ (5.9)

Parannettu Newton (MN) : $a_{i+1} = a_i - \lambda_{\min} J^{-1} G_i$ (5.10)

Kvasi-Newton (QN): $a_{i+1} = a_i - \lambda_{\min} H_i G_i$ (5.11)

Levenberg-Marquardt (LM) : $a_{i+1} = a_i - \left(J_i + \Lambda I\right)^{-1} G_i$ (5.12)

Parannettu Levenberg-Marquardt (MLM) : $a_{i+1} = a_i - \lambda_{\min} \left(H_i + \Lambda I\right)^{-1} G_i$ (5.12)

Kaavoissa (5.8)-(5.12) a_{i+1} on uusi laskentapiste, a_i on edellinen laskentapiste, $\lambda_{\min}$ laskettu askelpituus, G_i gradienttivektorin arvo edellisessä laskentapisteessä, J_i Hessin matriisin arvo edellisessä laskentapisteessä, H_i Hessin matriisin käänteismatriisia approksimoiva matriisin arvo edellisessä laskentapisteessä, I yksikkömatriisi ja Λ on kerroin, jonka aloitusarvona yleensä on 10^4.

---

[12] Gradienttivektori ilmoittaa suunnan, johon kuljettaessa funktio kasvaa voimakkaimmin. Huomaa, että minimoitaessa lähdetään vastakkaiseen suuntaan; miinusmerkki kaavoissa (5.8)-(5.12).

[13] Tunnetaan myös *Cauchyn* menetelmänä, kts. esim. Kreyszig (1993, 1086).

Menetelmistä voi todeta seuraavaa:

- SD on yksinkertaisin ja ehdottomasti varmin menetelmä; menetelmä takaa, että $I_{i+1} \le I_i$ eli minimi löytyy varmasti. Toisaalta SD on useimmissa tapauksissa hidas, jopa tolkuttoman hidas, eikä siten sovellu käytännön laskentaan.
- N on nopein menetelmä, kun ollaan riittävän lähellä minimiä mutta kaukana minimistä N on epävarma menetelmä. Menetelmä ei takaa, että $I_{i+1} \le I_i$; usein jopa $I_{i+1} >> I_i$. Em. syystä pelkkää N:ää ei käytetä minimointitehtävissä.
- Ottamalla N:ään mukaan askelpituuden laskenta saadaan MN, joka takaa, että $I_{i+1} \le I_i$. MN kuitenkin edellyttää osittaisderivaattojen tarkkaa laskentaa, joka ei aina ole mahdollista.
- QN menetelmän perusajatus on seuraava:
  - Laskennan alussa H_i-matriisi on yksikkömatriisi - eli haku vastaa SD-menetelmää.
  - Laskenna edistyessä H_i:matriisia päivitetään algoritmin avulla siten, että $H_i \to J_i^{-1}$. Eli laskennan edistyessä lähestytään N-menetelmää.
  - H_i-matriisin päivityskaavoja on useita - voidaan jopa sanoa, että aiheen tiimoilta on syntynyt kokonaisen kirjallisuuden laji. Tavallisimmin käytössä olevat päivityskaavat ovat Davidon-Fletcher-Powell (DFP) (Reklaitis *et al.* 1983, 115; Davidon 1959; Fletcher & Powell 1963; Jaako 1997, 65) ja Broyden-Fletcher-Goldfarb-Shanno (BFGS) (Gill *et al.* 1981, 119), joissa en ole omissa kokeiluissani huomannut sanottavia eroja.
- QN-menetelmässä[14] käytettävä DFP päivityskaava on seuraavanlainen

$$\text{DFP}: \; H_{i+1} = H_i - \frac{H_i y_i y_i^{'} H_i}{y_i^{'} H_i y_i} + \frac{s_i s_i^{'}}{s_i^{'} y_i} \tag{5.13}$$

jossa $y_i = G_i - G_{i-1}$ ja $s_i = a_i - a_{i-1}$. Luvussa 6 sovelletaan DFP-päivityskaavaa neliösumman minimointiin.

- QN-menetelmän huonona puolena on, että se joudutaan jossakin tapauksissa käynnistämään uudestaan.
- Alkuperäisessä LM-menetelmässä (päivityskaava $a_{i+1} = a_i - (J_i + \Lambda I)^{-1} G_i$) parametri Λ määrittää sekä hakusuunnan että haun askelpituuden. MLM-menetelmään on otettu mukaan optimaalinen askelpituuden laskenta - syy tähän selviää myöhemmin esitettävän esimerkin yhteydessä. MLM-menetelmää pidetään yleisesti parhaana neliösummien minimointimenetelmänä (ks. esim. Reklaitis *et al.* 1983, 107).

[14] Oikeastaan pitäisi puhua QN-menetelmistä, koska päivityskaavasta riippuen menetelmiä on useita.

6. Davidon-Fletcher-Powell-menetelmä

Seuraavissa esimerkeissä käytetään seuraavaa kohdefunktiota esimerkkinä. Kohdefunktio on neliösumma. Ongelman (6.1) ratkaisu on $\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

$$\min\left\{ f(x_1, x_2) = \left[(x_1 - 1)^2\right]^2 + (x_1 - x_2)^2 \right\} \qquad (6.1)$$

Ensiksi muodostetaan koodi, jonka avulla voidaan piirtää funktion $f(x_1, x_2) = (x_1 - 1)^4 + (x_1 - x_2)^2$ tasa-arvokäyrät (piirto1.m).

```
% piirto1.m - juha jaako 17.05.1999
% Piirretään funktion f(x1,x2) = (x1-1)^4 + (x1-x2)^2
% tasa-arvokäyrät.

function f = piirto1 (koord)

[x1, x2] = meshgrid (koord(1) : 0.1 : koord(2), ...
                     koord(3) : 0.1 : koord(4)) ;
A = (x1 - 1).^4 + (x1 - x2).^2 ;
ta = min(min(A)) : 1 : max(max(A)) ;
contour (x1, x2, A, ta) ;

% Tekstit.
xlabel ('\it{x_1}') ;
ylabel ('\it{x_2}') ;
title ('\it{f(x_1,x_2) = (x_1-1)^4 + (x_1-x_2)^2}') ;

f = 1 ;
```

Minimointi tehdään seuraavan pääohjelman avulla (dfp4.m). Haku alkaa pisteestä (-1, 2) ja hakua tehdään 20 kierrosta.

```
% dfp4.m
% juha jaako - 17.05.1999 - kts. Jaako (1997, s. 65) & James et al.
(1993, 748)

clear all ;                         % Putsataan muisti romusta.
format short ;
hold on ;

koord = [-1.5 2 0 3] ;
%axis (koord) ;
dummy = piirto1 (koord) ;

% Minimi pisteessä (1, 1).
mp = [1 ; 1] ; plot (mp(1), mp(2), 'k+') ;
tol1 = 1e-4 ;                       % Laskentatoleranssi.

% DFP-menetelmä.
ap = [ -1 ; 2 ] ;
plot (ap(1), ap(2), 'ko') ;
G = dfp4_d (ap) ;                   % Gradientin arvo pisteessä.

f = dfp4_f (ap) ;

iter = 0 ;                          % Alustetaan iteraatiolaskuri.
H = eye (2) ;                       % Yksikkömatriisi.
```

```
%while (max (abs (ap-mp)) > tol1) ,
for i = 1 : 20 ,                    % Lasketaan 20 kierrosta.
  T = H * G ;

  % Lasketaan askelpituus.
  t = fmin ('dfp4_m', -20, 20, [], ap, T) ;

  % Lasketaan uusi piste.
  ap_old = ap ;                     % Talletetaan vanha piste.
  ap = ap - t*T ;                   % Lasketaan uusi piste.
  plot (ap(1), ap(2), 'ko') ;       % Merkataan piste kuvaan.
  G_old = G ;                       % Talletetaan vanha gradientin arvo.
  G =  dfp4_d (ap) ;                % Gradientin arvo uudessa pisteessä.
  f = dfp4_f (ap) ;                 % Funktion arvo uudessa pisteessä.

  % H-matriisin päivitys.
  sk = ap - ap_old ;
  yk = G - G_old ;
  Bk = H ;

  % Davidon, Fletcher & Powell, kts. James s. 753
  te1 = Bk*yk*(yk')*Bk ;
  te2 = (yk')*Bk*yk ;
  te3 = sk*(sk') ;
  te4 = (sk')*yk ;
  H = H - (te1/te2) + (te3/te4) ;

  % Jyrkin lasku saadaan seuraavalla.
  % H = eye (2) ;

  iter = iter + 1 ;                 % Kasvatetaan iterointilaskuria.

end ;

ap
G
iter

hold off ;
```

Huomaa, että päivitysalgoritmi on koodattu seuraavasti

```
te1 = Bk*yk*(yk')*Bk ;
te2 = (yk')*Bk*yk ;
te3 = sk*(sk') ;
te4 = (sk')*yk ;
H = H - (te1/te2) + (te3/te4) ;
```

mutta se voidaan koodata myös yhdellä rivillä

```
H = H - ((Bk*yk*(yk')*Bk)/((yk')*Bk*yk)) + ((sk*(sk'))/((sk')*yk)) ;
```

Mielestäni ensimmäinen tapa on selkeämpi ja luettavampi. Lisäksi tarvitaan kolme funktiota (dfp4_f.m)

```
% dfp4_f.m
% Lasketaan funktion arvo.

function f = dfp4_f (x)

f = (x(1)-1)^4 + (x(1)-x(2))^2 ;
```

(dfp4_d.m)

```
% dfp4_d.m
% Lasketaan gradienttivektori.

function G = dfp4_d(x)

G = [ 4*(x(1)-1)^3 + 2*(x(1)-x(2)) ; -2*(x(1)-x(2)) ] ;
```

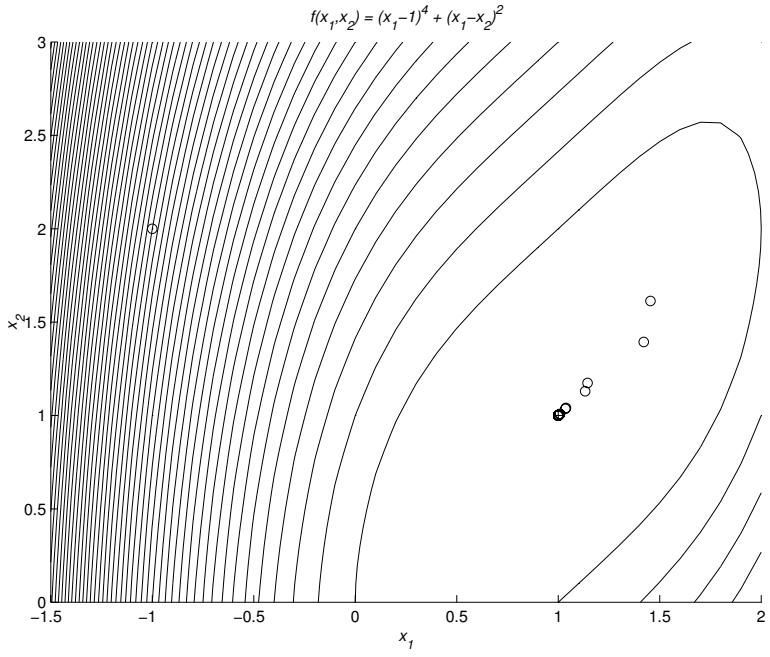
ja (dfp4_m.m).

```
% dfp4_mm
% Askelpituuden laskentafunktio.

function y = dfp4_m(x, ap, T)

xx = ap - x*T ;
y = dfp4_f(xx) ;
```

Ajamalla ohjelma, saadaan vastaukseksi

```
» dfp4
ap =
     1.0000
     1.0000
G =
   1.0e-012 *
     0.9044
    -0.9042
iter =
     20
```

ja kuva - huomaa miten kuva on piirretty ja miten eri komentoja on käytetty.



Kuva 6.1. Funktion $f(x_1, x_2) = (x_1 - 1)^4 + (x_1 - x_2)^2$ minimointi DFP-menetelmällä.

7. Levenberg-Marquardt-menetelmä

Levenberg-Marquardt-menetelmän laskenta-algoritmi ($a_{i+1} = a_i - (J_i + \Lambda I)^{-1} G_i$) löytyy ainakin lähteestä Reklaitis *et al.* (1983, 106) - se on esitetty myös liitteessä 1. Yleensä kirjoissa esitetyissä laskenta-algoritmeissa on virheitä - niin on tässäkin[15]. Seuraavassa on esitetty koodi, joka laskee minimin sijainnin (koodissa olevat STEPit viittaavat liitteen 1 STEPpeihin) (levmar.m)

```
% levmar.m
% Levenberg-Marquardt teht. 16
% Reklaitis, Ravindran & Ragsdell s. 106
% 22.01.1999 & 17.05.1999 (c) Juha Jaako

clear all ;                         % Putsataan muisti romusta.
format short ;
hold on ;

koord = [-1.5 2 0 3] ;
dummy = piirto1 (koord) ;

% Minimi pisteessä (1, 1).
mp = [1 ; 1] ;
plot (mp(1), mp(2), 'ko') ;

% Levenberg-Marquardt-menetelmä.

   % STEP 1
ap = [ -1 ; 2 ]  ;                  % Haun alkuarvo.
plot (ap(1), ap(2), 'k*') ;
M = 1e4 ;                           % Kierrosten maksimimäärä.
tol1 = 1e-10 ;                      % Laskentatoleranssi.

   % STEP 2
iter = 0 ;                          % Alustetaan iteraatiolaskuri.
L = 1e4 ;                           % Lambdan alkuarvo.
%L = 0 ;                            % Newton.

   % STEP 3
G = dfp4_d (ap) ;                   % Gradientin arvo.

f = dfp4_f (ap) ;                   % Funktion arvo.

I = eye (2) ;                       % Yksikkömatriisi.

   % STEP 4 & 5
while ( (max(abs(G))>tol1) & (iter<M) ) ,

%for i = 1 : 100 ,                    % Lasketaan 20 kierrosta.

   % STEP 6
  G = dfp4_d (ap) ;                 % Gradientin arvo.
  H = levmar_h (ap) ;               % Hessin matriisin arvo.
  temp1 = inv (H + L*I) ;
  temp2 = temp1*G ;

   % STEP 7
  ap_old = ap ;                     % Talletetaan vanha.
  ap = ap - temp2 ;
```

[15] Virheet käyvät selville, kun vertaat liitteen 1 algoritmia ja koodia!

```
    % STEP 8 - pieneni kö funkti on arvo?
  if ( dfp4_f(ap) < dfp4_f(ap_old) ) ,

    % STEP 9 - pi enennet ään l ambdaa
     L = L/2 ;
     iter = iter + 1 ;            % Kasvat et aan i ter oi nt i l askuri a.
     plot (ap(1), ap(2), 'k+') ;

  else

    % STEP 10 - kasvat et aan l ambdaa
     L = 2*L ;
     ap = ap_old ;
  end

end
abs (mp-ap)
G
H
iter

hold off ;
```

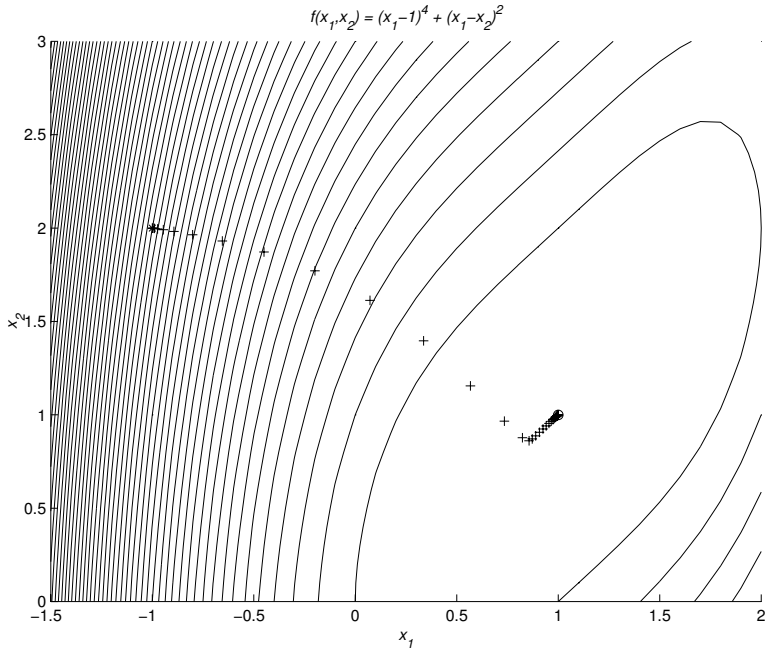
Lisäksi tarvitaan Hessin matriisin laskeva koodi (tässä toiset osittaisderivaatat on laskettu analyyttisesti eikä numeerisesti)

```
% levmar_h.m
% Lasket aan Hessi n matri i si .

functi on H = levmar_h (x)

H = [ (12*(x(1)-1)^2+2)          -2 ;
       -2                         2 ] ;
```

ohjelma käyttää lisäksi luvussa 6 esitettyjä funktioita (dfp4_m.m, dfp4_f.m, dfp4_d.m, piirto1.m). Nyt saamme vastaukseksi

```
» levmar
ans =
   1.0e-003 *
     0.1970
     0.1970
G =
   1.0e-010 *
    -0.6841
    -0.1621
H =
     2.0000    -2.0000
    -2.0000     2.0000
iter =
     38
```

ja kuvan



Kuva 7.1. Funktion $f(x_1, x_2) = (x_1 - 1)^4 + (x_1 - x_2)^2$ minimointi LM-menetelmällä.

Kuvasta 7.1 huomaa selvästi, että haku - varsinkaan alkuvaiheissaan - ei toimi kovin tehokkaasti. Tämä tehottomuus voidaan poistaa - luvussa 8 esitetään miten tämä tapahtuu.

8. Parannettu Levenberg-Marquardt

Luvun 7 algoritmia ($a_{i+1} = a_i - (J_i + \Lambda I)^{-1} G_i$) voidaan parantaa ottamalla mukaan askelpituuden laskenta ($a_{i+1} = a_i - \lambda_{\min} (H_i + \Lambda I)^{-1} G_i$). Tämä laskenta-algoritmi on esitetty liitteessä 1 (Edgar & Himmelblau 1988, 214). Koodattuna saamme seuraavanlaisen koodin (levmar1.m)

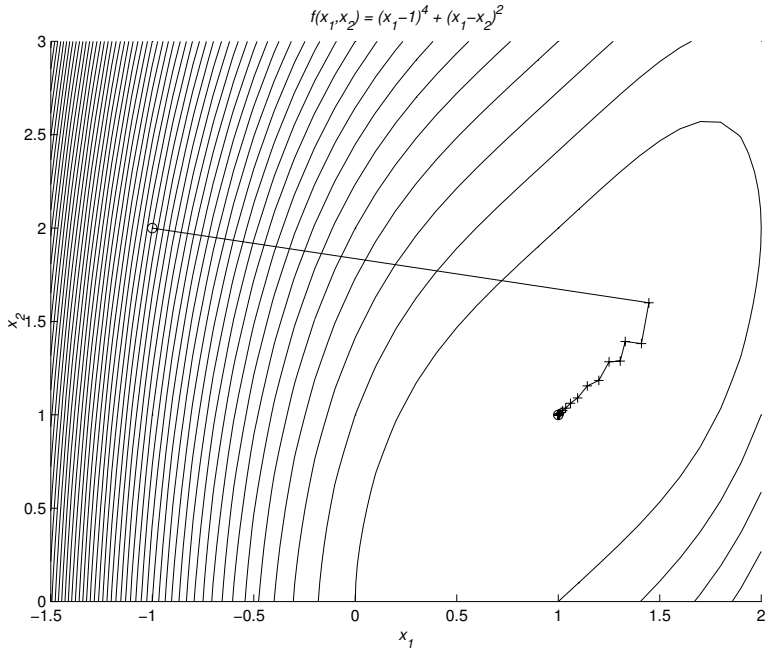
```
% levmar1.m
% Levenberg-Marquardt teht. 16
% Edgar & Himmelblau s. 214
% 25.01.1999 & 18.05.1999 (c) Juha Jaako

clear all ;                         % Putsataan muisti romusta.
format short ;
hold on ;

koord = [-1.5 2 0 3] ;
%axis ([-2 2 0 4]) ;
dummy = piirto1 (koord) ;

% Minimi pisteessä (1, 1).
mp = [1 ; 1] ;
plot (mp(1), mp(2), 'ko') ;

% Levenberg-Marquardt-menetelmä.

    % STEP 1
ap = [ -1 ; 2 ]  ;                  % Haun alkuarvo.
plot (ap(1), ap(2), 'ko') ;
M = 1e3 ;                           % Kierrosten maksimimäärä.
tol1 = 1e-10 ;                      % Laskentatoleranssi.

    % STEP 2
iter = 0 ;                          % Alustetaan iteraatiolaskuri.
Be = 1e3 ;                          % Beetan alkuarvo.

% Beeta voidaan laske myos ominaisarvojen avulla.
% Edgar & Himmelblau s. 214-215

    % STEP 3
G = dfp4_d (ap) ;                   % Gradientin arvo.
f = dfp4_f (ap) ;                   % Funktion arvo.
I = eye (2) ;                       % Yksikkomatriisi.

    % STEP 4
while ( (max(abs(G))>tol1) & (iter<M) ) ,

%for i = 1 : 100 ,                    % Lasketaan 20 kierrosta.

    % STEP 5
  G = dfp4_d (ap) ;                 % Gradientin arvo.
  H = levmar_h (ap) ;               % Hessin matriisin arvo.
  %disp (min (eig (H))) ;
  temp1 = inv (H + Be*I) ;
  temp2 = temp1*G ;

    % STEP 6
  ap_old = ap ;                     % Talletetaan vanha.
  t = fmin ('dfp4_m', 0, 1e3, [], ap, temp2) ;
  ap = ap - t*temp2 ;
```

```
    % STEP 7
  if ( dfp4_f(ap) < dfp4_f(ap_old) ) ,

    % STEP 8
     Be = Be/4 ;
     iter = iter + 1 ;              % Kasvatetaan iteraatiolaskuria.
     plot (ap(1), ap(2), 'k+') ;
     plot ([ap_old(1) ap(1)], [ap_old(2) ap(2)], 'k-') ;

  else

    % STEP 9
     Be = 2*Be ;
     ap = ap_old ;
  end

end
ap
G
H
iter

print -tiff -deps2 c:\temp\kkk ;
hold off ;
```

Kun ohjelma ajetaan, saadaan seuraavanlainen tulos

```
» levmar1
ap =
    1.0000
    1.0000
G =
  1.0e-010 *
   -0.7954
    0.7955
H =
    2.0000   -2.0000
   -2.0000    2.0000
iter =
    26
```

Ja seuraava kuva



Kuva 8.1. Funktion $f(x_1, x_2) = (x_1 - 1)^4 + (x_1 - x_2)^2$ minimointi MLM-menetelmällä.

Jos vertaa kuvaa 8.1 kuvaan 7.1, huomaat, että MLM-menetelmä on tehokkaampi kuin LM-menetelmä.

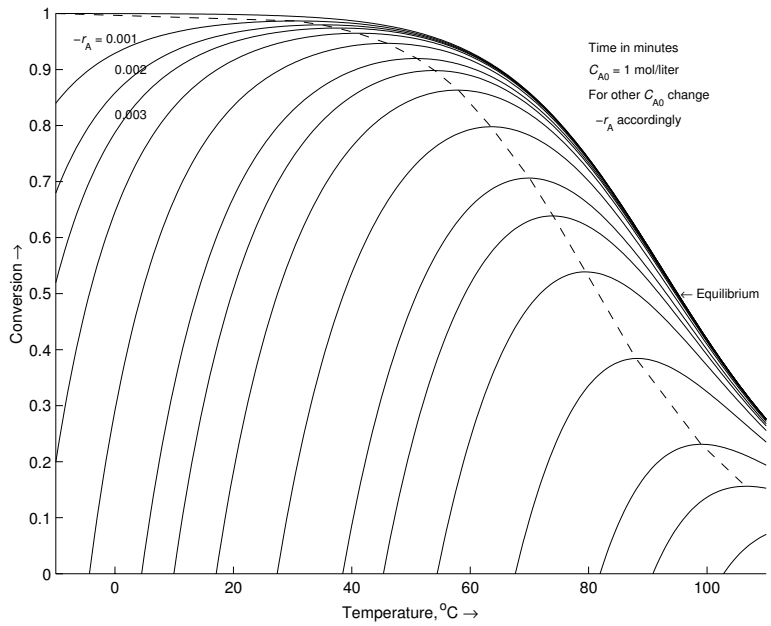
9. Monimutkaisen kuvan piirtäminen

Tähän sopii pienenä välipalana esitys monimutkaisen kuvan piirtämisestä. Levenspielin (1972, 233) kirjassa on kuva, johon on piirretty reaktionopeus, konversio ja lämpötila. Seuraavassa on muodostettu koodi, jonka tarkoituksena on näyttää, miten vastaava kuva voidaan piirtää ohjelmallisesti. Piirtokoodi on seuraavanlainen (temppro1.m).

```
% temppro1.m
% (c) Juha Jaako 08.01.1999 & 18.05.1999
% Levenspiel s. 233

clear all ;

XAe = 0.89 ;          % dimensioton
%CA0 = 4 ;             % mol/l
CA0 = 1 ;             % mol/l
E1 = 11600 ;          % cal/mol
E2 = 29600 ;
R  = 1.98 ;           % cal/(mol K)

T = 263 : 0.5 : 383 ;
N = size(T, 2) ;
mrA = [ 0 0.001 0.002 0.003 0.005 0.01 0.02 0.03 0.05 ...
        0.1   0.2   0.3   0.5   1    2    3    5    ] ;
M = size(mrA, 2) ;

hold on ;
axis ( [ (T(1)-273) (T(N)-273) 0 1 ] ) ;

for j = 1 : M,

  for i = 1 : N,
    k1 = exp(17.2-E1/(R*T(i))) ;
    k2 = exp(41.9-E2/(R*T(i))) ;
    temp1 = (k1*CA0-mrA(j))/CA0 ;
    temp2 = 1/(k1+k2) ;
    XA = temp1*temp2 ;
    xp(i) = T(i)-273 ;
    yp(i) = XA ;
  end

  % Piirretaan käyrä mrA:n vakioarvolla.
  plot(xp, yp, 'k-') ;

  % Käyrän maksimikohdat.
  [a,b] = max(yp) ;
  xmax(j) = xp(b) ;
  ymax(j) = yp(b) ;

end

% Maksimikohdat.
plot(xmax(1:j-1), ymax(1:j-1), 'k--') ;

% ************************************ Tekstit.
% Ensin akselit.
xlabel('Temperature, {^o}C \rightarrow') ;
ylabel('Conversion \rightarrow') ;
```

```
% Teksti oikeaan yläkulmaan.
siz = 30 ;
te(1,1:siz) = 'Time in minutes                    ' ;
te(2,1:siz) = '{\itC}_{A0} = 1 mol/liter          ' ;
te(3,1:siz) = 'For other {\itC}_{A0} change       ' ;
te(4,1:siz) = '  -{\itr}_{A} accordingly          ' ;
for j = 1 : 4 ,
  text (80, 0.94-(j-1)*0.045, te(j,1:siz), 'FontName', 'Arial',
'FontSize', 8 ) ;
end

% Teksti oikealle keskelle.
text (95.5, 0.5, '\leftarrow Equilibrium', 'FontName', 'Arial',
'FontSize', 8 ) ;

% Tekstit käyrien yhteyteen - ei kaikkia.
text (-7, 0.95, [ '-{\itr}_{A} = ' num2str(mrA(2)) ], 'FontName',
'Arial', 'FontSize', 8 ) ;
text ( 0, 0.90, [ num2str(mrA(3)) ], 'FontName', 'Arial',
'FontSize', 8 ) ;
text ( 0, 0.82, [ num2str(mrA(4)) ], 'FontName', 'Arial',
'FontSize', 8 ) ;

hold off ;
print -tiff -deps2 c:\temp\temppro1 ;
```

Ja kuvaksi saadaan.



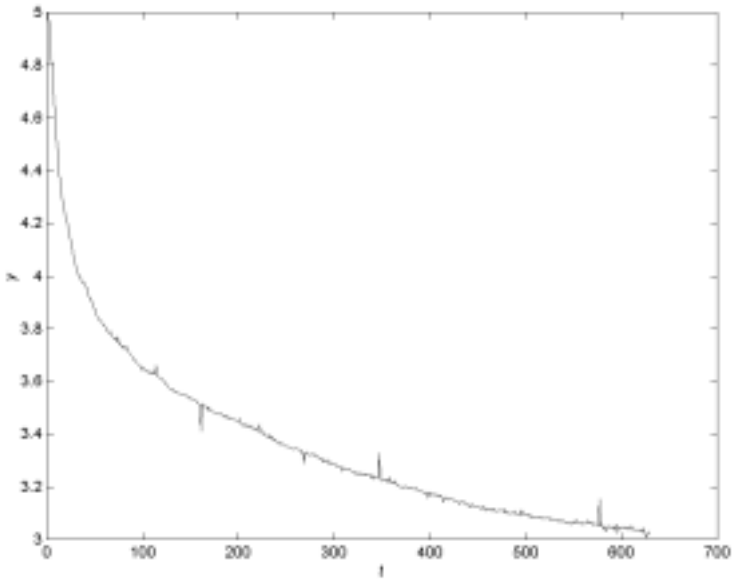
Kuva 9.1. Monimutkainen kuva.

Vertaa kuvaa Levenspielin kirjassa olevaan.

10. Mallidatan sovitus

Tarkastellaan seuraavaa ongelmaa, joka on esitetty Prosessien optimointi -kurssin tentissä 19.03.1999.

Kuvaan 10.1 on piirretty erään kokeen tulokset. Vaaka-akselilla on aika ja pysty-akselilla on tutkittavan aineen eräs ominaisuus (y). Huomaa, että kuva on JPEG-kuva - siis huonolaatuisempi kuin esimerkiksi kuva 9.1.



Kuva 10.1.

Tehtävänä on mallittaa kokeellisesti em. ominaisuuden riippuvuus ajasta.

Mallivaihtoehdot ovat seuraavat[16]

Mallin numero	Mallivaihtoehto
1	$y(t)=\frac{a}{t}+b$
2	$y(t)=a\,t+b$
3	$y(t)=a\,t^2+b\,t+c$
4	$y(t)=a\,e^{-bt}+c$
5	$y(t)=\frac{a}{\log(bt)}+c$

Mikä mallivaihtoehto sopii parhaiten? Miksi? Miten sovitat malliparametrit?

Valitaan ensin sopiva mallivaihtoehto.

Kuvan 10.1 dataa ei selvästikään pysty mallittamaan lineaarisella mallilla, joten malli 2 on poissa laskuista. Malli numero 3 edustaa ylös- tai alaspäin aukeavaa parabelia, joten sekään ei tule kysymykseen. Mallin numero 1 ongelma on, että ajan hetkellä nolla se antaisi aineen ominaisuudelle äärettömän suuria arvoja. Saman tyyppinen ongelma on mallilla 5; nollasta ei voi ottaa logaritmia ja kun vielä $\log\left(\frac{1}{\infty}\right)=-\infty$, ei malli 5 voi myöskään olla oikea. Jäljelle jää malli 4, joka on toimiva malli.[17]

$$y(t)=a\,e^{-bt}+c \qquad (10.1)$$

Malliparametrien sovittaminen tapahtuu seuraavasti. Tehtävän havainnollistamiseksi ei oteta käyttöön koko mittausdataa vaan vain osa, joka on esitetty taulukossa 10.1.

Taulukko 10.1

i	1	2	3	4	5	6	7
t	0	50	100	150	200	250	300
y	5.0	3.87	3.6535	3.5365	3.4495	3.352	3.2835

i	8	9	10	11	12	13
t	350	400	450	500	550	600
y	3.2215	3.179	3.125	3.097	3.07	3.047

[16] Mallivaihtoehdot tekn. yo. Katja Viirretin ja tekn. yo. Satu Kivelän tekemästä säätötekniikan erikoistyöstä.

[17] Alkuperäinen analyysi tekn. yo. Visa Virtasen tenttivastauksessa.

Kuva taulukon 10.1 datasta voidaan piirtää kuva seuraavanlaisen koodin avulla.

```
% koe21.m
% 20.05.1999 (c) Juha Jaako

clear all;
hold on;
t = [ 0      50     100    150    200    250    300 ...
      350    400    450    500    550    600    ];
y = [ 5.0000 3.8700 3.6535 3.5365 3.4495 3.3520 3.2835 ...
      3.2215 3.1790 3.1250 3.0970 3.0700 3.0470 ];
plot(t,y,'ko');
xlabel('\it{t}');
ylabel('\it{y}');
hold off;
```

Ongelmana on määrittää parametrit a, b ja c niin, että malli kuvaa datajoukkoa mahdollisimman hyvin. Mikäli olisi tarjolla vain 3 datapistettä, ratkaisu voitaisiin laskea muodostamalla yhtälöryhmä, jossa on kolme tuntematonta ja kolme yhtälöä. Nyt kuitenkin saamme yhtälöryhmän, jossa on kolme tuntematonta ja 13 yhtälöä. Tällainen ongelma voidaan ratkaista pienimmän neliösumman menetelmällä (kts. luku 5!), joka ongelmamme tapauksessa saa muodon

$$\min\left\{ I(a,b,c) = \sum_{i=1}^{13} \left[y_i - a\exp(-bt_i) - c \right]^2 \right\} \tag{10.2}$$

Eli ongelmamme on optimointitehtävä; I täytyy minimoida. Sijoittamalla taulukon 10.1 data saadaan kohdefunktio muotoon.

$$\min\left\{ \begin{aligned} I(a,b,c) &= \left[5 - a - c\right]^2 + \left[3.87 - a\exp(-50b) - c\right]^2 + \left[3.6535 - a\exp(-100b) - c\right]^2 \\ &+ \left[3.5365 - a\exp(-150b) - c\right]^2 + \left[3.4495 - a\exp(-200b) - c\right]^2 \\ &+ \left[3.352 - a\exp(-250b) - c\right]^2 + \left[3.2835 - a\exp(-300b) - c\right]^2 \\ &+ \left[3.2215 - a\exp(-350b) - c\right]^2 + \left[3.1790 - a\exp(-400b) - c\right]^2 \\ &+ \left[3.125 - a\exp(-450b) - c\right]^2 + \left[3.097 - a\exp(-500b) - c\right]^2 \\ &+ \left[3.07 - a\exp(-550b) - c\right]^2 + \left[3.047 - a\exp(-600b) - c\right]^2 \end{aligned} \right\} \tag{10.3}$$

[18]

Selvästikin (10.3):n ratkaiseminen esim. analyyttisesti on aika toivoton tehtävä. Ennenkuin ongelma ratkaistaan, voidaan tarkastella, missä parametrien arvot suunnilleen ovat. Nyt tiedämme, että

$$\lim_{t\to\infty}\left[a\exp(-bt) + c\right] \to c \tag{10.4}$$

joten voimme arvioida, että $c \approx 3$ (t = 600, y = 3.0470). Samoin voimme laskea, että kun t = 0 niin y = 5, joten saadaan

$$a\exp(-b*0) + c = 5 \tag{10.5}$$

$$a + c = 5 \Rightarrow a \approx 2 \text{ kun } c \approx 3 \tag{10.6}$$

[18] Ongelman (10.3) ratkaisu on a = 1.7451, b = 0.012457 ja c = 3.1632.

Kun nyt olemme arvioineet arvot a:lle ja c:lle, saamme kun (t = 50)

$$2\exp(-50b)+3=3.87 \tag{10.7}$$

$$\exp(-50b)=\frac{3.87-3}{2}=\frac{0.87}{2}=0.435 \tag{10.8}$$

$$b=\frac{\log(0.435)}{-50}=0.016648\approx 0.017 \tag{10.9}$$

Näillä arvoilla neliösumman (kohdefunktion) arvoksi tulee (a=2, b = 0.017, c = 3) 2.2387.

Seuraavassa ratkaistaan (10.3) seuraavilla menetelmillä: DFP, LM ja MLM. Käydään ensin kuitenkin läpi, miten lasketaan menetelmissä tarvittavat gradienttivektori ja Hessin matriisi. Oletetaan, että sovitettava malli on muotoa[19]

$$y=y(x;\mathrm{a}) \text{ , esim. } y=a_1\exp(-a_2x)+a_3 \tag{10.10}$$

minimoitava kohdefunktio on siten

$$F(a)=\sum_{i=1}^{N}[y_i-y(x_i;\mathrm{a})]^2 \tag{10.11}$$

Gradienttivektori saadaan seuraavasti

$$\frac{\partial F(a)}{\partial a_k}=-2\sum_{i=1}^{N}[y_i-y(x_i;\mathrm{a})]\frac{\partial y(x_i;\mathrm{a})}{\partial a_k} \tag{10.12}$$

ja Hessin (H) matriisi seuraavasti

$$\frac{\partial^2 F(a)}{\partial a_k\partial a_l}=2\sum_{i=1}^{N}\left\{\frac{\partial y(x_i;\mathrm{a})}{\partial a_k}\frac{\partial y(x_i;\mathrm{a})}{\partial a_l}-[y_i-y(x_i;\mathrm{a})]\frac{\partial^2 y_i(x_i;\mathrm{a})}{\partial a_l\partial a_k}\right\} \tag{10.13}$$

Jos esimerkkifunktionamme on $y=y(x;\mathrm{a})=a_1\exp(-a_2x)+a_3$, voidaan laskea

$$\begin{pmatrix}\frac{\partial y(x_i;\mathrm{a})}{\partial a_1}\\ \frac{\partial y(x_i;\mathrm{a})}{\partial a_2}\\ \frac{\partial y(x_i;\mathrm{a})}{\partial a_3}\end{pmatrix}=\begin{pmatrix}\exp(-a_2x_i)\\ -x_i\ a_1\exp(-a_2x_i)\\ 1\end{pmatrix} \tag{10.14}$$

[19] Tarkempi tarkastelu löytyy viitteestä Press *et al.* (1986, 522).

ja alkioita Hessin matriisiin

$$H(1,1)=\frac{\partial^2 F(a)}{\partial a_1 \partial a_1}=2\sum_{i=1}^{N}\left\{\frac{\partial y(x_i;\mathrm{a})}{\partial a_1}\frac{\partial y(x_i;\mathrm{a})}{\partial a_1}-\left[y_i-y(x_i;\mathrm{a})\right]\frac{\partial^2 y_i(x_i;\mathrm{a})}{\partial a_1 \partial a_1}\right\} \tag{10.15.1}$$

$$H(1,1)=\frac{\partial^2 F(a)}{\partial a_1 \partial a_1}=2\sum_{i=1}^{N}\left\{\exp(-a_2\ x_i)\ \exp(-a_2\ x_i)\right\} \tag{10.15.2}$$

jne.

Seuraavassa ratkaistaan minimointiongelma (10.3) ensin DFP-menetelmällä, seuraavaksi MLM-menetelmällä ja lopuksi käytetään MLM-menetelmää siten, että Hessin matriisi lasketaan käyttämällä Press *et al.* (1986, 523) esittämää yksinkertaistusta. Kaikissa menetelmissä haun aloituspisteenä on origo.

Kaikki menetelmät käyttävät seuraavia funktioita. Ensiksikin funktio, joka laskee kohdefunktion (10.3) arvot laskentapisteessä (dfp5_f.m). Tätä funktiota käyttää myöhemmin esiteltävä askelpituuden laskentafunktio (dfp5_m.m).

```
% dfp5_f.m
% Lasketaan funktion arvo.
% 21.05.1999 (c) Juha Jaako

function f = dfp5_f (x)

% Data.
xdata = [ 0       50      100     150     200     250     300 ...
          350     400     450     500     550     600     ] ;
ydata = [ 5.0000 3.8700 3.6535 3.5365 3.4495 3.3520 3.2835 ...
          3.2215 3.1790 3.1250 3.0970 3.0700 3.0470 ] ;
N = size (xdata, 2) ;

a = x(1) ; b = x(2) ; c = x(3) ;

% Lasketaan kohdefunktion arvo.
f = 0 ;
for i = 1 : N,
  f = f + ( ydata(i)-a*exp(-b*xdata(i))-c )^2 ;
end
```

Seuraavaksi tarvitaan funktio, joka laskee gradienttivektorin (dfp5_a.m)

```
% dfp5_a.m
% 25.01.1999 (c) Juha Jaako
% Lasketaan gradienttivektori analyyttisesti
% kts. Press s. 522

function G = dfp5_a (x)

% Data.
xd = [ 0       50      100     150     200     250     300 ...
       350     400     450     500     550     600     ] ;
yd = [ 5.0000 3.8700 3.6535 3.5365 3.4495 3.3520 3.2835 ...
       3.2215 3.1790 3.1250 3.0970 3.0700 3.0470 ] ;
N = size (xd, 2) ;

a = x(1) ; b = x(2) ; c = x(3) ;
```

```
% Lasketaan gradientin arvo.
G = zeros (3, 1) ;          % Nollataan gradienttivektori.
for i = 1 : N,
  t1 = ( yd(i)-a*exp(-b*xd(i))-c ) ;
  G(1) = G(1) + t1*exp(-b*xd(i)) ;
  G(2) = G(2) + t1*(-a*xd(i)*exp(-b*xd(i))) ;
  G(3) = G(3) + t1*(+1) ;
end

G = -2*G ;
```

Mikäli ei voida laskea gradienttivektoria analyyttisesti, se voidaan laskea numeerisesti (kts. esim. Kreyszig 1993, 966), joka tapa onkin käytännössä selkeämpi; saadaan (dfp5_d.m)

```
% dfp5_d.m
% Lasketaan gradienttivektori numeerisesti.
% 21.05.1999 (c) Juha Jaako
% Kreyszig (1993, 966)

function G = dfp5_d (x)

h = 1e-4 ;      % Tama on sopiva arvo h:lle.
                % h = 1e-5 on liian pieni.

f0 = dfp5_f ([ x(1)-2*h  x(2)       x(3) ]) ;
f1 = dfp5_f ([ x(1)-h    x(2)       x(3) ]) ;
f3 = dfp5_f ([ x(1)+h    x(2)       x(3) ]) ;
f4 = dfp5_f ([ x(1)+2*h  x(2)       x(3) ]) ;
G1 = (f0 - 8*f1 + 8*f3 - f4) / (12*h) ;

f0 = dfp5_f ([ x(1)       x(2)-2*h  x(3) ]) ;
f1 = dfp5_f ([ x(1)       x(2)-h    x(3) ]) ;
f3 = dfp5_f ([ x(1)       x(2)+h    x(3) ]) ;
f4 = dfp5_f ([ x(1)       x(2)+2*h  x(3) ]) ;
G2 = (f0 - 8*f1 + 8*f3 - f4) / (12*h) ;

f0 = dfp5_f ([ x(1)       x(2)       x(3)-2*h ]) ;
f1 = dfp5_f ([ x(1)       x(2)       x(3)-h   ]) ;
f3 = dfp5_f ([ x(1)       x(2)       x(3)+h   ]) ;
f4 = dfp5_f ([ x(1)       x(2)       x(3)+2*h ]) ;
G3 = (f0 - 8*f1 + 8*f3 - f4) / (12*h) ;

G = [ G1 ; G2 ; G3 ] ;
```

Esimerkiksi pisteessä (1, 1, 1) saadaan

```
» format long
» dfp5_a ([ 1 1 1])
ans =
  -6.00000000000000
   0.00000000000000
 -61.76900000000001
```

ja

```
» dfp5_d ([ 1 1 1])
ans =
  -5.99999999995049
  -0.00000000003553
 -61.76899999995082
```

eli analyyttisesti laskettu gradienttivektori on melkein sama kuin numeerisesti laskettu gradienttivektori.

Hessin matriisin alkiot voidaan laskea seuraavasti (dfp5_H.m)

```
% dfp5_H.m
% 25.01.1999 (c) Juha Jaako
% Lasketaan Hessin matriisi analyyttisesti.
% kts. Press s. 522

function H = dfp5_H(x)

% Data.
xd =  [ 0      50     100    150    200    250    300 ...
        350    400    450    500    550    600    ] ;
yd =  [ 5.0000 3.8700 3.6535 3.5365 3.4495 3.3520 3.2835 ...
        3.2215 3.1790 3.1250 3.0970 3.0700 3.0470 ] ;
N = size(xd, 2) ;

a = x(1) ; b = x(2) ; c = x(3) ;

% Lasketaan Hessin matriisin arvo.
H = zeros(3) ;          % Nollataan matriisi.
for i = 1 : N,
  t2 = exp(-b*xd(i)) ;

  da = t2 ;
  db = -a*xd(i)*t2 ;
  dc = 1 ;

  dada = 0 ;
  dadb = -xd(i)*t2 ;
  dadc = 0 ;

  dbda =  -xd(i)*t2 ;
  dbdb = a*xd(i)*xd(i)*t2 ;
  dbdc = 0 ;

  dcda = 0 ;
  dcdb = 0 ;
  dcdc = 0 ;

  t1 = yd(i) - a*t2-c  ;

  H(1,1) = H(1,1) + (da*da-t1*dada) ;
  H(2,1) = H(2,1) + (db*da-t1*dadb) ;
  H(3,1) = H(3,1) + (dc*da-t1*dadc) ;

  H(1,2) = H(1,2) + (da*db-t1*dbda) ;
  H(2,2) = H(2,2) + (db*db-t1*dbdb) ;
  H(3,2) = H(3,2) + (dc*db-t1*dbdc) ;

  H(1,3) = H(1,3) + (da*dc-t1*dcda) ;
  H(2,3) = H(2,3) + (db*dc-t1*dcdb) ;
  H(3,3) = H(3,3) + (dc*dc-t1*dcdc) ;

end

H = 2*H ;
```

Tekemättä suurta virhettä[20] voidaan Hessin matriisi laskea myös paljon yksinkertaisemmin seuraavasti (dfp5_H1.m)

```
% dfp5_H1.m
% 25.01.1999 & 27.01.1999 (c) Juha Jaako
% Lasketaan Hessin matriisi analyyttisesti.
% kts. Press s. 522
% yksinkertaistettu dfp5_H.m:sta, kts Press s. 523

function H = dfp5_H1 (x)

% Data.
xd =  [ 0       50      100     150     200     250     300 ...
        350     400     450     500     550     600     ] ;
yd =  [ 5.0000 3.8700 3.6535 3.5365 3.4495 3.3520 3.2835 ...
        3.2215 3.1790 3.1250 3.0970 3.0700 3.0470 ] ;
N = size (xd, 2) ;

a = x(1) ; b = x(2) ; c = x(3) ;

% Lasketaan Hessin matriisin likiarvo.
H = zeros (3) ;
for i = 1 : N,

  da =          exp(-b*xd(i)) ;
  db = -xd(i)*a*exp(-b*xd(i)) ;
  dc = 1 ;

  V1 = [ da ; db ; dc ] ;
  V2 = V1' ;

  H = H + V1*V2 ;

end

H = 2*H ;
```

Lisäksi tarvitaan vielä edellä mainittu askelpituuden laskentafunktio (dfp5_m.m)

```
% dfp5_m.m
% Askelpituuden laskentafunktio.

function y = dfp5_m(t, ap, T)

x = ap - t*T ;
y = dfp5_f (x) ;
```

Ratkaistaan ongelma (10.3) ensin DFP-menetelmällä. Käytössä on seuraava koodi (dfp5a.m)

```
% dfp5a.m
% 26.01.1999 (c) Juha Jaako
% Analyyttiset derivaatat

clear all ;                         % Putsataan muisti romusta.
format short e ;

% Minimi pisteessa.
mp = [ 1.7451 ; 0.0125 ; 3.1632 ] ;
tol1 = 1e-6 ;                       % Laskentatoleranssi.
```

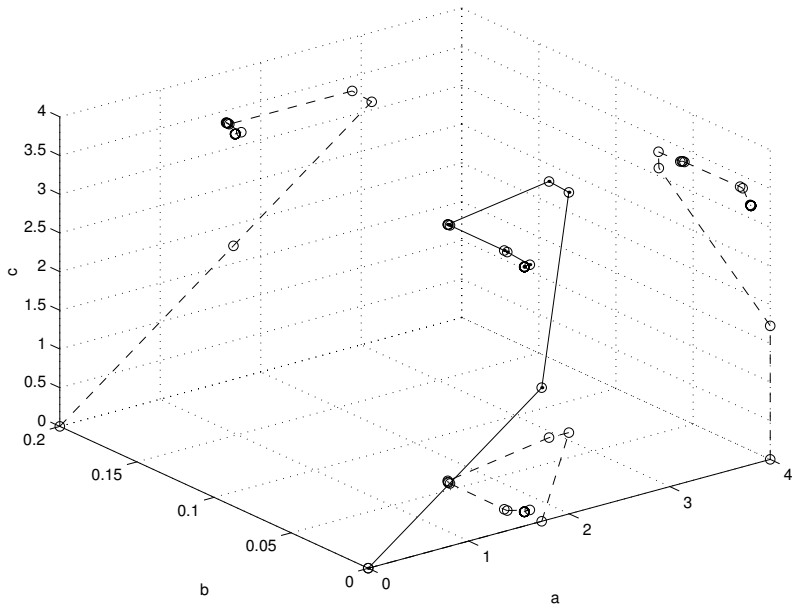
[20] Perustelut seuraavan koodin käytölle löytyvät lähteestä Press *et al.* (1986, 523). Tässä perustelut ohitetaan.

```
% DFP-menetelmä.
iter = 1 ;                          % Alustetaan iteraatiolaskuri.
%ap = [ 2.0000 ; 0.0170 ; 3.0000 ] ;
ap = [ 0 ; 0 ; 0 ] ;
pl1(iter) = ap(1) ; pl2(iter) = ap(2) ; pl3(iter) = ap(3) ;
G = dfp5_a (ap) ;                   % Gradientin arvo pisteessä.
f(iter) = dfp5_f (ap) ;             % Funktion arvo alkupisteessä.
H = eye (3) ;                       % Yksikkömatriisi.

loppu = 20 ;                        % Lasketaan 20 kierrosta.

while (max (abs (G)) > tol1) ,
%for i = 1 : loppu ,
  T = H * G ;

  % Lasketaan askelpituus.
  t = fmin ('dfp5_mi', -20, 20, [], ap, T) ;

  % Lasketaan uusi piste.
  ap_old = ap ;                     % Talletetaan vanha piste.
  ap = ap - t*T ;                   % Lasketaan uusi piste.
  G_old = G ;                       % Talletetaan vanha gradientin arvo.
  G = dfp5_a (ap) ;                 % Gradientin arvo uudessa pisteessä.
  iter = iter + 1 ;                 % Kasvatetaan iteraatiolaskuria.
  pl1(iter) = ap(1) ; pl2(iter) = ap(2) ; pl3(iter) = ap(3) ;
  f(iter) = dfp5_f (ap) ;           % Funktion arvo uudessa pisteessä.

  % H-matriisin päivitys.
  sk = ap - ap_old ;
  yk = G - G_old ;
  Bk = H ;

  % Davidon, Fletcher & Powell, kts. James s. 753
  te1 = Bk*yk*(yk')*Bk ;
  te2 = (yk')*Bk*yk ;
  te3 = sk*(sk') ;
  te4 = (sk')*yk ;
  H = H - (te1/te2) + (te3/te4) ;

end ;

tol1
ap
disp ('kohdefunktion arvo =') ;
disp (f(iter)) ;
G
iter

xx = 1:iter ;
%semilogy (xx(3:iter), f(3:iter), 'k+') ;
%pause (3) ;

NLT = zeros (1, iter) ;
N4T = 4*ones (1, iter) ;
N1T = 0.2*ones (1, iter) ;

plot3 (pl1, pl2, pl3, 'k.-',  pl1, pl2, pl3, 'ko' , ...
       N4T, pl2, pl3, 'b--' , N4T, pl2, pl3, 'bo' , ...
       pl1, N1T, pl3, 'b--' , pl1, N1T, pl3, 'bo' , ...
       pl1, pl2, NLT, 'b--' , pl1, pl2, NLT, 'bo'  ...
      ) ;
axis ( [0 4 0 0.2 0 4] ) ;
grid ;
xlabel ('a') ; ylabel ('b') ; zlabel ('c') ;

print -tiff -deps2 c:\temp\adfp5 ;
```

Ohjelma tulostaa seuraavaa

```
» dfp5a
tol1 =
   1.0000e-006
ap =
   1.7451e+000
   1.2457e-002
   3.1632e+000
kohdefunktion arvo =
   1.4155e-001
G =
  -2.3238e-008
   6.6878e-008
  -7.5542e-009
iter =
      16
```

ja myös seuraavan kuvan



Kuva 10.1. Mallidatan sovitus DFP-menetelmällä

Huomaa, että kuva on 3-ulotteinen (plot3-funktio). Kuvasta käy ilmi, miten haku lähtee origosta ja päätyy optimispisteeseen; lisäksi haku on projisoitu tasoille a-b, a-c ja b-c.

MLM-menetelmä koodattuna antaa seuraavan koodin (levmar2.m).

```
% levmar2.m
% Levenberg-Marquardt ProsOpt 98 h
% Edgar & Himmelblau s. 214
% 25.01.1999 & 26.01.1999 (c) Juha Jaako

clear all ;                          % Putsataan muistiromusta.
format short ;
```

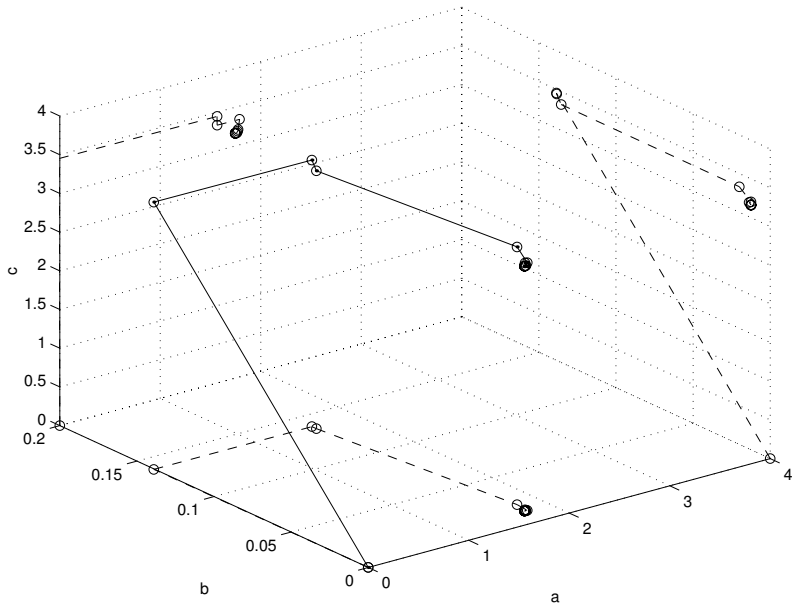
```
% Minimi pisteessa.
mp = [ 1.7451 ; 0.0125 ; 3.1632 ] ;

% Levenberg-Marquardt-menetelmä.

   % STEP 1
ap = [ 0 ; 0 ; 0 ]  ;            % Haun alkuarvo.
disp (ap') ;
M = 1e3 ;                        % Kierrosten maksimimaara.
tol1 = 1e-6 ;                    % Laskentatoleranssi.

   % STEP 2
iter = 1 ;                       % Alustetaan iteraatiolaskuri.
riter = 1 ;
pl1(iter) = ap(1) ;
pl2(iter) = ap(2) ;
pl3(iter) = ap(3) ;
Be = 1e3 ;                       % Beetan alkuarvo.

% Beeta voidaan laskea myos ominaisarvojen avulla.
% Edgar & Himmelblau s. 214-215

   % STEP 3
G = dfp5_a (ap) ;                % Gradientin arvo.
f = dfp5_f (ap) ;                % Funktion arvo.
I = eye (3) ;                    % Yksikkomatriisi.

   % STEP 4
while ( (max(abs(G))>tol1) & (iter<M) ) ,

%for i = 1 : 20 ,                % Lasketaan 20 kierrosta.

   % STEP 5
  G = dfp5_a (ap) ;              % Gradientin arvo.
  H = dfp5_H (ap) ;              % Hessin matriisin arvo.
  temp1 = inv (H + Be*I) ;
  temp2 = temp1*G ;

   % STEP 6
  ap_old = ap ;                  % Talletetaan vanha.
  t = fmin ('dfp5_m', 0, 1e3, [], ap, temp2) ;
  ap = ap - t*temp2 ;

   % STEP 7
  if ( dfp5_f(ap) < dfp5_f(ap_old) ) ,

   % STEP 8 - Kohti Newtonin menetelmaa.
    Be = Be/4 ;
    iter = iter + 1 ;            % Kasvatetaan iteraatiolaskuria.
    pl1(iter) = ap(1) ;
    pl2(iter) = ap(2) ;
    pl3(iter) = ap(3) ;
    %disp (ap') ;
    %disp (dfp5_f (ap)) ;        % Funktion arvo.

  else

   % STEP 9 - Kohti jyrkimman laskun menetelmaa.
    %disp ('kasvatan Beetaa') ;
    Be = 2*Be ;
    ap = ap_old ;

  end

  riter = riter + 1 ;

end
```

```
ap
dfp5_f (ap)
G
H
iter
riter

NLT = zeros (1, iter) ;
N4T = 4*ones (1, iter) ;
N1T = 0.2*ones (1, iter) ;

plot3 (pl1, pl2, pl3, 'k.-',   pl1, pl2, pl3, 'ko' , ...
       N4T, pl2, pl3, 'b--' , N4T, pl2, pl3, 'bo' , ...
       pl1, N1T, pl3, 'b--' , pl1, N1T, pl3, 'bo' , ...
       pl1, pl2, NLT, 'b--' , pl1, pl2, NLT, 'bo'  ...
      ) ;
axis ( [0 4 0 0.2 0 4] ) ;
grid ;
xlabel ('a') ; ylabel ('b') ; zlabel ('c') ;
print -tiff -deps2 c:\temp\adfp5 ;
```

Ohjelma tulostaa seuraavaa

```
» levmar2
     0      0      0
ap =
    1.7451
    0.0125
    3.1632
ans =
    0.1415
G =
  1.0e-007 *
    0.0108
   -0.2891
    0.0100
H =
  1.0e+004 *
    0.0003   -0.0099    0.0004
   -0.0099    1.2451   -0.0434
    0.0004   -0.0434    0.0026
iter =
    13
riter =
    19
```

ja lisäksi kuvan



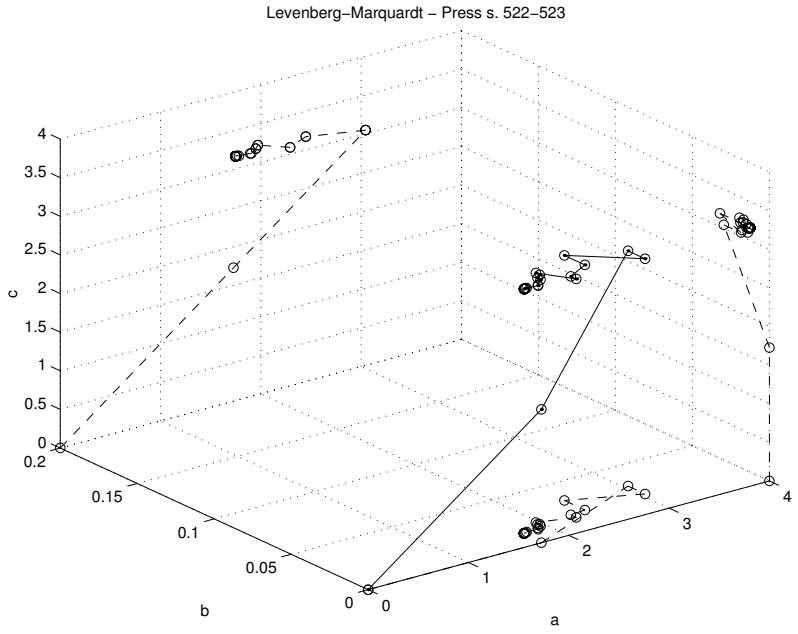
Kuva 10.2. Mallidatan sovitus MLM-menetelmällä.

Vertaa kuvaa 10.2 kuvaan 10.1. Huomaat, että MLM toimi paremmin kuin DFP. Nyt voidaan testata vielä, kuinka yksinkertaistettu Hessin matriisin laskenta vaikuttaa hakuun. Jos koodissa levmar2.m funktiokutsu dfp5_H korvataan kutsulla dfp5_H1, saadaan seuraava tuloste[21].

```
» levmar3

Ratkaisu = 1.7451      0.012457        3.1632
kohdefunktion arvo = 0.14155
gradienttivektori = 1.0111e-008 -3.1295e-008 5.1721e-008
Hessin matriisin approksimaatio =
   1.0e+004 *
     0.0003    -0.0099     0.0004
    -0.0099     1.5613    -0.0434
     0.0004    -0.0434     0.0026
Hakukierrokset = 26
Kaikki kierrokset = 46
```

[21] Tässä ei ole esitetty koodia!

ja seuraava kuva



Kuva 10.3. Mallidatan sovitus yksinkertaistetulla MLM-menetelmällä.

Verrataan kuvia 10.3 ja 10.2 huomataan, että MLM-päivitysrutiinin yksinkertaistaminen heikentää hakua jonkin verran. Kuitenkin myös kuvan 10.3 tapauksessa päädytään optimiin. Mallidatan sovitus voidaan myös tehdä käyttämällä Matlabin fmins-funktiota ja sijoittamalla sovitettava data omaan tiedostoonsa - tämä toteutus on esitetty liitteessä 2.

11. Lähdeluettelo

1. Bird R B, Stewart W E & Lightfoot E N (1960), *Transport Phenomena*. John Wiley & Sons 1960. 780 s.
2. Bishop R H (1997), *Modern Control Systems Analysis and Design Using Matlab and Simulink*. Addison-Wesley 1997. ISBN 0-201-49846-4. 251 s.
3. Carhahan B (ed.) (1996), *Computers in Chemical Engineering*. Cache Corp. 1996. ISBN 0-9655891. 287 s.
4. Davidon W C (1959) *Variable Metric Method for Minimization*. AEC Res. Develop. Rep., ANL-599. 1959.
5. Edgar T F & Himmelblau D M (1988), *Optimization of Chemical Processes*. McGraw-Hill 1988. ISBN 0-07-018991-9. 652 s.
6. Fletcher R & Powell M J D (1963) *A Rapidly Convergent Descent Method for Minimization*. Computer J., 6, 163 (1963).
7. Gill P E, Murray W & Wright M H (1981), *Practical Optimization*. Academic Press 1981. 12. painos 1999. ISBN 0-12-283952-8. 401 s.
8. Jaako J (1997), *Laskuharjoitusmoniste kurssiin 47434S Prosessien optimointi*. Moniste 50, Prosessitekniikan osasto, Oulun yliopisto, 1997. ISBN 951-42-4662-4. 117 s.
9. Jaako J (1998), *Nopeusyhtälön parametrien sovittaminen*. Säätötekniikan laboratorio, Oulun yliopisto. Raportti B No 6, 1998. ISBN 951-42-4961-5. 25 s.
10. James G, Burley D, Dyke P, Searl J, Steele N & Wright J (1993), *Advanced Modern Engineering Mathematics*. Wokingham: Addison-Wesley 1993. ISBN 0-201-56519-6. 901 s.
11. Kivelä S K (1980), *Matriisilasku ja lineaarialgebra*. Otakustantamo 1980. ISBN 951-671-266-5. 268 s.
12. Kreyszig E (1993), *Advanced Engineering Mathematics*. 7th edition. John Wiley & Sons 1993. ISBN 0-471-59989-1. 1271 s.
13. Laine J (1997), *Prosessilaskennan matemaattiset apuneuvot*. 1997. ISBN 952-90-8589-3. 254 s.
14. Laininen P (1980), *Todennäköisyyslasku ja tilastomatematiikka*. Otakustantamo 1980. ISBN 951-671-260-6. 254 s.
15. Levenspiel O (1972), *Chemical Reaction Engineering*. John Wiley & Sons 1972. ISBN 0-471-53019-0.
16. Ogata K (1997), *Modern Control Engineering*. Prentice-Hall 1997. ISBN 0-13-261389-1.
17. Press W H, Flannery B P, Teukolsky S A & Vetterling W T (1986), *Numerical Recipes - The Art of Scientific Computing*. Cambridge University Press, UK, 1986. ISBN 0-521-30811-9. 818 s.
18. Ray W H & Szekely J (1973), *Process Optimization with Applications in Metallurgy and Chemical Engineering*. John Wiley & Sons 1973. ISBN 0-471-71070-9. 371 s.
19. Reklaitis G V, Ravindran A & Ragsdell K M (1983), *Engineering Optimization - Methods and Applications*. John Wiley & Sons 1983. ISBN 0-471-05579-4. 684 s.
20. *Using MATLAB Graphics*, Version 5.1. The Mathworks, Inc., 1997. Toimitetaan MATLAB-ohjelmistopaketin mukana tiedostona `Graphg.pdf`. Tiedoston katseluun ja tulostamiseen tarvitaan Adobe Acrobat Reader.

Liite 1 - Algoritmeja

Marquardt's Compromise (Reklaitis *et al.* 1983, 106)

Step 1	Define $x^{(0)}$ = initial estimate of x*, M = maximum number of iterations allowed, ε = convergence criterion
Step 2	Set $k = 0$, $\lambda^{(0)} = 10^4$.
Step 3	Calculate $\nabla f\left(x^{(k)}\right)$.
Step 4	Is $\left\|\nabla f\left(x^{(k)}\right)\right\| < \varepsilon$? Yes: Go to step 11. No: Continue.
Step 5	Is $k \geq M$? Yes: Go to step 11. No: Continue.
Step 6	Calculate $s\left(x^{(k)}\right) = -\left[\mathrm{H}^{(k)} + \lambda^{(k)}\mathrm{I}\right]^{-1} \nabla f\left(x^{(k)}\right)$
Step 7	Set $x^{(k+1)} = x^{(k)} + s\left(x^{(k)}\right)$.
Step 8	Is $f\left(x^{(k+1)}\right) < f\left(x^{(k)}\right)$. Yes: Go to step 9. No: Go to step 10.
Step 9	Set $\lambda^{(k+1)} = \frac{1}{2}\lambda^{(k)}$ and $k = k+1$. Go to step 3.
Step 10	Set $\lambda^{(k)} = 2\,\lambda^{(k)}$. Go to step 6.
Step 11	Print results and stop.

A modified Marquardt method (Edgar & Himmelblau 1988, 214)

Step 1	Pick x^0 the starting point, *Let* ε = convergence criterion
Step 2	Set $k = 0$. Let $\beta^0 = 10^3$.
Step 3	Calculate $\nabla f\left(\mathrm{x}^k\right)$.
Step 4	Is $\left\|\nabla f\left(\mathrm{x}^k\right)\right\| < \varepsilon$? If yes, terminate. If no, continue.
Step 5	Calculate $\mathrm{s}^k = -\left[\mathrm{H}^k + \beta^k \mathrm{I}\right]^{-1} \nabla f\left(\mathrm{x}^k\right)$
Step 6	Calculate $\mathrm{x}^{k+1} = \mathrm{x}^k + \lambda^k \mathrm{s}^k$.
Step 7	Is $f\left(\mathrm{x}^{k+1}\right) < f\left(\mathrm{x}^k\right)$? If yes, go to step 8. If no, go to step 9.
Step 8	Set $\beta^{k+1} = \frac{1}{4}\beta^k$ and $k = k+1$. Go to step 3.
Step 9	Set $\beta^k = 2\,\beta^k$. Go to step 5.

Liite 2 - Matlabin fmins-funktion käyttö

fmins-funktio etsii monimuuttujaisen funktion minimin rajoittamattomassa tapauksessa. Käydään ensin yksinkertaisen esimerkin avulla läpi funktion käyttö.

* * *

Ratkaistaan

$$\min\left[f(x_1, x_2) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2 \right]$$

Haun aloituspisteenä on [-1.2 1]. Kirjoitetaan m-tiedosto (eli erillinen tiedosto ohjelmaeditorilla)

```
% fun.m

function f = fun(x)
f = 100*(x(2)-x(1)^2)^2 + (1-x(1))^2 ;
```

Optimi haetaan käskyin

```
» x = [ -1, 1 ] ;
» x = fmins('fun', x)
```

Vastaukseksi saadaan

```
x = 1.0000   1.0000
```

* * *

Ratkaistaan nyt mallidatan sovitustehtävä fmins-funktion avulla. Nyt data kannattaa tallettaa esilliseksi tiedostoksi; saadaan seuraavaa[22]

```
% ajoke.m
% (c) Juha Jaako 13.11.1998

T = [ 0      50     100    150    200    250    300 ...
      350    400    450    500    550    600    ] ;

DO = [ 5.0000 3.8700 3.6535 3.5365 3.4495 3.3520 3.2835 ...
       3.2215 3.1790 3.1250 3.0970 3.0700 3.0470 ] ;

save c:\data\yy\tehtke1 T DO ;      % Talletetaan työtilaa!
```

Huomaa ohjelmassa oleva save-komento, jonka avulla voidaan Matlabin työtilassa olevat matriisit tallettaa levylle. Nämä levylle talletetut matriisit voidaan edelleen ottaa käyttöön jossakin muussa Matlab-ohjelmassa käyttämällä load-komentoa; tätä komentoa käytetään seuraavaksi esiteltävässä koodissa (ajoke1.m). Huomaa myös, että data välitetään kutsuttaviin funktioihin parametrilistan avulla.

---

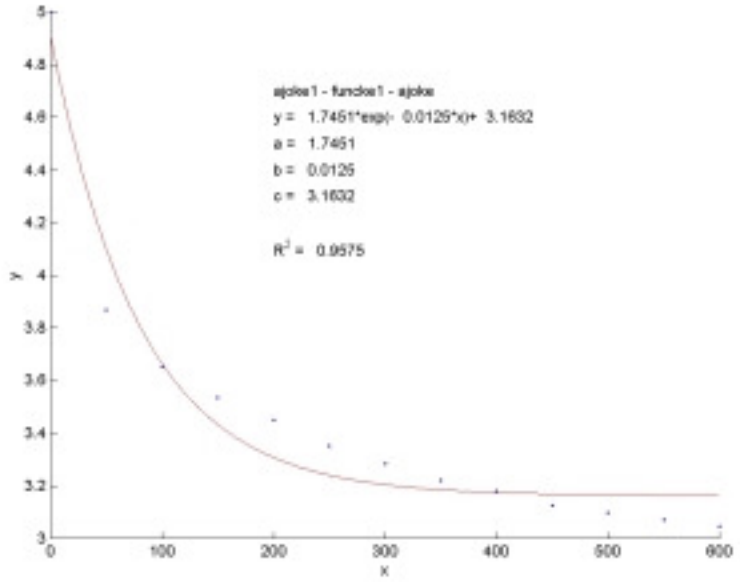
[22] Esimerkki esitetty alunperin kurssilla *Prosessien optimointi* 23.11.1998.

```
% ajoke1.m
% (c) Juha Jaako 19.11.1998

clear all ;  % HUOM!
fstr = '%8.4f' ;

% Ladataan data.
str = 'c:\data\yy\tehtke1' ;
load (str) ;                    % Luetaan talletettu työtila!
N = size (T, 2) ;               % Määritetään matriisin koko.

% Optiot-vektori.
options = zeros (1, 14) ;
options(1) = 0 ;
tol = 1.0e-8 ;
options(2) = tol ;
options(3) = tol ;
options(14) = 500 ;

% Haun alkuvektori.
x = [ 2  0.017  3 ] ;

% Kohdefunktion minimin haku.
xx = fmins ('funcke1', x, options, [], T, DO) ;

% Mallin antamat pisteet.
DOkalc = zeros (1,N) ;
a = xx(1)
b = xx(2)
c = xx(3)
str1 = sprintf (fstr, a) ;
str2 = sprintf (fstr, b) ;
str3 = sprintf (fstr, c) ;

% a*exp(-b*T)+c
for i = 1 : N ,
  DOkalc(i) = a*exp(-b*T(i))+c ;
end

% Viivan piirto.
vT = [ 0 : max(T)-1 ] ;
for i = 1 : max(T) ,
  v(i) = a*exp(-b*vT(i))+c ;
end

% Laskettu selitysaste.
R2 = seliaste (DO, DOkalc, N)
str4 = sprintf (fstr, R2) ;

% Kuva.
hold on ;
plot (T, DO, 'b.') ;
plot (vT, v, 'r-') ;
xlabel ('x') ; ylabel ('y') ;

pos = 200 ; lk = 4.8 ; skl = 0.1 ;

text (pos, lk-1*skl, 'ajoke1 - funcke1 - ajoke') ;
text (pos, lk-2*skl, ['y = ' str1 '*exp(-' str2 '*x)+' str3] ) ;
text (pos, lk-3*skl, ['a = ' str1] ) ;
text (pos, lk-4*skl, ['b = ' str2] ) ;
text (pos, lk-5*skl, ['c = ' str3] ) ;
text (pos, lk-7*skl, ['R^2 = ' str4] ) ;
hold off ;

print -djpeg100 c:\temp\ajoke ;
```

Ohjelma tulostaa tekstiä

```
» ajoke1
a =      1.7451
b =      0.0125
c =      3.1632
R2 =     0.9575
```

ja kuvan (JPEG-kuva!). Huomaa, miten ohjelma sijoittaa tekstit kuvaan.



Kuva L.2.1. Kohdefunktion minimointi

Laskentaa varten tarvitaan myös kohdefunktion arvot laskeva funktio (funcke1.m).

```
% funcke1.m
% (c) Juha Jaako 13.11.1998

function f = funcke1 (x, T, DO)

N = size (T, 2) ;

a = x(1) ; b = x(2) ; c = x(3) ;

% Lasketaan kohdefunktion arvo.
% a*exp(-b*T)+c
f = 0 ;
for i = 1 : N ,
  f = f + ( DO(i)-a*exp(-b*T(i))-c )^2 ;
end
```

Liite 3 - Matlabin fmin-funktion käyttö

fmin etsii yhden muuttujan funktion minimin annetulta välitä; kaavamuodossa $\min[f(x)]$ siten, että $x_1 < x < x_2$

Sinifunktion minimi valillä [0, 2π] saadaan seuraavasti

```
» a = fmin ('sin', 0, 2*pi)
a = 4.7124
```

Etsitään edelleen funktion $f(x) = (x-3)^2 - 1$ minimi väliltä [0, 5]. Kirjoitetaan m-tiedosto

```
% fun.m

function f = fun (x)
f = (x-3).^2 - 1 ;
```

Optimi haetaan käskyllä

```
» a = fmin ('fun', 0, 5)
```

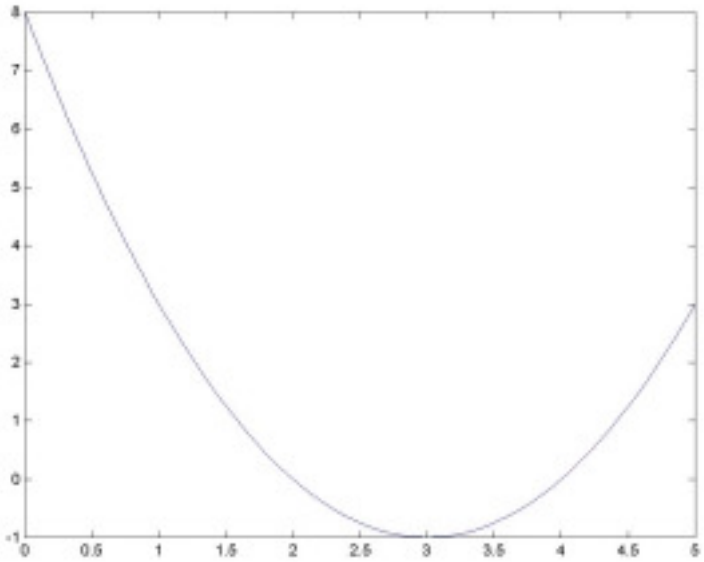
Vastaukseksi saadaan

```
a = 3
```

Funktion $f(x) = (x-3)^2 - 1$ kuvaaja saadaan piirrettyä seuraavalla käskyllä

```
» fplot('(x-3).^2 - 1', [0 5]);
```

joka tuottaa kuvan



Liite 4 - Luettelo ohjelmista

Tässä kirjassa olevien ohjelmien nimet ovat seuraavassa

ajoke.m	ajoke1.m	Bfh2Wm.m	BfhF2WmK.m
dfp4.m	dfp4_d.m	dfp4_f.m	dfp4_m.m
dfp5_a.m	dfp5_d.m	dfp5_f.m	dfp5_H.m
dfp5_H1.m	dfp5_m.m	dfp5a.m	EHals.m
EHals1.m	F2K.m	funcke1.m	koe21.m
laine187.m	laink187.m	levmar.m	levmar_h.m
levmar1.m	levmar2.m	piirto1.m	recycle.m
recycle1.m	seliaste.m	simpint.m	simpson.m
simpson1.m	temppro1.m	threwall.m	

Ohjelmakoodi on saatavissa osoitteesta

ht t p: / / nt sat . oul u. f i / j aako/ koodi j pj . zi p

Ohjelmat eivät kuitenkaan välttämättä ole samassa muodossa kuin tässä kirjassa esitetyt.

ISBN 951-42-5354-X
ISSN 1238-9404
Oulun yliopisto
Säätötekniikan laboratorio - Sarja B
Toimittaja: Leena Yliniemi

1. Yliniemi L & Koskinen J, Rumpukuivaimen sumea säätö. Joulukuu 1995. 17 s. 6 liitettä. ISBN 951-42-4301-3.
2. Leiviskä K, Rauma T, Ahola T, Juuso E, Myllyneva J & Alahuhta P, Sumea mallintaminen, viritys ja säätö. Tammikuu 1996. 44 s. 951-42-4348-X.
3. Altavilla M, Koskinen J & Yliniemi L, Rumpukuivaimen säätö neuroverkolla. Tammikuu 1996. 12 s. ISBN 951-42-4373-0.
4. Myllyneva J, Leiviskä K, Heikkinen M, Kortelainen J & Komulainen K, Sumean säädön käyttömahdollisuudet hiertämön ohjauksessa. Huhtikuu 1997. 52 s. ISBN 951-42-4647-0.
5. Leiviskä K & Heikkinen M, TMP-prosessin mallintaminen ja mallipohjainen säätö. Huhtikuu 1997. 68 s. ISBN 951-42-4646-2.
6. Jaako J, Nopeusyhtälön parametrien sovittaminen. Huhtikuu 1998. 25 s. ISBN 951-42-4961-5.
7. Myllyneva J, Kortelainen J, Latva-Käyrä K, Nystedt H & Leiviskä K, Hiertämön laatusaadöt. Syyskuu 1998. ISBN 951-42-5023-0.
8. Lähteenmäki M & Leiviskä K, Tilastollinen prosessinohjaus: perusteet ja menetelmät. Lokakuu 1998. ISBN 951-42-5064-8.
9. Tervahartiala P & Leiviskä K, Tilastollinen prosessinohjaus: ohjelmistovertailu. Elokuu 1999. ISBN 951-42-5343-4.
10. Jaako J, Eräitä optimointitehtäviä. Syyskuu 1999. 39 s. ISBN 951-42-5352-3.
11. Jaako J, Yksinkertaisia prosessimalleja. Syyskuu 1999. 73s. ISBN 951-42-5353-1.
12. Jaako J, MATLAB-ohjelman käyttö eräissä prosessiteknisissä laskuissa. Syyskuu 1999. 61 s. ISBN 951-42-5354-X.

Oulun yliopistopaino
Oulu 1999