Lyhyt Matlab-opas

Tarkoitettu Turun yliopiston insinöörimatematiikan kursseille

1 Asentaminen

Turun yliopiston hankkiman kampuslisenssin suomin oikeuksin opiskelijat voivat asentaa Matlab-ohjelmiston käytettäväksi omille koneilleen lataamalla ohjelman osoitteesta https://utushop.utu.fi/p/3645-matlab/. Asennukseen tarvitaan Turun yliopiston käyttäjätunnus ja salasana. Optionaalisista työkalupakeista valitse ainakin symbolisen laskennan työkalupakki asennuksen yhteydessä.

Matlabin etäkäyttö saattaa vaatia VPN-yhteyden Turun yliopiston verkkoon ja ohjelma tätä varten löytyy osoitteesta https://utushop.utu.fi/p/3715-anyconnet-vpn/.

Raportointi Matlabin asennukseen tai toiminnallisuuteen liittyvistä ongelmista tulee osoittaa Turun yliopiston IT-palveluille, esimerkiksi osoitteeseen helpdesk@utu.fi.

2 Perusteet

2.1 Matriisit

Matlabin keskeinen tietorakenne on matriisi. Matematiikan terminologian mukaisesti $m \times n$-matriisilla tarkoitetaan suorakulmaista kaaviota, johon $m \times n$ alkiota (yleensä lukuja) on asetettu m:ään riviin ja n:ään sarakkeeseen. Esimerkiksi

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 2 \\ 1 & -1 & 1 & -1 & 2 \\ 1 & -1 & -1 & -1 & 2 \end{pmatrix}$$

on 4×5-matriisi. $1 \times n$-matriisia, jossa on vain yksi n-pituinen rivi, kutsutaan *rivivektoriksi* ja $n \times 1$-matriisia, jossa on vain yksi sarake, kutsutaan *sarakevektoriksi*.

Matriisin ollessa keskeinen tietorakenne myös luvut Matlab käsittelee 1×1-matriiseina. Matriisit syötetään hakasulkeissa, alkiot erotetaan pilkuilla ja rivit puolipisteillä. Syöte

```
A=[1,1,1,1,1;1,2,3,4,2;1,-1,1,-1,2;1,-1,-1,-1,2]
```

siis määrittelee yllämainitun 4×5-matriisin A. Tärkeä syöttökomento on myös `x=[a:d:b]`, joka muodostaa rivivektorin eli $1 \times (k+1)$ -matriisin `x`, jonka alkiot ovat `a`, `a+d`, `a+2d`, `...`, `a+kd`, missä k on suurin luku, jolle $a + kd \leq b$. Esimerkiksi `x=[0:0.01:4]` muodostaa 1×401-matriisin (rivivektorin) `[0,0.01,0.02, ...,3.99,4]`. Jos `d` puuttuu, on oletusarvona `d=1`. Kun komennon perään kirjotetaan puolipiste (`x=[0:0.01:4];`) ei Matlab tulosta matriisia `x` syötteen antamisen jälkeen vaan otaksuu, että kyseessä on ohjelmaa varten annettu rivi. Matlab-ohjelmien rivit päättyvät aina puolipisteeseen.

Matriiseja voidaan yhdistää seuraavasti: Jos esimerkiksi `A=[1,1;0,1]` ja `B=[0,1;1,1]` (2×2-matriiseja), on `C=[A B]` 2×4-matriisi jossa A ja B on asetetty vierekkäin ja `D=[A;B]` 4×2-matriisi jossa A ja B asetetaan päällekkäin. Merkintä `A(i,j)` viittaa matriisin `A` alkioon rivillä i ja sarakkeella j. Sarakkeen j k ensimmäistä alkiota taas saadaan

syötteellä `A(1:k,j)` ja esimerkiksi rivivektorin `x` viisi ensimmäistä alkiota saadaan syötteellä `x(1:5)`. Mille hyvänsä matriisille A `size(A)` kertoo matriisin koon (tuloste on kaksi lukua 1×2-matriisissa). Komento `sum(A)` laskee matriisin `A` alkioiden summan.

Matriisien `A` ja `B` yhteenlasku lasketaan muodossa `A+B`, kertolasku `A*B`, ja matriisin `A` transpoosi saadaan syötteellä `A'`. Neliömatriisin determinantti saadaan komennolla `det(A)` ja kääntyvän matriisin käänteismatriisi komennolla `inv(A)`. Komento `rref(A)` muuntaa matriisin A redusoituun porrasmuotoon.

Komento `format rat` asettaa Matlabin laskemaan desimaaliesitysten sijaan murtoluvuilla. Pelkkä `format` kumoaa tämän. Matlabin varaama tila rationaalilukujen tallennukselle on kuitenkin hyvin rajallinen, ja yritys käyttää rationaalilukuja joissa olisi suuri nimittäjä johtaa melko nopeasti vääristymiin.
Esimerkiksi matriisit

$$A = \begin{pmatrix} 1 & 3 & -3 & -2 \\ 0 & 2 & 1 & -2 \\ -4 & 1 & 2 & -3 \end{pmatrix} \quad \text{ja} \quad B = \begin{pmatrix} 2 & 0 & -1 & -1 \\ 1 & 0 & 3 & 3 \\ -1 & 5 & 0 & -2 \end{pmatrix}$$

Syötetään seuraavasti:

```
>> A=[1,3,-3,-2;0,2,1,-2;-4,1,2,-3]

>> B=[2,0,-1,-1;1,0,3,3;-1,5,0,-2]
```

Matriisien summa lasketaan seuraavasti:

```
>> A+B

ans =
     3     3    -4    -3
     1     2     4     1
    -5     6     2    -5

>>
```

Matriisin A redusoitu porrasmuoto saadaan seuraavasti:

```
>> rref(A)

ans =

      1            0            0          -5/4
      0            1            0          13/4
      0            0            1          13/4

>>
```

Matlab-ohjelmistossa matriisikertolasku ilmaistaan asteriskilla `*` ja potenssi ylänuolella `^`

Huom: Matlabiin ohjelmoidut funktiot, kuten sin kohdistuvat matriisin jokaiseen alkioon erikseen. Esimerkiksi `x=[0:0.1:2*pi];` tuottaa 1×63-matriisin `x=[0,0.1000,0.2000,`

..., 6.1000,6.2000], ja samankokoinen on myös y=sin(x)=[0,0.0998,0.1987, ..., -0.1822,-0.0831].

Sen sijaan matriiseille määritellyt operaatiot kertolasku * ja potenssiin korotus ^ (haluttaessa syöttää ^ Windows-käyttöliittymässä pitää ^ näppäillä kaksi kertaa!) Matlab pyrkii toteuttamaan matriisikertolaskuna. Tämä voidaan kieltää kirjoittamalla .* ja .^ (huom piste). Esimerkiksi matriisin A toinen potenssi lasketaan seuraavasti: A^2, mutta matriisin A *alkioiden* toinen potenssi taas seuraavasti: A.^2

Esimerkki: Lasketaan yhteen luvut $(99/100)^{10}$, $(99/100)^{11}$, ... $(99/100)^{100}$. Tämä voidaan tehdä seuraavasti: n=[10:100], A=(99/100).^n ja sum(A).

2.2 Determinantti, ominaisarvot ja diagonalisointi

Matlabissa determinantti matriisille a saadaan komennolla det(a). Kääntyvälle matriisille a saadaan käänteismatriisi komennolla a^(-1) tai inv(a).

Komento eig(a) tuottaa vektorin, jossa ovat matriisin a ominaisarvot. Syötteellä [V,D]=eig(a) saadaan diagonaalimatriisi D, jonka diagonaalialkiot ovat matriisin a ominaisarvot ja matriisi V, jonka sarakkeina ovat matriisin a ominaisvektorit. Matriisin V sarakejärjestys on sama kuin matriisin D diagonaalialkioiden.

Esimerkki: Olkoon

$$a = \begin{pmatrix} 2 & 3 & -3 \\ 1 & 4 & -1 \\ 3 & 9 & -4 \end{pmatrix}.$$

Tälle ominaisarvot ja diagonalisaation välittävä matriisi saadaan seuraavasti:

```
>> a=[2,3,-3;1,4,-1;3,9,-4]

a =

     2     3    -3
     1     4    -1
     3     9    -4

>> [V,D]=eig(a)

V =

   -0.7071   -0.9487    0.5774
    0.0000    0.3162   -0.5774
   -0.7071   -0.0000   -0.5774


D =

   -1.0000         0         0
         0    1.0000         0
         0         0    2.0000
```

Tuloste pitää tulkita siten, että ominaisarvot ovat $-1, 1$ ja 2, sekä näihin liittyvät ominaisvektorit $(-0.7071, 0, -0.7071)^T$, $(-0.9487, 0.3162, 0)^T$ ja $(0.5774, -0.5774, -0.5774)^T$.

2.3 Tarkkojen arvojen käyttö

Yllämainittu diagonalisointi antaa vastauksen likiarvoina. Toisinaan on kuitenkin tarpeen saada sekä ominaisarvot että ominaisvektorit tarkkoina arvoina, ja Matlabin symbolisen laskennan työkalupaketti pystyy tähän ainakin jossain määrin.

Tarkkojen arvojen käyttö ominaisarvojen etsimisessä tapahtuu määrittelemällä matriisi symboliseksi:

```
>> a=sym([2,3,-3;1,4,-1;3,9,-4])

a =

[ 2, 3, -3]
[ 1, 4, -1]
[ 3, 9, -4]

>> [V,D]=eig(a)

V =

[ -3, -1, 1]
[  1,  1, 0]
[  0,  1, 1]


D =

[ 1, 0,  0]
[ 0, 2,  0]
[ 0, 0, -1]

>>
```

Tämän mukaan matriisista D luettavat ominaisarvot ovat 1, 2, ja -1 ja näihin liittyvät ominaisvektorit $(-3,1,0)^T$, $(-1,1,1)^T$ ja $(1,0,1)^T$ tässä järjestyksessä. Aiemman esimerkin mukaiset numeeriset ominaisvektorit eivät ole ristiriidassa tämän esimerkin kanssa, sillä kertomalla ominaisvektori jollakin vakiolla saadaan myös ominaisvektori. On myös huomattava, että ominaisarvojen järjestys on viimeisimmässä esimerkissä erilainen kuin aiemmassa.

2.4 Jordanin normaalimuoto

Matriisille `A` ja Jordanin normaalimuodon välittävä matriisi `J` saadaan kirjoittamalla `[P,J]=jordan(A)`.
Esimerkki. (`sym`-optio matriisille määrittää laskennan symboliseksi.)

```
>> A=sym([2,-2,4;1,5,-3;1,2,1])

A =

[ 2, -2,  4]
[ 1,  5, -3]
[ 1,  2,  1]
```

```
>> [P,J]=jordan(A)

P =

[  3,  2, -2]
[ -2, -1,  2]
[ -1,  0,  1]


J =

[ 2, 0, 0]
[ 0, 3, 1]
[ 0, 0, 3]

>>
```

2.5 Kuvaajan piirtäminen

plot Jos `x` ja `y` ovat samanpituisia rivivektoreita, piirtää `plot(x,y)` piirtää kuvaajan, jossa pisteet `(x(i),y(i))` kaikilla i:n mahdollisilla arvoilla ovat liitettyjä janalla toisiinsa. Paraabelin $y = x^2$ kuvaaja välillä $[-1, 3]$ saadaan esimerkiksi seuraavasti: `x=[-1:0.01:3]`, `y=x.^2`, `plot(x,y)`.

Matlabin `plot`-komento on luontevin parametriesityksen yhteydessä: Jos on piirrettävä käyrän $\{(x(t)), y(t) \mid t \in I\}$ kuvaaja, muodostetaan I riittävän tihein välein parametrin arvoja esittävään vaakavektoriin `t`, minkä jälkeen lasketaan `x(t)` ja `y(t)`. Lopuksi `plot(x,y)` tuottaa kuvaajan. Esimerkiksi `t=[0:0.01:2*pi]`, `x=cos(t)`, `y=sin(t)` ja `plot(x,y)` piirtää yksikköympyrän.

Huom: `plot`-komennossa on myös mahdollista, että vektorit `x` ja `y` ovat sarakevektoreita, tai että toinen on rivi- ja toinen sarakevektori. Lisäksi on mahdollista, että `x` on vektori, mutta `y` on matriisi. Esimerkiksi `x=[-3:0.01:3]`, `y=[2*x;x.^2]`, `plot(x,y)` piirtää kaksi kuvaajaa.

fplot: Symbolisen laskennan työkalun avulla kuvaaja voidaan piirtää myös pelkästään funktion lausekkeen perusteella, esimerkiksi:

```
>> syms x
>> fplot(sin(x),[-10,10])
```

Ylläolevista riveistä ensimmäinen määrittelee x:n muuttujasymboliksi.

2.6 Polynomit

Ilman symbolisen laskennan pakettia **Matlab**-ohjelmistossa polynomit esitetään jonona kertoimia muuttujan suurimmasta potenssista alaspäin. Esimerkiksi polynomit $p(x) = x^3 + 2x - 1$ ja $q(x) = x^2 - 1$ syötetään muodossa `p=[1,0,-2,1]` ja `q=[1,0,-1]`. Polynomim arvo lasketaan komennolla `polyval(p,x)`, missä `p` on polynomi ja x on piste, missä polynomin arvo lasketaan (voi myös olla rivivektori). Polynomien `p` ja `q` kertolasku lasketaan komennolla `conv(p,q)`.

```
>> p=[1,0,2,-1]

>> q=[1,0,-1]
```

Polynomin arvo jossakin pisteessä lasketaan komennolla `polyval`. Esimerkiksi $q(2) = 2^2 - 1 = 3$ lasketaan seuraavasti:

```
>> polyval(q,2)

ans =

     3

>>
```

Polynomien tulo lasketaan komennolla `conv`:

```
>> conv(p,q)

ans =

      1    0    1   -1   -2    1

>>
```

Vastaus tulkitaan polynomiksi $x^5 + x^3 - x^2 - 2x + 1$. Nollakohtien etsiminen puolestaan tapahtuu seuraavasti:

```
>> roots(p)

ans =

  -0.2267 + 1.4677i
  -0.2267 - 1.4677i
   0.4534 + 0.0000i

>>
```

Esimerkki: Polynomin $x^5 - 3x + 1$ nollakohdille saadaan likiarvot seuraavasti:

```
>> roots([1,0,0,0,-3,1])

ans =

  -1.3888
  -0.0803 + 1.3284i
  -0.0803 - 1.3284i
   1.2146
   0.3347

>>
```

Matlabissa on symbolisen matematiikan työkalut myös tekijöihinjakoa varten. Tälläin tulee luoda `syms`-komennolla tarvittavat muuttujasymbolit. Esimerkiksi

```
>> syms x
>> factor(x^6-1)

ans =

[x-1, x+1, x^2+x+1, x^2-x+1]

>>
```

Pelkkä polynomien jakolasku taas saadaan aikaan komennolla `deconv`. Jos esimerkiksi on jaettava polynomi $p(x) = x^3+2x-1$ polynomilla $q(x) = x^2-1$ toimitaan seuraavasti:

```
>> [a,r]=deconv(p,q)

a =
     1     0

r =

     0     0     3    -1

>>
```

Vastaus tulkitaan siten että osamäärä on x ja jakojäänös $3x - 1$.

Matlabissa on myös työkalu osamurtojen löytämiseksi:

```
>> [r,p,k]=residue(p,q)

r =

     2
     1

p =

    -1
     1

k=

     1     0

>>
```

tulkitaan siten, että

$$\frac{p(x)}{q(x)} = \frac{2}{x+1} + \frac{1}{x-1} + x.$$

Sarakevektori `r` ilmaisee siis osoittajat 2 ja 1, kun puolestaan sarakevektori `p` ilmaisee nimittäjien nollakohdat -1 ja 1. Nimittäjät ovat siis $x-(-1) = x+1$ ja $x-1$. k on osamääräpolynomin $1 \cdot x + 0$ esitys.

3 Paloittain määritellyt funktiot

Paloittain määritellyt funktiot kuten

$$f(x) = \begin{cases} x^2, & \text{jos } x < 0 \\ x, & \text{jos } x > 0 \end{cases}$$

Voidaan määritellä ja piirtää Matlabissa välillä $[-2, 2]$ seuraavasti:

```
syms x;
y = piecewise(x<0, x^2, x>0, x);
fplot(y,[-2,2]);
```

4 Kompleksiluvut

Kompleksiluvut annetaan **Matlab**ille seuraavan esimerkin mukaisesti:

```
>> (5+3i)/(1+2i)

ans =

    11/5      -    7/5i
```

(jos on annettu `format rat` aiemmin) Vastaus siis tulkitaan siten, että $\frac{5+3i}{1+2i} = \frac{11}{5} - \frac{7}{5}i$.

Kompleksikonjugaatin tuottaa `conj(z)` ja itseisarvon `abs(z)`. Vaihekulma puolestaan saadaan komennolla `angle(z)`. Reaali- ja imaginaariosat saadaan eroteltua komennoilla `real(z)` ja `imag(z)`.

Mieti miksi vastaus syötteeseen on alla olevan kaltainen:

```
>> imag(log(-1))

ans =

    3.1416

>>
```

5 Differentiaali- ja integraalilaskenta

Matlabiin on ohjelmoitu työkaluja myös raja-arvojen määrittämiseen. Tällöin tulee huolehtia siitä, että tarvittavat symbolit määritellään *syms*-komennolla

```
>> syms x
>> limit(sin(x)/x,x,0)

ans =

1
```

`limit`-komennossa ensimmäinen paikka on varattu funktiolle, toinen muuttujalle joka lähestyy kolmannessa paikassa ilmaistua arvoa. Raja-arvot äärettömyydessä ja toispuoleiset raja-arvot lasketaan seuraavien esimerkkien mukaan.

$$\lim_{x\to\infty} \frac{x^{100}}{e^x}:$$

```
>> limit(x^100/exp(x),x,inf)

ans =

0

>>
```

$$\lim_{x\to 2-} \frac{1}{x-2}:$$

```
>> limit(1/(x-2),x,2,'left')

ans =

-Inf

>>
```

Differentiaalilaskennan tehtäviä voi suorittaa **Matlab**-ohjelmistossa symbolisen matematiikan työkalujen avulla. Tällöin tulee `syms`-komennolla luoda tarvittavat muuttujasymbolit.

Esimerkki:

```
>> syms x y
>> f=sin(x*y^2)

f =

sin(x*y^2)

>> diff(f,x)

ans =

y^2*cos(x*y^2)

>>
```

Antiderivaatan etsimistä varten puolestaan on toiminto `int`:

```
>> int(f,x)

ans =

-cos(x*y^2)/y^2
```

Matlabin `x = fzero(fun,x0)` etsii funktion `fun` nollakohtaa annetun luvun `x0` lähistöltä.

Esimerkki:

```
>> x=fzero(@sin,1)

x =

  1.5485e-024

>>
```

Huomautus Edellisessä esimerkissä käytettiin Matlabin ns. *function handle*-tietotyyppiä (ei vakiintunutta suomennosta, sanotaan vaikka funktiokahva). Funktiokahvaa käytetään tavallisesti kun pitää jokin funktio sisällyttää johonkin Matlab-rutiiniin. Funktiokahva luodaan @-symbolilla, tästä nähdään esimerkkejä jäljempänä. Yllä oleva on esimerkki *nimetyn funktion kahvasta*, mikä viittaa siihen, että sini on kiinteä osa Matlabia.

Omat funktiot kirjoitetaan Matlabissa ns. m-tiedostoiksi (.m -pääte) (new $\rightarrow$ function $\rightarrow$). m-tiedosto voi olla funktion määrittelevä ohjelma kaikkine mahdollisine ohjelmointirakenteineen. Tässä esimerkissä määrittely on yksinkertainen, vain yhden rivin pituinen.

Esimerkki :

```
function y = esim(x)
% esim on funktio cos x-x
y=cos(x)-x
end
```

Tämän jälkeen funktion esim2 nollakohdat voidaan määrittää fzero-toiminnolla

```
>> fzero(@esim,0)

ans =

    0.7391

>>
```

MacLaurinin polynomien määrittämiseksi voidaan **Matlab**-ohjelmistossa toimia seuraavasti:

```
>> syms x
>> taylor(sin(x),x,'order',7)

ans =

x^5/120 - x^3/6 + x

>>
```

Yllä annetussa komennossa 7 määrää ensimmäisen pois jätetyn termin asteen ja x muuttujan. `taylor(f,x,'order',n,'expansionpoint',a)` tuottaa funktion f pisteeseen a liittyvän taylorin astetta n olevan polynomin.

Matlabissa on määrätyn integraalin laskemiseksi symbolisen matematiikan työkalu:

```
>> syms x
>> f=x^5

f =

x^5

>> int(f,x,1,3)

ans =

364/3

>>
```

Määrätyn integraalin likiarvon laskemista (numeerista integrointia) varten Matlabissa on `integral`, `quad`, ja `quadl`-toiminnot. Esimerkiksi

$$\int_{-4}^{4} \frac{\sin x}{x}\, dx$$

voidaan numeerisesti arvottaa seuraavalla tavalla:

```
>> f=@(x) sin(x).*x.^(-1)

f =

  function_handle with value:

    @(x)sin(x).*x.^(-1)

>> integral(f,-4,4)

ans =

    3.5164
```

Huomautus Yllä määriteltyä funktiokahvaa f kutsutaan anonyymin funktion kahvaksi, koska se määritellään käyttäjän antamalla lausekkeella $\sin(x)\cdot x^{-1}$. On myös huomattava, että tässä `x`:ää pidetään vektorina, ja sen vuoksi kertolaskua ja potenssin -1 korotusta edeltävät pisteet.

Matlabin symbolisen matematiikan työkalun avulla voidaan myös määrittää joidenkin epäoleellisten integraalien arvoja:

```
>> syms x
>> f=exp(-x^2)

f =

exp(-x^2)

>> int(f,x,-inf,inf)
```

```
ans =

pi^(1/2)
```

Matlab tuntee myös Γ-funktion:

```
>> gamma(1/2)

ans =

    1.7725

>>
```

Kompleksikonjugaatin tuottaa `conj(z)` ja itseisarvon `abs(z)`. Vaihekulma puolestaan saadaan komennolla `angle(z)`. Reaali- ja imaginaariosat saadaan eroteltua komennoilla `real(z)` ja `imag(z)`.

6 Jakaumat

Binomijakauma: Kertymäfunktion

$$\mathbb{P}(X \leq x) = \sum_{k=0}^{x} \binom{n}{k} p^k (1-p)^k$$

arvo saadaan Matlabissa komennolla

```
binocdf(x,n.p)
```

Poisson-jakauma: Kertymäfunktion

$$\mathbb{P}(X \leq x) = e^{-l} \sum_{k=0}^{x} \frac{l^k}{k!}$$

arvo saadaan Matlabissa komennolla

```
poisscdf(x,l)
```

Normaalijakauma: Kertymäfunktion

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}}\, dt$$

arvo saadaan Matlabissa komennolla

```
normcdf(x)
```

7 Fourier-muunnokset

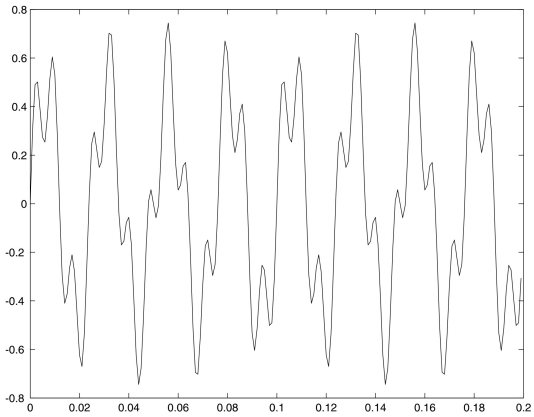
Matlabiin ohjelmoitu `fft` laskee annetuista datapisteistä kurssilla esitetyn määritelmän mukaisen diskreetin Fourier-muunnoksen *ilman* kerrointa $\frac{1}{\sqrt{N}}$ ja käänteinen operaatio `ifft` laskee vastaavan määritelmän mukaisen käänteisen Fourier-muunnoksen, jossa kertoimen $\frac{1}{\sqrt{N}}$ sijasta on kerroin $\frac{1}{N}$. Täten tällä kurssilla esitetty diskreetti fourier-muunnos saadaan lisäämällä kerroin $\frac{1}{\sqrt{N}}$.

Tarkastellaan esimerkiksi kahdesta taajuudesta, 40 ja 130 koostuvaa signaalia $s(t) = \frac{1}{2}\sin(2\pi \cdot 40t) + \frac{1}{4}\sin(2\pi \cdot 130t)$ välillä $t \in [0, 1]$:

```
>> t=0:0.001:1;
>> s=(1/2)*sin(2*pi*40*t)+(1/4)*sin(2*pi*130*t);
```

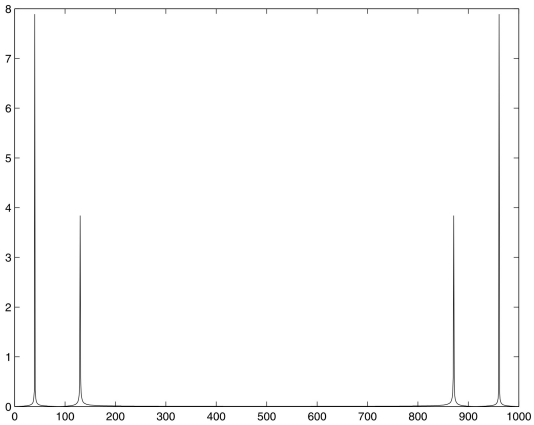
Signaalin alkuosa saadaan piirrettyä seuraavasti:

```
>> plot(t(1:200),s(1:200))
```



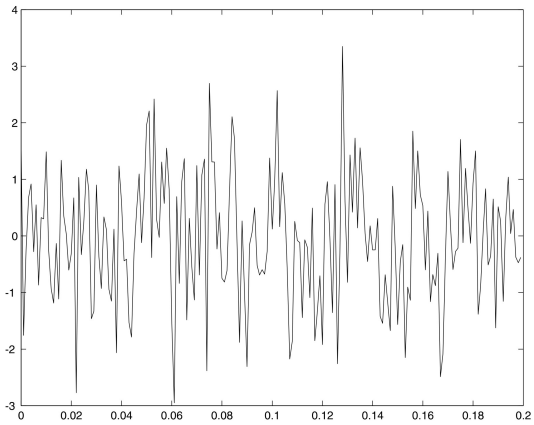
Lasketaan signaalille (1001 datapistettä) diskreetti Fourier-muunnos ja sen itseisarvo (amplitudispektri):

```
>> S=(1001)^(-1/2)*fft(s);
>> AbsS=abs(S);
>> plot(1000*t,AbsS)
```



Fourier-muunnokset tarjoavat keinon erottaa signaali kohinasta. Lisätään edellisen esimerkin signaaliin satunnaiskohinaa ja tarkastellaan näin muokatun funktion Fourier-muunnosta:

```
>> s1=s+randn(size(t));
>> plot(t(1:200),s1(1:200));
```



Tämän amplitudispektri puolestaan saadaan piirrettyä seuraavasti:

```
>> S1=(1001)^(-1/2)*fft(s1);
>> AbsS1=abs(S1);
>> plot(1000*t,AbsS1)
```

8 Laplace-muunnokset

Matlabin symbolisen laskennan työkaluihin on ohjelmoitu Laplace-muunnoksia ja käänteismuunnoksia:

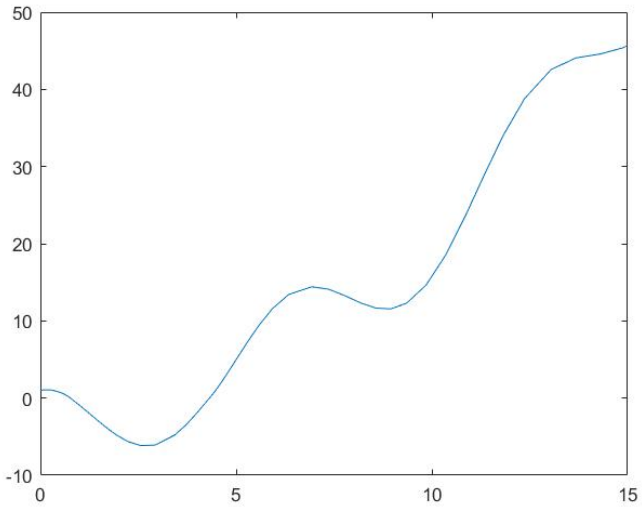
```
>> syms t
>> laplace(t^4)

ans =

24/s^5

>>
>> syms s a
>> ilaplace(1/(s-a)^2)

ans =

t*exp(a*t)

>>
>> ilaplace(1/((s-1)*(s+2)^2))

ans =

exp(t)/9 - exp(-2*t)/9 - (t*exp(-2*t))/3
```

9 Differentiaaliyhtälöiden numeerinen ratkaiseminen

Matlabiin ohjelmoitu `ode23` on muunnelma Runge-Kutta -menetelmästä, jossa pisteistö $\{t_0, t_1, t_2 \ldots\}$ ei välttämättä ole tasavälinen, vaan algoritmi voi modifioida väliä jokaisella askeleella. Korkeamman kertaluvun modifioitu Runge-Kutta -menetelmä `ode45` on niinikään ohjemoitu Matlabiin. Algoritmi `ode45` on epäilemättä numeerisesti tarkempi, mutta `ode23` lienee monissa tilanteissa nopeampi.

Esimerkki Tarkastellaan differentiaaliyhtälöä $y' = \sqrt{|y|} - 6\sin(t)$, jossa t on muuttuja ja y tuntematon funktio ja alkuehto on $y(0) = 1$. Tämä on epälineaarinen 1. kertaluvun yhtälö, johon kurssilla esitetyt eksatit menetelmät eivät lainkaan pure. Numeerinen ratkaisu ja sen kuvaaja voidaan kuitenkin esim. välillä $[0, 15]$ piirtää Matlabilla seuraavasti:

```
>> [t,y]=ode23(@(t,y)(sqrt(abs(y))-6*sin(t)),[0,15],1);
>> plot(t,y)
```



Ylläolevassa esityksessä funktiokahva `@(t,y)(sqrt(abs(y))-6*sin(t))` viittaa differentiaaliyhtälön oikeaan puoleen, $[0, 15]$ väliin jolla numeerinen ratkaisu lasketaan ja viimeisin luku esittää alkuehtoa $y(0) = 1$. Algoritmi `ode23` antaa tulosteeksi aikapisteitä kuvaavan sarakevektorin `t` sekä funktion arvoja esittävän sarakevektorin `y`, joista kuvaaja voidaan piirtää komennolla `plot(t,y)`.

Tarkastellaan differentiaaliyhtäläparia

$$\begin{cases} \frac{dx}{dt} &= -2x^2 \\ \frac{dy}{dt} &= -\frac{1}{2}xy \end{cases} \tag{1}$$

alkuehdoilla $x(0) = 2$, $y(0) = 1$.

DY-parin (1) tapauksessa ei liene mielekästä yrittää syöttää Matlabiin yhdelle syöteriville koko paria vaan pikemminkin määrittää DY-parin (1):n oikea puoli omaksi vektoriarvoiseksi funktiokseen `xypari`, jonka jälkeen voi käyttää edellisen esimerkin kaltaista Matlab-komentoa

$$[\texttt{t}, \texttt{xy}] = \texttt{ode23}(@\texttt{xypari}, [0, 10], [2\ 1]), \tag{2}$$

jossa tulosteena on aikaväli `t` sekä pystyvektori `xy`, jonka ylempi alkio koostuu funktion `xy(1)` (=`x`) arvoista ja alempi funktion `xy(2)` (=`y`) arvoista vektorin `t` osoittamissa pisteissä. Komennossa `[0,10]` viittaa parametrin `t` vaihteluväliin ja viimeisin `[2 1]` viittaa alkuarvoihin: funktion `xy(1)` arvo pisteessä nolla on 2 ja funktion `xy(2)` arvo pisteessä nolla on 1.

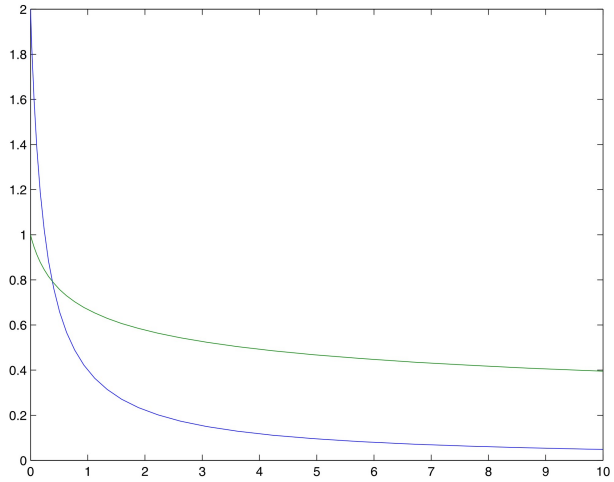
Rivin (2) suorittamista varten voidaan funktio xypari määritellä seuraavasti (ja tallentaa m-tiedostoksi).

```
function arvo=xypari(t,xy)
x=xy(1);
y=xy(2);
```

```
arvo(1,1)=-2*x*x;
arvo(2,1)=-0.5*x*y;
%Yllä oleva määrittelee funktion xypari. Funktion arvo
%on 2-pituinen pystyvektori, joka Matlabissa voidaan
%esittää 2 x 1-matriisina: arvo(1,1) on 1. alkio ja
%arvo(2,1) pystyvektorin 2. alkio.
end
```

Tässä yhteydessä on huomattava että vaikka DY-pari (1) on autonominen, eli aika `t` ei esiinnyt eksplisiittisesti oikealla puolella, on se silti otettava muodollisesti huomioon muuttujana funktiota `xypari` määriteltäessä. Syy tähän on se, että `ode23` vaatii muuttujan `t`.

Kun `xypari` on tallennettu m-tiedostoksi, voidaan suorittaa rivi (2) ja sen jälkeen piirtää funktioiden `xy(1)` (=`x`) ja `xy(2)` (=`y`) kuvaajat komennolla `plot(t,xy)`.



Kuviossa funktion x kuvaaja alkaa arvosta 2 ja y:n arvosta 1. Tarkempi analyysi osoittaa, että molempien funktioiden kuvaajat lähestyvät nollaa t:n kasvaessa.

On myös mahdollista kirjoittaa kaikki Matlabin tarvitsemat komentorivit yhteen tiedostoon, mukaanlukien kuvaajan piirtäminen. Tätä havainnollistaa seuraava esimerkki.

Esimerkki Ratkaistaan RCL-piiriin liittyvä differentiaaliyhtälö

$$I'' + 2 \cdot 10^3 \cdot I' + 10^6 I = 325 \cdot 10^4 \pi \cos(100\pi t)$$

numeerisesti välillä $[0, 0.1]$. Muutetaan ensin differentiaaliyhtälö DY-ryhmäksi, jossa esiintyy vain ensimmäisen kertaluvun derivaattoja. Tätä varten otetaan käyttöön uusi funktio $J = I'$, jolloin saadaan DY-pari

$$\begin{cases} I' &= J \\ J' &= -2 \cdot 10^3 J - 10^6 I + 325 \cdot 10^4 \pi \cos(100\pi t) \end{cases} .$$

Tämän numeeriseen ratkaisemiseen voidaan käyttää seuraavanlaista Matlab-koodia (Kokeile itse):

```
function esim
[t,tulos]=ode23(@RCL,[0 0.1],[0 0]);
plot(t,tulos(:,1))
return

%Yllä olevat rivit käyttävät ode23-ohjelmaa DY-parin likimääräiseen
%ratkaisemiseen ja kuvaajan piirtämiseen. tulos(:,1) poimii
%matriisin tulos 1. sarakkeen.
```

```
function arvo=RCL(t,IJ)
I=IJ(1);
J=IJ(2);
arvo(1,1)=J;
arvo(2,1)=-2*10^3*J-10^6*I+325*10^4*pi*cos(100*pi*t);
return

%Yllä oleva määrittelee funktion RCL. Funktion arvo on 2-pituinen
%pystyvektori, joka Matlabissa voidaan esittää 2*1-matriisina:
%arvo(1,2) on 1. alkio ja arvo(2,1) pystyvektorin 2. alkio.
```

Kuvaaja: