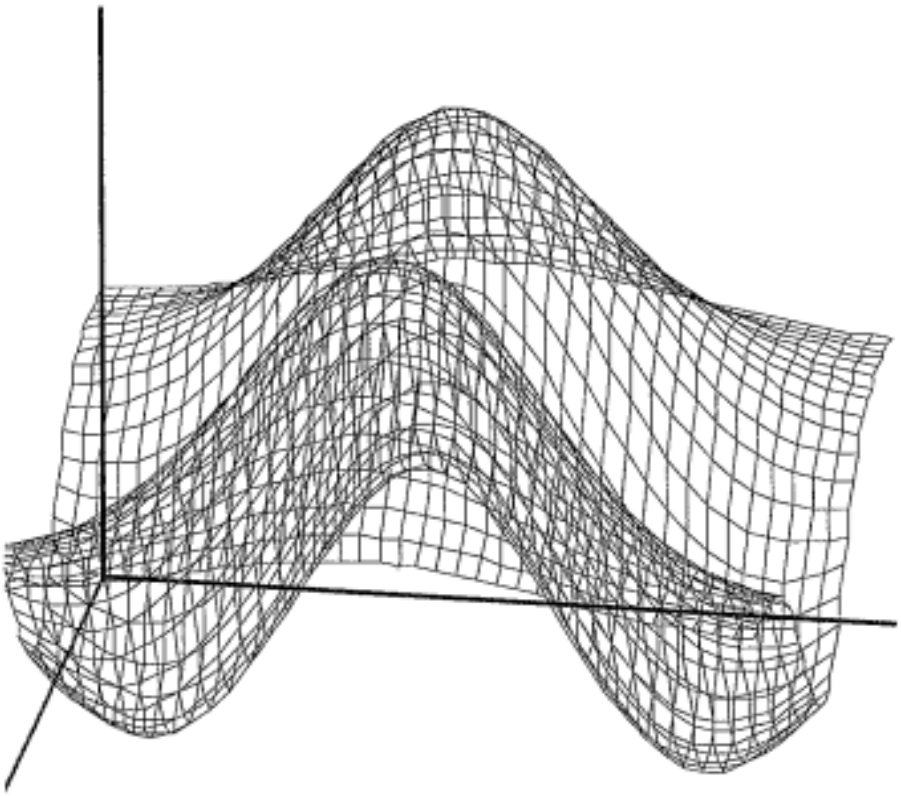
MATLAB

Usean muuttujan funktiot

Timo Mäkelä



Tammertekniikka

12. USEAN MUUTTUJAN FUNKTIOT

12.1 Vektorikentät

Vektoriarvoista funktiota sanotaan **vektorikentäksi**. Esimerkiksi tason vektorikenttä on vektoriarvoinen kahden muuttujan funktio

$$\bar{F}(x,y) = f_1(x,y)\bar{i} + f_2(x,y)\bar{j}.$$

Tason vektorikenttä voidaan havainnollistaa **nuolikuviolla**, joka saadaan aikaiseksi komennolla

quiver(X,Y,U,V),

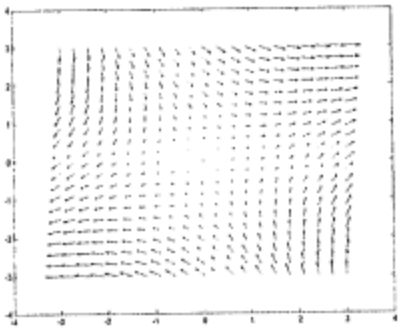
missä **X**, **Y**, **U** ja **V** ovat saman kokoisia matriiseja, joilla on seuraavat merkitykset:
(X_{ij}, Y_{ij}) ilmoittaa tason pisteen, johon vektori $U_{ij}\bar{i} + V_{ij}\bar{j}$ piirretään. MATLAB skaalaa piirrettävät vektorit sopivasti.

Esim. Piirretään vektorikenttä

$$\bar{F}(x,y) = (x+y)\bar{i} + (x-y)\bar{j}$$

välillä $-3 \le x \le 3$, $-3 \le y \le 3$.

```
>> [X,Y] = meshgrid(-3:.3:3);
>> U = X+Y;
>> V = X-Y;
>> quiver(X,Y,U,V)
```



12.2 Gradientti

Derivoituvan kahden muuttujan funktion $f(x,y)$ **gradientilla** tarkoitetaan tasovektoria

$$\nabla f(x,y) = \frac{\partial}{\partial x} f(x,y)\bar{i} + \frac{\partial}{\partial y} f(x,y)\bar{j},$$

missä

$$\frac{\partial}{\partial x} f(x,y) \text{ ja } \frac{\partial}{\partial y} f(x,y)$$

ovat funktion f osittaisderivaattoja x:n ja y:n suhteen. Gradientti on siis tason vektorikenttä.
Kolmen muuttujan funktion $f(x,y,z)$ gradientti määritellään vastaavasti

$$\nabla f(x,y) = \frac{\partial}{\partial x} f(x,y)\bar{i} + \frac{\partial}{\partial y} f(x,y)\bar{j} + \frac{\partial}{\partial z} f(x,y)\bar{k}.$$

Tämä yleistyy useamman muuttujan funktioihin.
Pisteessä p funktion arvot

- kasvavat nopeimmin gradientin suuntaan $\nabla f(p)$ ja
- pienenevät nopeimmin gradientille vastakkaiseen suuntaan $-\nabla f(p)$.

Kahden muuttujan funktion $f(x,y)$ gradientin laskemiseksi on ensin muodostettava matriisi

$$F = \begin{bmatrix} f(x,y) & f(x+h_1,y) & \cdots & f(x+nh_1,y) \\ f(x,y+h_2) & f(x+h_1,y+h_2) & \cdots & f(x+nh_1,y+h_2) \\ \vdots & \vdots & \cdots & \vdots \\ f(x,y+mh_2) & f(x+h_1,y+mh_2) & \cdots & f(x+nh_1,y+mh_2) \end{bmatrix}.$$

Gradientti muodostetaan nyt komennolla

[Fx,Fy] = gradient(F,h1,h2),

missä

- **F** on yo. muotoa oleva matriisi
- **h1** on muuttujan x muutos siirryttäessä sarakkeelta toiselle
- **h2** on muuttujan y muutos siirryttäessä riviltä toiselle

Tuloksena saadaan matriisit

- **Fx**, jossa on likiarvot osittaisderivaatoilla $\frac{\partial}{\partial x} f(x,y)$
- **Fy**, jossa on likiarvot osittaisderivaatoilla $\frac{\partial}{\partial y} f(x,y)$

Matriisit **Fx** ja **Fy** ovat samaa kertalukua kuin matriisi **F**. Matriisin **Fx** alkiot ovat seuraavan matriisin likiarvot (vastaavasti **Fy**):

$$Fx \approx \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) & \frac{\partial}{\partial x} f(x+h_1,y) & \cdots & \frac{\partial}{\partial x} f(x+nh_1,y) \\ \frac{\partial}{\partial x} f(x,y+h_2) & \frac{\partial}{\partial x} f(x+h_1,y+h_2) & \cdots & \frac{\partial}{\partial x} f(x+nh_1,y+h_2) \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial}{\partial x} f(x,y+mh_2) & \frac{\partial}{\partial x} f(x+h_1,y+mh_2) & \cdots & \frac{\partial}{\partial x} f(x+nh_1,y+mh_2) \end{bmatrix}.$$

Komennolla on lyhyemmät muodot:

- Komento

  [Fx,Fy] = gradient(F,h)

  tarkoittaa komentoa **[Fx,Fy] = gradient(F,h,h)**

- Komento

  [Fx,Fy] = gradient(F)

  tarkoittaa komentoa **[Fx,Fy] = gradient(F,1,1)**

Tämä yleistyy kolmen ja useamman muuttujan funktioihin. Tällöin lähtökohtana on 3- tai useampiulotteinen taulukko **F**.

Mitä pienempiä parametrien **h1** ja **h2** itseisarvot ovat, sitä tarkempi likiarvo funktion gradientille saadaan.

Esim. Määritetään funktion $f(x,y) = xe^{-x^2-y^2}$ gradientti välillä $-2 \le x \le 2$, $-2 \le y \le 2$, askelpituus on 0,2.

```
>> [X,Y] = meshgrid(-2:.2:2);
>> Z = X.*exp(-X.^2 - Y.^2);
>> [DX,DY] = gradient(Z,.2);
```

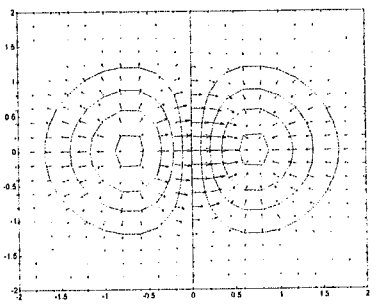
Piirretään funktion f tasa-arvokäyrästö

```
>> contour(X,Y,Z)
>> hold on
```

ja gradienttikenttä nuolikuviona

```
>> quiver(X,Y,DX,DY)
>> hold off
```

Kuvasta nähdään, missä funktion kasvusuunta ja suhteellinen kasvunopeus eri pisteissä.



Kahden muuttujan funktion gradientti voidaan laskea myös seuraavalla ohjelmalla

```
function [d1,d2] = grad(fun,x,y,h)
d1 = (feval(fun,x+h,y)-feval(fun,x-h,y))./(2*h);
d2 = (feval(fun,x,y+h)-feval(fun,x,y-h))./(2*h);
```

Ohjelman parametrit ovat

- **fun** otin derivoitavaan funktioon
- **x** ja **y** määrittävät pisteen **(x,y)**, jossa gradientti lasketaan
- **h** on askelpituus

Esim. Määritetään funktion $f(x,y) = xe^{-x^2-y^2}$ gradientti pisteessä (0,5; 1):

```
>> [dx,dy] = grad(inline('x .* exp(-x.^2 - y.^2)'),0.5,1,0.01)
dx =
    0.1433
dy =
   -0.2865
```

Siis gradientti on vektori

$$0{,}1433\vec{i} - 0{,}2865\vec{j}$$

12.3 Minimointi ja maksimointi

12.3.1 Perusmuoto

Usean muuttujan funktion[3] **f(x)** *lokaali minimikohta* voidaan määrittää komennolla

x = fminsearch(@fun, x0),

missä

- **@fun** on *otin* tiedostossa **fum.m** sijaitsevaan funktioon. Funktio annetaan muodossa

  function y = fun(x)

  ...

  y = ... % Funktion f(x) arvo pisteessä x.

  missä **x** on vektori.
- **x0** on vektori, joka sisältää minikohdan alkuarvauksen

Tuloksena saadaan

- vektori **x**, joka sisältää minimikohdan likiarvon.

Jos funktio **f(x)** ei sijaitse tiedostossa lokaali minimikohta voidaan määrittää komennolla

x = fminsearch(inline('f(x)'), x0),

jossa käytetään *inline-objektia*.

Jos komento annetaan muodossa

[x,fval] = fminsearch(...),

[3] Argumentti **x** on vektori.

niin tulokseksi saadaan minimikohdan likiarvon **x** lisäksi minimoitavan funktion arvo **fval** kohdassa **x**.

Funktion **fminsearch** käyttää minimin etsimiseen simpleksi-hakumenetetelmää. Tämä on suora menetelmä, joka ei käytä hyväksi gradienttia. Menetelmä lähtee liikkeelle annetusta alkuarvauksesta.

Esim. Edellisen kappaleen esimerkin kuvasta havaitaan, että funktiolla $f(x,y) = xe^{-x^2-y^2}$ on lokaali minikohta lähellä pistettä (0,7; 0). Lasketaan tämä tarkemmin.

Otinta käyttäen:

m-tiedosto fun.m:

```
function y = fun(x);
y = x(1)*exp(-x(1)^2-x(2)^2);
```

komento:

```
>> [x,arvo] = fminsearch(@fun,[-0.7,0])
x =
   -0.7071    0.0000
arvo =
   -0.4289
```

Inline-objektia käyttäen:

```
>> [x, arvo] = fminsearch(inline('x(1)*exp(-x(1)^2-x(2)^2)'),[-0.7,0])
x =
   -0.7071    0.0000
arvo =
   -0.4289
```

Molemmissa tapauksessa saadaan minikohdaksi (–0,7071;0) ja minimiarvoksi –0,4289.

Koska

funktion **f(x)** *lokaali maksimikohta on funktion* **–f(x)** *lokaali minimikohta,*

voidaan *lokaali maksimikohta* määritetään seuraavasti:

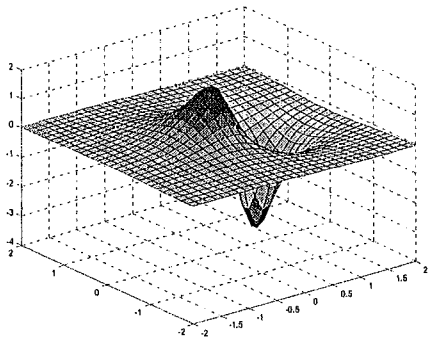
x = fminsearch(inline('-fun(x)'), x0),

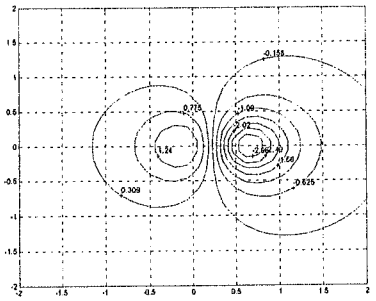
Esim. Määritetään funktion $f(x,y) = \frac{1}{0{,}3+x^2+y^2} - \frac{1}{0{,}2+(x-0{,}6)^2+y^2}$ mahdolliset lokaalit maksimi ja minikohdat.

Ratkaisu: Piirretään funktion $z = f(x,y)$ kuvaaja.

```
>> [X,Y] = meshgrid(linspace(-2,2,30));
>> Z =1./(0.3+X.^2+Y.^2)-1./(0.2+(X-0.6).^2+Y.^2);
>> figure(1)
>> surf(X,Y,Z);
>> figure(2)
>> C = contour(X,Y,Z,10); clabel(C);
```

Kuvista nähdään, että funktiolla on lokaali minimi pisteessä (0,7;0) ja lokaali maksimi pisteessä (–0,2;0). Määritetään pisteet tarkemmin. Lasketaan myös arvot näissä pisteissä.





Muodostetaan m-tiedosto fun.m:

```
function z = fun(v);
x = v(1); y = v(2);
z = 1/(0.3+x^2+y^2)-1/(0.2+(x-0.6)^2+y^2);
```

Lokaali minimi:

```
>> [x,f] = fminsearch(@fun,[0.7,0])
x =
    0.6510    0.0000
f =
   -3.5542
```

Lokaali maksimi:

```
>> [x,f] = fminsearch(inline('-fun(x)'),[-0.2,0])
x =
   -0.1357    0.0000
f =
   -1.7915
>> -f
ans =
    1.7915
```

Siis funktiolla on lokaali minimi –3,5542 pisteessä (0,6510; 0) ja lokaali maksimi 1,7915 pisteessä (–0,1357; 0).

12.3.2 Lisäparametrit

Jos lokaalin minimikohdan määrittämisessä on ongelmia, voidaan turvautua komennon laajempaan muotoon joka on seuraava:

[x,fval,exitflag,output] = fminsearch(fun, x0,asetukset),

missä **x**, **flag**, **fun** ja **x0** ovat kuten edellä ja

- **exitflag** kuvaa numeerisen menetelmän onnistumista:
  - **>0**: numeerinen menetelmä konvergoi ratkaisuun **x**
  - **0**: iteraatioiden maksimimäärä saavutettu
  - **<0**: numeerinen menetelmä ei kovergoinut.
- **output** tietuemuuttuja[4], joka kuvaa minimointia:
  - **output.algorithm**: käytetty menetelmä

[4] Tietueita käsitellään ohjelmoinnin yhteydessä.

- **output.funcCount**: funktiokutsujen lukumäärä
- **output.iterations**: iteraatioiden määrä.

- **asetukset** joilla voidaan vaikuttaa minimointiin. Tämä on tietuemuuttuja sisältää mm. seuraavat kentät:
  - **TolX**: iteraation lopetustoleranssi (oletus. 1e-4)
  - **MaxIter**: iteraatiokierrosten maksimimäärä
  - **MaxFunEvals**: funktion laskentakertojen maksimimäärä.
  - **Display**: välitulosten esittäminen:
    - 'off': ei esitetä
    - 'final': vain lopputulos
    - 'iter': jokaisen iteraation tulos esitetään
    - 'notify': (oletusarvo) tulos esitetään vain, jos menetelmä ei kovergoi

Asetukset asetetaan voimaan **optimset**-funktiolla seuraavasti:

asetukset = optimset('asetus1', arvo1, 'asetus2', arvo2, ...)

Asetuksia muutetaan komennolla

asetukset = optimset(vanha_asetus, 'asetus1', arvo1, 'asetus2', arvo2, ...)

Asetus voidaan tulostaa **optimget**-funktiolla:

optimget(asetukset, 'nimi')

Esim. Tarkastellaan aiemman esimerkin funktion $f(x,y) = \frac{1}{0{,}3+x^2+y^2} - \frac{1}{0{,}2+(x-0{,}6)^2+y^2}$ lokaalin minikohdan saavuttamista lähtien alkuarvauksesta 0,7. Tulostetaan jokaisen iteraation tulos.

Asetetaan asetus:

```
>> asetukset = optimset('Display', 'iter')
asetukset =
       ActiveConstrTol: []
       DerivativeCheck: []
           Diagnostics: []
         DiffMaxChange: []
         DiffMinChange: []
               Display: 'iter'
     GoalsExactAchieve: []
...
```

Määritetään lokaali minimikohta:

```
>> [x,f] = fminsearch(@fun,[0.7,0],asetukset)
 Iteration   Func-count     min f(x)         Procedure
     1           3         -3.49608         initial
     2           5         -3.54926         reflect
     3           7          -3.5539         expand
     4           9          -3.5539         contract outside
     5          11          -3.5539         contract inside
     6          13          -3.5539         contract inside
     7          15         -3.55405         contract inside
     8          17          -3.5542         contract inside
     9          18          -3.5542         reflect
    10          20          -3.5542         contract inside
    11          22          -3.5542         contract outside
```

```
    12          24          -3.55421        contract inside
    13          25          -3.55421        reflect
    14          27          -3.55421        contract inside
    15          29          -3.55421        contract inside
    16          31          -3.55421        contract outside
    17          33          -3.55421        reflect
    18          35          -3.55421        contract inside
    19          37          -3.55421        contract inside

Optimization terminated successfully:
 the current x satisfies the termination criteria using OPTIONS.TolX of
1.000000e-004
 and F(X) satisfies the convergence criteria using OPTIONS.TolFun of
1.000000e-004

x =
    0.6510    0.0000

f =
   -3.5542
```

Esim. Etsitään funktion $f(x,y) = \frac{1}{0{,}3+x^2+y^2} - \frac{1}{0{,}2+(x-0{,}6)^2+y^2}$ lokaalia minikohtaa lähtien alkuarvauksesta –0,7.

```
>> [x,fval,exitflag,output] = fminsearch(@fun,[-0.7,0])

Exiting: Maximum number of function evaluations has been exceeded
         - increase MaxFunEvals option.
         Current function value: 0.000000

x =
  1.0e+016 *
   -1.1426    0.0015

fval =
     0

exitflag =
     0

output =
    iterations: 141
     funcCount: 403
     algorithm: 'Nelder-Mead simplex direct search'
```

Lokaalia minikohtaa ei varmaankaan löydetty.

12.4 Tasointegraali

Tasointegraalin

$$\int_{ymin}^{ymax}\left(\int_{xmin}^{xmax} f(x,y)dx\right)dy$$

lasketaan komennolla

q = dblquad(fun,xmin,xmax,ymin,ymax)

missä

- fun = @f, jos funktio on tiedostossa f.m
- fun = inline('f(x,y)'), jos funktio ei ole tiedostossa.

Integraali lasketaan tarkkuudella 10^{-6}. Jos halutaan laskea muulla tarkkuudella käytetään komentoa

q = dblquad(fun,xmin,xmax,ymin,ymax,tol)

missä **tol** ilmoittaa toleranssin.

Integroinnin sisempi muuttuja määräytyy seuraavasti:

- m-funktiossa sisempänä integroidaan ensimmäisen kutsuargumentin suhteen.
- inline-objektissa sisempänä integroidaan aakkostossa lähempänä alkupäätä olevan muuttujan suhteen.

Inline-objektin integroimisjärjestyksen voi selvemmin ilmoittaa antamalla objekti muodossa

inline('f(x,y)','x','y').

Tässä x on sisempi integroimismuuttuja.

Esim. Lasketaan integraali $\int_{\pi}^{2\pi}\left(\int_{0}^{\pi}(y\cos x + x\sin y)dx\right)dy$

Otinta käyttäen:

m-tiedosto intgrd.m:

```
function z = intgrd(x,y)
z = y.*cos(x)+x.*sin(y);
```

komento:

```
>> Q = dblquad(@intgrd, 0, pi, pi, 2*pi)
Q =
   -9.8696
```

Inline-objektina:

```
>> Q = dblquad(inline('y*cos(x)+x*sin(y)'), 0, pi, pi, 2*pi)
Q =
   -9.8696
```

Sama toisin:

```
>> Q = dblquad(inline('y*cos(x)+x*sin(y)','x','y'), 0, pi, pi, 2*pi)
Q =
   -9.8696
```

Komennolla

```
>> Q = dblquad(inline('y*cos(x)+x*sin(y)','y','x'), 0, pi, pi, 2*pi)
Q =
   29.6088
```

lasketaan integraali $\int_{\pi}^{2\pi}\left(\int_{0}^{\pi}(y\cos x + x\sin y)dy\right)dx$.