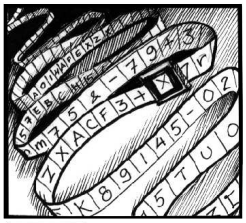


Vektorit, matriisit, Händel ja vaalit

Heikki Apiola
Matematiikan laitos, Teknillinen korkeakoulu

Lineaarialgebran maistiaiset

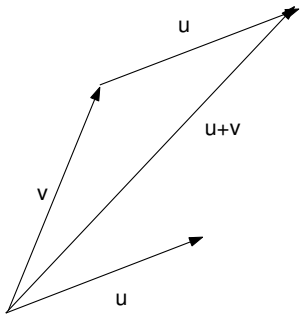
Lineaarialgebra on matematiikan ala, joka on sekä teorian että sovellusten kannalta hyvin keskeinen. Se yleistää havainnollisen vektorikäsitteen mitä moninaisimpiin ja yllättävimpiin tilanteisiin.
Matriisit ovat lukutaulukoita, joille määritellään laskutoimitukset. Taulukkoon koottua tietoa käsitellään yhtenä objektina samankaltaisin laskusäännöin kuin lukuja. Kun matriisia sovelletaan vektoriin, syntyy liikettä, dynamiikkaa.
Lineaarialgebran merkitys matematiikan sovelluksille on kasvanut käsi kädessä tietotekniikan kehityksen kanssa. Niinpä alan ohjelmisto on varsin kypsässä ja kattavassa tilassa, toki kaiken aikaa kehittyen.
”Lineaarialgebran kieltä” puhuvia laadukkaita ohjelmistoja on saatavilla. Kirjoituksessa esittelemme esimerkkien valossa vapaasti saatavissa olevaa ohjelmaa SCILAB, joka noudattaa varsin uskollisesti tunnetun ja maailmalla hyvin suositun MATLAB-ohjelman syntaksia.

Kirjoituksessa pyrimme valottamaan mm. vektorikäsitteen moninaisia ja kenties yllättäviä ilmentymiä havainnollisista tason suuntanuolista aina digitaalisiin musiikkivektoreihin. Matriisien alalla korostamme tässä dynamiikkaa ja prosessia aina eduskuntavaaleihin saakka.

Kyseessä ei ole yritys opettaa varsinaista lineaarialgebran kieltä yhden kirjoituksen puitteissa, vaan pikemminkin antaa maistiaisia ja ruokahalua sekä näyttää kenties yllättäviäkin näiden käsitteiden käyttöalueita ja -tapoja.

Geometriset ja fysiikan vektorit

Geometriset vektorit tasossa ja avaruudessa lienevät koulumatematiikasta ja fysiikasta tuttuja olioita . Tyypillinen fysiikan ensituttavuus on voima, jolla on suuruus ja suunta. Tällaista suuretta sanotaan *vektorisuureeksi*. Sitä voidaan geometrisesti kuvata voiman suuntaa osoittavalla nuolella, jonka pituus kuvaa voiman suuruutta.



KUVA 1. Vektorisumma $\boldsymbol{u} + \boldsymbol{v}$

Vektorien $\boldsymbol{u}$ ja $\boldsymbol{v}$ summa $\boldsymbol{u}+\boldsymbol{v}$ määritellään kuvan osoittamalla tavalla. On syytä panna merkille, että vektorin määrää suunta ja pituus, ts. kaikki **samansuuntaiset ja yhtä pitkät** suuntajanat alkupisteestä riippumatta **edustavat samaa vektoria**. Kuvassa se ilmenee kahtena $\boldsymbol{u}$:lla merkittynä suuntajanana.

Tason vektoreita voidaan toisaalta käsitellä puhtaasti algebrallisesti lukupareina $\boldsymbol{p} = (x, y)$. Geometrisena taustana on tällöin tason pisteen koordinaattiesitys. Yhteys edelliseen nuoliajatukseen saadaan piirtämällä vektorinuoli, jonka alkupiste asetetaan koordinaatiston origoon O. Pisteiden $\boldsymbol{p}_1 = (x_1, y_1)$ ja $\boldsymbol{p}_2 = (x_2, y_2)$ summa määritellään luontevasti: $\boldsymbol{p}_1+\boldsymbol{p}_2 = (x_1+x_2, y_1+y_2)$, ts. lasketaan vastinkoordinaatit yhteen. Geometrisesti tulos saadaan laskemalla vektorien $\boldsymbol{v} = \overline{0\,\boldsymbol{p}_1}$ ja $\boldsymbol{u} = \overline{\boldsymbol{p}_1\,\boldsymbol{p}_2}$ summavektori, jonka kärjen koordinaatit antavat juuri edellä määritellyn summapisteen $\boldsymbol{p}_1 + \boldsymbol{p}_2$.

Vektoreille määritellään myös toinen laskutoimitus, luvulla eli *skalaarilla* [1] kertominen.

Vektorin $\boldsymbol{v}$ kertominen skalaarilla c tarkoittaa geometrisessa mallissa vektorin pituuden kertomista $|c|$:lla ja lisäksi suunnan vaihtoa, jos $c < 0$. Algebrallisessa mallissa vektorin komponentit kerrotaan c:llä, ts. jos $\boldsymbol{v} = (x, y)$, niin

$$c\,\boldsymbol{v} = (c\,x, c\,y).$$

Edellä oleva yleistyy suoraan 3-ulotteiseen avaruuteen, algebrallisessa vektorimallissa otetaan mukaan kolmas koordinaatti: $\boldsymbol{v} = (x, y, z)$, geometrisessa suuntajanat leijailevat avaruudessa.

Järjestettyjen lukuparien joukkoa kutsutaan (vektori)tasoksi ja merkitään symbolilla $\mathbb{R}^2$. Vastaavasti lukukolmikoiden joukkoa kutsutaan (3-ulotteiseksi) vektoriavaruudeksi ja käytetään merkintää $\mathbb{R}^3$.

Geometriaan viittaavassa puheessamme nimitämme samaa oliota vuoroin "pisteeksi" ja vuoroin "vektoriksi". Jälkimäisessä tapauksessa meillä on takaraivossamme ajatus origosta pisteeseen piirretystä paikkavektorista. Esimerkiksi laskiessamme "pisteitä" yhteen, laskemme geometrisesti ajatellen yhteen vastaavia paikkavektoreita.

Lisää ulottuvuuksia

Algebrallisen vektorimallin etu on, että se voidaan suoraan yleistää mielivaltaisen korkeaan dimensioon n yksinkertaisesti kutsumalla **vektoreiksi** lukujonoja $\boldsymbol{u} = (x_1, x_2, \ldots, x_n)$ ja määrittelemällä laskutoimitukset *vektoreille* $\boldsymbol{u} = (x_1, x_2, \ldots, x_n)$ ja $\boldsymbol{v} = (y_1, y_2, \ldots, y_n)$ sekä skalaarille c asettamalla:

$$\begin{cases} \boldsymbol{u} + \boldsymbol{v} = (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n) \\ c\,\boldsymbol{u} = (c\,x_1, c\,x_2, \ldots, c\,x_n) \end{cases}$$

Vektorien *vähennyslasku* suoritetaan vähentämällä toisistaan vastinkoordinaatit, *nollavektori* on vektori

$$\mathbf{0} = (0, 0, \ldots, 0)\,.$$

Vektorien yleiset laskusäännöt ovat aivan samat kuin vastaavat reaalilukujen laskusäännöt, jos ajattelemme skaarilla kertomista tulona. Näitä sääntöjä ovat liitäntä- vaihdanta- ja osittelulait. Kaikki palautuvat suoraan koordinaateilla suoritettaviin lukujen vastaaviin sääntöihin.

Toinen yleistysmahdollisuus on sallia kompleksiset komponentit ja skalaarit. Pitäydymme tässä kirjoituksessa "turvallisuussyistä" reaalisissa vektorikäsitteissä, vaikka lopussa mainitussa ominaisarvoteoriassa kompleksisilta ei voida välttyä.

Sisätulo, pituus, kulma

Geometrisessa vektorimallissa vektorin $\boldsymbol{u}$ pituus on tunnettu peruskäsite, jota merkittäköön $||\boldsymbol{u}||$. Jos $\boldsymbol{u} = (x, y)$, niin *Pythagoraan lauseen* mukaan $||\boldsymbol{u}|| = \sqrt{x^2 + y^2}$. Jos kyseessä on 3-ulotteisen avaruuden vektori $\boldsymbol{u} = (x, y, z)$, niin $||\boldsymbol{u}|| = \sqrt{x^2 + y^2 + z^2}$.

Tässä on selvä malli yleiseen tilanteeseen, määritellään siis vektorin $\boldsymbol{u} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ pituus eli *normi* kaavalla

$$||\boldsymbol{u}|| = \sqrt{x_1^2 + x_2^2 + \ldots + x_n^2}.$$

Määritellään nyt suoraan yleisessä tilanteessa vektorien $\boldsymbol{u}$ ja $\boldsymbol{v}$ *sisätulo* (skalaaritulo, pistetulo) kaavalla

$$\boldsymbol{u} \cdot \boldsymbol{v} = x_1\,y_1 + x_2\,y_2 + \ldots + x_n\,y_n.$$

Geometrisessa vektorimallissa sisätulo voidaan määritellä kordinaattivapaasti kaavalla
$\boldsymbol{u} \cdot \boldsymbol{v} = ||u||\,||\boldsymbol{v}||\,\cos\gamma$, missä γ on vektorien $\boldsymbol{u}$ ja $\boldsymbol{v}$ välinen kulma. Määritelmien yhtäpitävyys voidaan osoittaa esimerkiksi *kosinilauseella*.

Yleisessä n-ulotteisessa vektorimallissa ei luonnostaan ole käsitettä "vektorien välinen kulma". Nyt voimme halutessamme määritellä tuon kulman yllä olevalla sisätulokaavalla. Yleisen vektorikulman tärkein merkitys liittyy *kohtisuoruuden* eli *ortogonaalisuuden* käsitteen yleistykseen, ehtona on $\boldsymbol{u} \cdot \boldsymbol{v} = 0$

[1] Skalaari, "scalar" – sellainen, jolla skaalataan

Olemme irrottaneet vektoriopin geometrisesta taustastaan ja antaneet täsmällisen sisällön n-ulotteisen avaruuden vektoreille. Tämä on pohjana alkujohdannossa kuvatulle tärkeälle matematiikan alalle, *lineaarialgebralle*. Erityisen viehättävää lineaarialgebrassa on, että pohjana on koko ajan geometrinen intuitio 2- ja 3-ulotteisten geometristen vektorien parissa, joka auttaa koko perusteorian ymmärtämisessä ja todistusten ideoinnissa olennaisella tavalla. Se toimii yleisen teorian "mallikuvana" kaiken aikaa.

Vektorilaskentaa tietokoneella

Sovelluksissa suoritamme vektorilaskentaa usein hyvin moniulotteisilla vektoreilla. Laskut muodostuvat pitkiksi ja tuloksen jälkikäsittely saattaa sisältää vaikkapa vektorin muuntamisen kuvaksi tai ääneksi.

Ohjelmat, joiden perustietorakenteisiin kuuluvat vektorit, ja laskuoperaatiot kattavat vektorialgebran, ovat korvaamattomia työvälineitä.

Olen aiemmin esitellyt symbolilaskentaohjelmistoa MAPLE
`http://solmu.math.helsinki.fi/1999/5/apiola/`
ja numeerista vektori/matriisiohjelmistoa MATLAB.
`solmu.math.helsinki.fi/2004/3/apiola.pdf`
Ongelmana lukijoille on ohjelmien maksullisuus. Esittelen tällä kertaa vapaasti saatavaa ohjelmaa SCILAB, joka on perusperiaatteiltaan aivan samanlainen kuin MATLAB. Ohjelmassa on avoimesti pyritty varsin pitkälle menevään yhteensopivuuteen MATLAB'n kanssa. (Toinen vastaava vapaasti saatava ohjelmisto on OCTAVE.)

Ohjelman voit ladata itsellesi osoitteesta
`http://www.scilab.org/` . Käyttöohjeita on ranskan, saksan ja englanninkin kielellä saatavissa:
`http://www.scilab.org/publications/`

Lyhyt Scilab/Matlab-käyttöohje

SCILAB-ohjelman opiskelun voit aloittaa jostakin MATLAB-oppaasta, suomenkielinen MATLAB-käyttöohje on viitteissä.

Kun SCILAB käynnistetään, avautuu komentoikkuna. Tähän voidaan kirjoittaa ohjelman tuntemia komentoja, jotka ohjelma tulkitsee ja toteuttaa samantien (tai antaa virheilmoituksen). Tarkoituksena on antaa pieniä maistiaisia ohjelman käytöstä senverran, että esittämämme esimerkit voidaan suorittaa ja ymmärtää. Tarkoitus on innostaa lukijaa omiin kokeiluihin ohjelmien avustus- ja dokumentointijärjestelmien ja viitteissä mainittujen lähteiden avulla. Voit aloittaa kirjoittamalla jotain tämäntyylistä SCILAB- (tai MATLAB-) komenoikkunaan:

```
// Ensimmäinen Scilab/Matlab-istunto
// Scilab-kommentti: //  Matlab-kommentti: %
u=[1 2 3 4]    // Luodaan vektori u
v=[-2 2 -1 5 ]  // samoin toinen vektori v
w=-u+2*v // Lineaarikombinaatio sijoitetaan
        // muuttujaan w
u.*v   // Kerrotaan vektorin vastinkomponentit
u*v'   // u:n ja v:n sisätulo.
```

Selityksiä:

1. Lineaarikombinaatio tarkoittaa muotoa
$c_1 \boldsymbol{v}_1 + \boldsymbol{c}_2 \boldsymbol{v}_2 + \ldots + c_k \boldsymbol{v}_k$ olevaa summaa.

2. Viimeistä edellinen rivi: `u.*v` on pisteittäin eli vastinalkioittain tapahtuva kertolasku. Se ei siis ole varsinainen vektorilaskutoimitus, mutta monessa yhteydessä, erityisesti vektorilausekkeissa tuiki tarpeellinen.

3. Viimeinen rivi on erikoistapaus matriisikertolaskusta, siitä enemmän tuonnempana.

Ohjelman käytön käytännön vihjeitä

Yllä olevia ja muutamia vastaavia kirjoitettuasi ei kannata jatkaa työskentelyä suoraan komentoikkunassa, koska työsi häviää taivaan tuuliin. Ainoa, mitä siinä voit tehdä, on selata komentopuskuria nuoli-ylösnäppäimellä ja editoida komentoriviä.

Sensijaan kannattaa avata otsikkopalkista `Editor` ja kirjoittaa komentoja kommentteineen editoriikkunaan. SCILAB tarjoaa varsin kätevän toimintatavan: Maalataan halutut komentorivit ja painetan `CTR-Y` tai valitaan otsikkoriviltä `Execute`. Kyseiset rivit siirtyvät välittömästi tulkille suoritettaviksi.
Toinen mahdollisuus on käyttää omaa mielieditoria, kuten EMACS, NOTEPAD, WORDPAD, ym. ja leikata/liimata editorista komentoja SCILAB-komentoikkunaan. Tämä vaatii yhden näppäinpainalluksen enemmän, mutta on ehkä tutumpana turvallisempaa.

Kuvia ja musiikkia vektoreilla

Siirrytään nyt katsomaan vektorikäsitteen erilaisia ilmenemismuotoja. Nähdään, että yleistys uusiin ulottuvuuksiin ei ole pelkkää matemaatikkojen harjoittamaa esteettistä, maailmalle vierasta abstrahointihöpinää, vaan se avaa niin silmiä kuin korviakin näkemään ja kuulemaan uusilla tavoilla. Viimemainitusta olkoon esimerkkinä digitaalinen äänenen käsittely, josta itse kukin voi nauttia CD-levyjä ja äänitiedostoja kuunnellessaan. Ensinmainitusta voitaisiin vastaavasti ottaa digitaalinen kuvankäsittely, mutta asiaa harkittuani valitsin tällä kertaa äänen. Toki kuviakin sentään katsellaan funktioiden kuvaajien muodossa.

Funktion f kuvaajan piirto käsipelillä tapahtuisi yksinkertaisimmillaan niin, että laskettaisiin tasavälisessä pisteistössä $x_1, x_2, \ldots, x_n$ funktion arvot $y_i = f(x_i), i = 1, \ldots, n$, pyöräyteltäisiin kynällä pisteet koordinaatistoon ja yhdistettäsiin peräkkäiset pisteet jananpätkillä.

Juuri näin tekee `plot`-funktio MATLAB/SCILAB:ssa. Peruskutsu on `plot(x,y)`, missä x ja y ovat samanpituisia vektoreita. Ohjelmissamme on funktio `linspace`, jolla saadaan aikaan tasavälinen x-pisteistö. Seuraavassa esimerkissä muodostetaan x-vektori jakamalla väli $[-\pi, \pi]$ 15:een osaan, ja sitten y-vektori laskemalla sin-funkion arvot x-pisteissä. Kaikki matemaattiset funktiot näissä vektoriohjelmissa toimivat niin, että sovellettaessa funktiota (x-)vektoriin, tuloksena saadaan (y-)vektori, joka koostuu vastaavista funktion arvoista. Tämä tekee käytön hyvin vaivattomaksi.

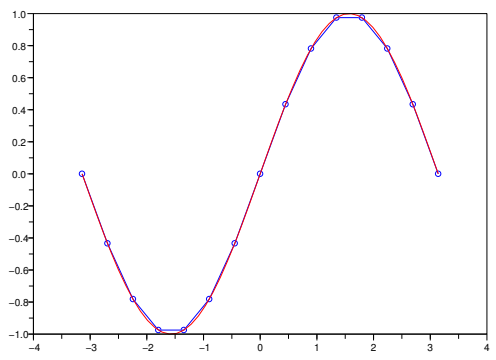
Pari pientä, hyödyllistä yksityiskohtaa:

1. Puolipiste (;) komennon perässä estää tulostuksen ruudulle.
2. Kummassakin ohjelmassa on erikoissymboleja joillekin vakioille, π on MATLAB:ssa `pi` ja SCILAB:ssa `%pi`. Toimimme tähän tapaan:

```
pi=%pi   // Matlab-yhteensopivuuden vuoksi
x=linspace(-pi,pi,15); y=sin(x);
[x' y'] // xy-arvojen taulukko
clf()   // Grafiikkaruudun pyyhkiminen
plot(x,y); // Pisteet ja murtoviiva
plot(x,y,'o') // Pelkät pisteet
```

Piirretään tarkemmin jakamalla väli 50:een osaan. Piirretään samaan kuvaan punaisella värillä. Nähdään, että tässä on riittävä määrä "näytteitä" sini-funktiosta ao. välillä, jotta silmä näkee kuvaajan kauniisti kaartuvana sinikäyränä.

```
x=linspace(-pi,pi,50); y=sin(x);
plot(x,y,'r') // 'r' viittaa väriin "red"
```



SININ KUVAAJA, näytteitä 15 ja 50

Musiikkia vektoreilla

Edellä näimme, miten vektoreilla voidaan saada aikaan silmänruokaa vaikkapa sini-funktion kuvaajan ihastelemiseksi. Saataisiinko myös korvin kuultavaa? Jatkakaamme sini-funktion parissa.

Perussävelenä tarkastelkaamme yksiviivaista a-säveltä. Kyseessä on siniaalto, jonka värähtelytaajuus on 440 Hz, ts. 440 värähdystä sekunissa. Värähtelyn kulmataajuus on tällöin $2\,\pi\,440$ radiaania sekunnissa ja sitä esittää siniaalto $y(t) = \sin(2\,\pi\,440\,t)$.

Kun tällaista siniaaltoa digitoidaan, täytyy näytteenottotaajuuden olla mielellään selvästi enemmän kuin $2 \cdot 440$. Otetaan varman päälle: 10 000. Seuraavassa istunnossa jaetaan ensin 1 sekunnin aikaväli $(0, 1)$ 10 000:een osaan, piirretään 100 ensimmäistä kuvaajan pistettä vastaten 1/100 sekunnin aikaa ja sitten kuunnellaan. Funktio `sound` on samaa korvalle kuin `plot` on silmälle.

```
a1=2*%pi*440;  // Scilab
t=linspace(0,1,10000);y2=sin(a1*2*%pi*t);
clf();plot(t(1:100),y2(1:100))
sound(y2,10000);
```

Koska lukija ei lehden sivulta saa ääntä kuuluviin, niin jätettäköön kuvakin tulostamatta.

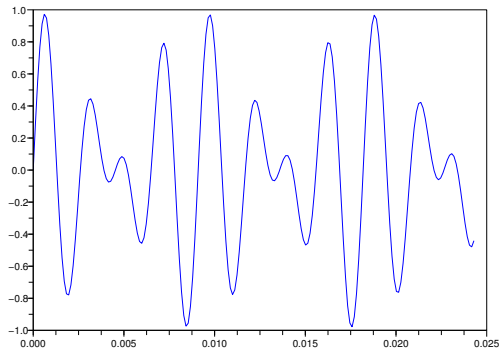
Sointuja

Esittelemme muutaman perusidean ja työkalun digitaalisen äänenkäsittelyn alalla. Muodostetaan yksiviivaisen oktaavin nuottien taajuudet ja sijoitetaan muuttujiin

```
c=261.6;d=293.7;e=329.6;f=349.2;
g=392;a=440;h=493.9;
```

Jospa soittaisimme sekunnin verran intervallia *kvintti*, joka voidaan toteuttaa vaikkapa viulun d- ja a-kieliä yhtäaikaa soittamalla. Käytetään näytteenottotaajuutta `nt=8192`. Intervalli saadaan laskemalla värähtelyjen summa, ts. laskemme näytevektorit yhteen. Skaalataan jakamalla 2:lla, koska `sound`-funktio haluaa värähtelyn arvoalueen (y-arvot) välille $[-1, 1]$.

```
nt=8192; pi=%pi; t=linspace(0,1,nt);
kvintti=0.5*sin(2*pi*d*t)+0.5*sin(2*pi*a*t);
sound(kvintti,nt);
clf();plot(t(1:200),kvintti(1:200))
```



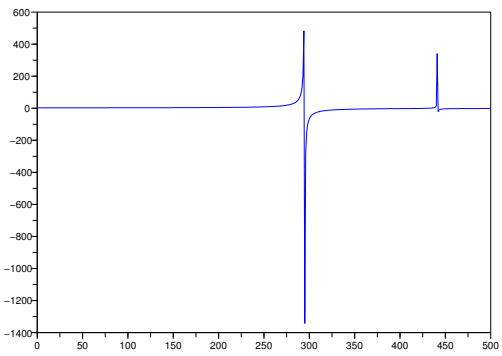
KVINTTI: Viulun d- ja a-kielet soivat.

Raottaaksemme ovea hiukan signaalinkäsittelyn matemaattisten perustyökalujen suhteen, otamme esimerkin ns. nopeasta Fourier-muunnoksesta, FFT ("Fast Fourier Transform"). Kyseessä on väline, joka on mainittu joissain arvioissa 1900-luvun merkittävimpiin matemaattisiin algoritmeihin kuuluvaksi. Paino on sanalla "nopea", joka on mahdollistanut reaaliaikaiset digitaalisen signaalinkäsittelyn sovellukset.

Tässä yhteydessä emme selitä, mitä Fourier-muunnos tarkoittaa. Katsomme edellistä esimerkkiä jatkaen, minkälaiseen maailmaan tuo muunnos meidät vie.

MATLAB/SCILAB:ssa on funkio `fft`, jota käytämme "mustana laatikkona". Tämä muunnos toimii kaikkein tehokkaimmin vektoreille, joiden pituus on jokin 2:n potenssi. Näytteenottotaajuutemme valittiin siitä syystä luvuksi 8192, kun tämä sattuu olemaan 2^{13}. Mainittakoon, että CD-levyn musiikin näytteenottotaajuus on 44 100 näytettä sekunnissa. (Tämä ei ole 2:n potenssi, syy on se, että $2^{15} = 32768$ on liian pieni, jotta korkeimmat ylä-äänet eivät vääristyisi ja $2^{16} = 65\,536$ on aivan turhan suuri.)

Suoritetaanpa nyt kvintti-vektorillemme Fourier-muunnos SCILAB:n `fft`-funktiota käyttäen.



KVINTIN FOURIER-MUUNNOS Viulun d- ja a-taajuudet erottuvat.

```
Fkvintti=fft(kvintti);
clf();plot(Fkvintti(1:500))
```

Kahden siniverähtelyn kuva on muunnettu "taajuustasoon". Vaaka-akselilla on taajuudet ja pystyakseli edustaa kunkin taajuuden tehoon viittaavaa arvoa. Nähdään, että kvintin taajuuskuvassa kaikki muut taajuudet ovat nolla-tehoisia, paitsi ~ 293 Hz ja ~ 440 Hz.

Käänteismuunnoksella päästään takaisin. Se on niinikään suoraan saatavilla ohjemissamme nimellä `ifft`. Jos haluaisimme viulun kieliä vaikka vähän virittää, se tapahtuisi helposti taajuustasossa, sitten palaisimme käänteismuunnoksella aikatasoon soittamaan puhdasta kvinttiä.

Tuiki tuiki tähtönen – Halleluja

Tehdään pieni viihteellinen/taiteellinen sivuhyppäys, kun meillä nyt on koko nuotisto näpeissämme. Kirjoitetaan vektori `tuiki`, jossa on laulun nuotit. Oletetaan, että edellä olevat nuottien taajuusarvot on sijoitettu muuttujiin $c, d, \ldots$

```
tuiki=[c,c,g,g,a,a g f f e e d d];
tuiki=[tuiki c g g f f e e d];//loistat vaan..
```

Yksinkertaisella ohjelmasilmukalla voimme nyt soittaa suositun lastenlaulun. Komento `halt()` pysäyttää ohjelman siihen saakka, kunnes painetaan jotain näppäintä.

```
t=linspace(0,1,8192);
for k=1:length(tuiki)
  halt()
  y=sin(tuiki(k)*t);sound(y,8192)
end;
```

Jos laulat kuorossa, etkä soita sujuvasti mitään instrumenttia, voit tällä menetelmällä harjoitella helposti omaa stemmaasi, otin sen itse käyttöön näiden kehittelyjen seurauksena.

Jotta pääsisimme käsiksi hiukan vaativampaan taidenautintoon, mainittakoon, että MATLAB:ssa on digitoituna Ote *Händelin* Messias-oratorioon kuuluvasta Halleluja-kuorosta. Kas näin:

```
>> load handel
>> sound(y)  % Halleluja halleluja ...
             % Ei Hard Rock!
```

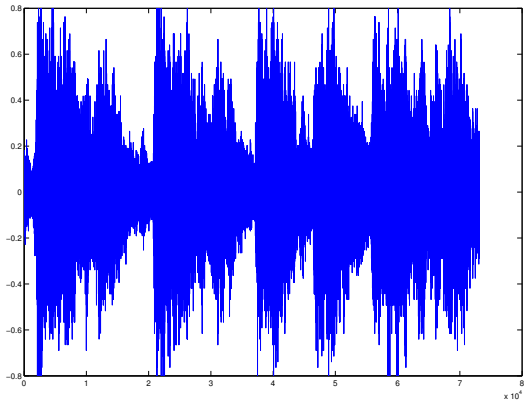
Katsotaan hiukan tuota Halleluja-vektoria:

```
>> length(y)  % Vektorin pituus
ans =
       73113

>> y(1:5)    % 5 ensimmäistä komponenttia

ans =
         0
   -0.0062
   -0.0750
   -0.0312
    0.0062  % Ihan totta, pelkistä tylsistä
            % numeroista koostuva vektori!

>> plot(y)
```



Händelin Messias: Halleluja-kuoro

Lineaarisia yhtälöryhmiä

Palatkaamme korkealentoisista taidenautintojen sfääreistä takaisin maanpinnalle arkiseempaan aherrukseen.
Matriisilaskennan ensimmäinen ja tärkeä käyttöaalue on lineaaristen yhtälöryhmien teoria ja ratkaisumenetelmät.

En tällä kertaa paneudu tähän aiheeseen muuten kuin selittämällä aivan lyhyesti ratkaisutekniikan pääperiaatteet samalla johdatellen matriiseihin.

Katsotaanpa esimerkkiä:

$$\begin{cases} x_1 - 2\,x_2 + x_3 &= 0 \\ 2\,x_2 - 8\,x_3 &= 8 \\ -4\,x_1 + 5\,x_2 + 9\,x_3 &= -9 \end{cases}$$

Tällaisia yhtälöryhmiähän ratkaistaan niin, että yhtälöitä muokataan operaatioilla, joissa yhtälöryhmän ratkaisujoukko pysyy samana. Näitä operaatioita ovat yhtälöiden yhteenlasku, vakiolla kertominen ja yhtälöiden järjestyksen vaihtaminen.

Koulussahan eliminointitekniikkaa harjoitellaan, mutta tietääkseni ei esitetä systemaattista tapaa, *Gaussin eliminaatiomenetelmää*. Siinä päädytään aina ”yläkolmiomuotoon”, josta yhtälöryhmän ratkaisujen olemassaolo- ja lukumääräkysymykset voidaan selvittää ja samalla saada ratkaisut lasketuksi silloin, kun niitä on.

Huomataan, että yhtälöryhmä on annettu, kun kaikki tuntemattomien kertoimet ja tunnetut oikean puolen luvut on annettu. Niinpä kaikki yhtälöryhmää koskeva informaatio sisältyy vasemmanpuoleiseen ”matriisiin”, eli lukutaulukkoon. Merkitään symbolilla ($\sim$) yhtälöryhmiä edustavien matriisien ekvivalenssia, eli ratkaisujoukkojen samuutta.

$$\begin{bmatrix} 1 & -2 & 1 & 0 \\ 0 & 2 & -8 & 8 \\ -4 & 5 & 9 & -9 \end{bmatrix} \sim \begin{bmatrix} 1 & -2 & 1 & 0 \\ 0 & 2 & -8 & 8 \\ 0 & -3 & 13 & -9 \end{bmatrix}$$

Kirjoitusvaivoja voidaan hiukan säästää kohdistamalla sallitut muokkausoperaatiot (”Gaussin rivioperaatiot”) suoraan tähän matriisiin ja mikä tärkeämpää, tällöin menettelyn muuntaminen tietokoneohjelmaksi käy vaivattomasti.

Tavoitteena on saada nollat päälävistäjän alapuolelle. Ensimmäisessä sarakkeessa on jo yksi 0, toinen saadaan näin: **Kerrotaan** 1. **rivi** 4**:llä ja lisätään kolmanteen**, vain kolmas rivi muuttuu. Näin päädytään oikeanpuoleiseen matriisiin.
Nyt voidaan 2. rivi (yhtälö) jakaa 2:lla ja sen jälkeen kertoa 3:lla ja lisätä kolmanteen. Näin päädytään matriisiin

$$\begin{bmatrix} 1 & -2 & 1 & 0 \\ 0 & 1 & -4 & 4 \\ 0 & 0 & 1 & 3 \end{bmatrix} \quad \leftrightarrow \quad \begin{cases} x_1 - 2\,x_2 + x_3 = 0 \\ x_2 - 4\,x_3 = 4 \\ x_3 = 3 \end{cases}$$

Eliminaatiovaihe on valmis, päädyimme yläkolmiomuotoon. Koska x_3:n kerroin $\neq 0$, saamme yksikäsitteisen ratkaisun ratkaisemalla alhaalta ylöspäin edeten kolme erillistä ensimmäisen asteen yhtälöä. Näemme, että saadaan yksikäsitteinen ratkaisu, olipa oikean puolen ”pystyvektori” valittu miten hyvänsä. Tässä tapauksessa saadaan alimmasta: $x_3 = 3$, sitten toisesta: $x_2 = 16$ ja vihdoin ensimmäisestä: $x_1 = 29$.

Gaussin eliminaatiomenettely voidaan nähdä kaksivaiheisena prosessina: 1) Eteenpäin eliminointi (”forward elimination”) ja takaisinsijoitus (”backsubstitution”), jälkimmäinen siinä tapauksessa, että 1. vaihe lupaa ratkaisuja olevan.

Laskentaa matriiseilla

Matriisilla tarkoitetaan yksinkertaisesti suorakulmion muotoista lukutaulukkoa. Jos rivejä on m ja sarakkeita n kappaletta, puhutaan $m \times n$-matriisista.

Merkitään edellä olevassa esimerkissä
A:lla yhtälöryhmän kertoimien muodostamaa
3×3-matriisia, joka siis koostuu edellä olevan matriisin 3:sta ensimmäisestä sarakkeesta.

$$A = \begin{bmatrix} 1 & -2 & 1 \\ 0 & 2 & -8 \\ -4 & 5 & 9 \end{bmatrix}$$

Otetaan myös käyttöön vektorit $\boldsymbol{x} = [x_1, x_2, x_3]^T$ ja $\boldsymbol{b} = [0, 4, 3]^T$. Yläindeksi ^{T} viittaa ns. *transpoosiin*, joka tarkoittaa tässä, että vektorit ajatellaan "pystyvektoreiksi", eli kyseessä on oikeastaan 3×1-matriisi. [2]

Määrittelemme **matriisi kertaa vektori**- tulon niin, että yhtälöryhmämme voidaan kirjoittaa muotoon $A\,\boldsymbol{x} = \boldsymbol{b}$. Siten $A\,\boldsymbol{x} =$

$$\begin{bmatrix} 1 & -2 & 1 \\ 0 & 2 & -8 \\ -4 & 5 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 - 2\,x_2 + x_3 \\ 2\,x_2 - 8\,x_3 \\ -4\,x_1 + 5\,x_2 + 9\,x_3 \end{bmatrix}$$

Toisin sanoen: tulo $A\,\boldsymbol{x}$ on (pysty)vektori, jonka komponentit saadaan sisätuloina $\boldsymbol{a}_1 \cdot \boldsymbol{x}, \boldsymbol{a}_2 \cdot \boldsymbol{x}, \boldsymbol{a}_3 \cdot \boldsymbol{x}$, kun matriisin A vaakavektoreita merkitään: $\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3$. Matriisin rivin on oltava yhtä pitkä kuin kerrottava sarakevektori, ts. matriisin sarakkeiden lukumäärän ja kerrottavan vektorin pituuden on oltava samoja. Tulosvektorin pituus on sama kuin matriisin sarakkeen pituus (= rivien lukumäärä).

Tähän saakka esittämämme määritelmät näyttävät antavan meille ainakin esteettistä nautintoa, yhtälöryhmämme saa muodollisesti hyvin yksinkertaisen asun: $A\,\boldsymbol{x} = \boldsymbol{b}$.

Mikään ei estäisi käyttämästä analogiaa tavallisen ensimmäisen asteen yhtälön ratkaisulle ja merkitsemästä yhtälöryhmän ratkaisua jakolaskuun viittaavilla tavoilla:

$$\boldsymbol{x} = \frac{\boldsymbol{b}}{A} = A^{-1}\boldsymbol{b}.$$

Jakolaskumuoto ei ole yleisessä käytössä matemaattisena merkintätapana, sensijaan jälkimmäinen muoto, jossa ratkaisu esitetään "käänteismatriisilla" kertomisena, on matriisilaskennan arkipäivää. Mikään ei toki estä käyttämästä jakolaskuasymboliikkaa ainakaan tietokoneohjelmassa. Näin onkin tehty mm. MATLAB/SCILAB:ssa:

Esimerkkiyhtälöryhmämme ratkaistaisiin näin:

```
-->A=[1 -2 1; 0 2 -8;-4 5 9]
 A  =
    1.  - 2.    1.
    0.    2.  - 8.
  - 4.    5.    9.

-->b=[0;8;-9]
 b  =
    0.
    8.
  - 9.

-->x=A\b // "matriisilla A jako"
 x  =
    29.
    16.
    3.
// Tarkista kertomalla: A*x, antaako b:n ?
```

Jos kyseessä olisi lukuja (1×1-matriiseja) koskeva yhtälö, olisi yhtälön ratkaisun eli jakolaskun onnistumisen ehtona $A \neq 0$. Yhtälöryhmän tapauksessa matriisia A koskeva ehto on juuri se, johon Gaussin eliminaatiomenettely johtaa. Ehto voidaan lausua monessa muodossa, joiden esittelyyn emme tässä ryhdy.

Matriisitulo yleisesti

Osaamme muodostaa tulon $A\,\boldsymbol{b}$, kun $\boldsymbol{b}$ on vektori, jonka pituus [3] on sama kuin matriisin rivin pituus (ts. sarakkeiden lukumäärä). Jos vaikka A on 2×3-matriisi ja $\boldsymbol{b}$ on $\mathbb{R}^3$:n vektori, niin

$$A\,\boldsymbol{b} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{a}_1 \cdot \boldsymbol{b} \\ \boldsymbol{a}_2 \cdot \boldsymbol{b} \\ \boldsymbol{a}_3 \cdot \boldsymbol{b} \end{bmatrix},$$

missä $\boldsymbol{a}_i$ tarkoittaa A-matriisin rivivektoria numero i.

Matriisitulo $A\,B$ voidaan nyt määritellä soveltamalla tuloa $A\,\boldsymbol{b}$ kuhunkin B-matriisin sarakkeseen ja latomalla näin saadut sarakevektorit vierekkäin. Matriisin A rivin on oltava samanpituinen kuin B:n sarake. Jos A on $m \times n$ ja B on $n \times p$, niin $C = A\,B$ on $m \times p$. Erityisesti neliömatriiseja voidaan aina kertoa keskenään ja tuloksena on samankokoinen neliömatriisi.

[2] Vektorikäsitteen kannalta on yhdentekevää, kirjoitetaanko koordinaatit pysty- tai vaakasuoraan (tai vaikka S:n muotoon tms.), kunhan komponenttien järjestys on selvä. Matriisilaskennan kannalta on huomattavaa laskentateknistä hyötyä apukäsitteistä pysty- ja vaakavektori.

[3] Matriisilaskennassa puhutaan usein vektorin "pituudesta", kun oikeasti tarkoitetaan dimensiota.

Matriisialgebraa

Samankokoisille matriiseille voidaan määritellä yhteenlasku vastinalkioittain aivan samoin kuin vektoreilla. Samoin määritellään luvun c ja matriisin tulo kertomalla kukin matriisin alkio c:llä. Samat laskusäännöt ovat voimassa. (Itse asiassa $(m \times n)$-matriiseja voidaan näitä laskutoimituksia ajatellen pitää $m\,n$-ulotteisina vektoreina.)

Kun otamme kertolaskun mukaan, puhumme jatkossa yksinkertaisuuden vuoksi vain neliömatriiseista, jolloin matriisit ovat aina kertomiskelpoisia.

Kaikki vektorilaskennasta tutut laskusäännöt pätevät, mutta miten toimivat laskusäännöt, kun mukaan otetaan matriisikertolasku. Osoittautuu, että yhtä poikkeusta lukuunotamatta kaikki tutut algebran säännöt ovat voimassa. Niinpä liitännäisyys ja osittelulait pätevät. Siis $A(B\,C) = (A\,B)C$, $A(B + C) = A\,B + A\,C$, jne. Todistukset seuraavat matriisitulon määritelmästä suoraviivaisesti, mutta eivät ole samalla tavoin itsestäänselvyyksiä kuin vektorien laskuominaisuudet.

Entä päteekö vaihdannaisuus, eli onko $A\,B = B\,A$?

Tämä voidaan heti osoittaa vastaesimerkillä vääräksi. Voit ottaa melkein mitkä tahansa, vaikkapa 2×2-matriisit ja kertoa ne keskenään molemminpäin. Huomaat, että saat eri tulokset. Kokeile vaikka:

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

$$A\,B = \begin{bmatrix} 4 & -4 \\ 3 & -3 \end{bmatrix}, \quad B\,A = \begin{bmatrix} 3 & -2 \\ 3 & -2 \end{bmatrix}.$$

Suorita laskut käsin! Kokeile myös SCILAB:lla, siinä kertomerkki $(*)$ tarkoittaa juuri matriisikertolaskua. kuten edellä jo näimme.

Kenties elämäsi ensimmäisen kerran olet tekemisissä sellaisen laskentajärjestelmän kanssa, jossa kertolasku ei ole vaihdannainen.

Neliömatriisille A voidaan määritellä **potenssi** aivan kuten luvuille:

$$A^p = \underbrace{A\,A \cdots A}_{p \text{ kertaa}}.$$

Tehtävä: Päteekö neliömatriiseille binomin neliön kaava? Ellei, niin mihin muotoon on kirjoitettava:
$(A + B)^2 = A^2 + B^2 + \underline{\quad}$?

Dynaamisia systeemejä, Markovin prosesseja

Monilla sovellusalueilla, kuten ekologiassa, kansantaloudessa, erilaisissa insinööritieteissä mallinnetaan ajan mukana muuttuvaa systeemiä. Systeemin tilaa ajanhetkellä t_k kuvatkoon vektori $\boldsymbol{x}_k$. Puhutaan diskreetistä dynaamisesta systeemistä. Yksinkertaisinta tyyppiä on lineaarinen systeemi, jossa lähdetään alkutilasta $\boldsymbol{x}_0$ ja seuraavaan tilaan päästään kertomalla neliömatrisiilla A. Tällöin $x_1 = A\,\boldsymbol{x}_0, \boldsymbol{x}_2 = A\,\boldsymbol{x}_1, \ldots$, yleisesti:

$$\boldsymbol{x}_{k+1} = A\,\boldsymbol{x}_k, k = 0, 1, 2, \ldots$$

Kuten heti nähdään, tehtävälle voidaan kirjoittaa ratkaisukaava: $\boldsymbol{x}_k = A^k \boldsymbol{x}_0$. Tässä yksi syy tarpeeseen oppia keinot, joilla matriisipotensseja voidaan tehokkaasti laskea.

Tarkastelemme erityistä tyyppiä olevia matriiseja, ns. *stokastisia matriiseja*, joita myös *Markovin matriiseiksi* kutsutaan. Näissä kaikki alkiot ovat ei-negatiivisia $(a_{ij} \geq 0)$ ja sarakesummat ovat $= 1$. Tällaisen matriisin määräämää dynaamista systeemiä kutsutaan *Markovin prosessiksi*.

Eduskuntavaalit

Aiheen ajankohtaisuuden vuoksi valitsen esimerkin, jossa joudun sotkeutumaan puoluepolitiikkaan. Aineistona käytän HS:n sunnuntaina 25.2.07 s. A4 julkaisemia, Suomen Gallupin keräämiä puolueiden kannatuslukemia. Lehdessä on julkaistu 9×9-matriisi, jossa riveillä on puolueiden kannatusprosentit. Matriisin sarakkeet edustavat kannatusmittauksia keskimäärin noin $2 : n$ kuukauden välein alkaen helmi-maaliskuusta 2006. (Sarake 1, Eduskuntavaalien 2003 tulos, jätetään pois.) Tehtävän yksinkertaistamiseksi tarkastelen vain kolmen suurimman puolueen keskinäisiä kannatusosuuksia. Annetusta matriisista valitaan siten 3 ensimmäistä riviä ja kukin sarake jaetaan sarakesummalla. (Voidaan tehdä hyvin elegantilla matriisilausekkeella MATLAB/SCILAB:ssa.)

Näin meillä on Gallup-matriisi G:

$$\begin{bmatrix} 0.320 & 0.325 & 0.325 & 0.324 & 0.317 & 0.314 & 0.311 \\ 0.313 & 0.316 & 0.327 & 0.331 & 0.337 & 0.341 & 0.343 \\ 0.367 & 0.359 & 0.348 & 0.344 & 0.346 & 0.344 & 0.346 \end{bmatrix}.$$

Rivit ovat järjestyksessä (Kok, Ke, Sd) ja ajat $2/2006 - -2/2007$.

Kysymykset 1. Voidaanko prosessia kuvata Markovin matriisilla A ?
2. Jos voidaan, niin mistä tuo matriisi saadaan?
Ohitetaan toistaiseksi nämä kysymykset ja tempaistaan ns. "hatusta" tällainen Markovin matriisi:

$$A = \begin{bmatrix} 0.93 & 0.02 & 0.04 \\ 0.03 & 0.93 & 0.04 \\ 0.04 & 0.05 & 0.92 \end{bmatrix}$$

Olkoon $\boldsymbol{x}_0 = [k_0, c_0, d_0]^T$, k=Kokoomus, c=Keskusta ("center"), d=Demarit. Nyt

$$A\,\boldsymbol{x}_0 = \begin{bmatrix} 0.93\,k_0 + 0.02\,c_0 + 0.04\,d_0 \\ 0.03\,k_0 + 0.93\,c_0 + 0.04\,d_0 \\ 0.04\,k_0 + 0.05\,c_0 + 0.92\,d_0 \end{bmatrix} = \boldsymbol{x}_1 = \begin{bmatrix} k_1 \\ c_1 \\ d_1 \end{bmatrix}$$

Rivi 1 sanoo: 93% Kok-kannattajista pysyy, 2% Ke-kannattjista ja 4% Sd-kannatajista siirtyy Kok-kannattajaksi. Aivan vastaavasti muut. Siten A-matriisin sarakkeet ilmaisevat sijamuotoa elatiivi (sta/stä) ja rivit illatiivia (een,uun,hin). Huomaa, että sarakesummat ovat 1 (Jokaisesta siirrytään jonnekin, mahdollisesti samaan.), mutta rivisummien ei tarvitse olla 1, voisihan joku puolue ääritapauksessa vaikka romahtaa totaalisesti ja päätyä 0-kannatukseen. Tämä näkyisi matriisissa ao. puolueen kohdalla nollarivinä. Matriisimme on varsin "lävistäjävaltainen", mikä merkitsee voimakasta puolueuskollisuutta.

Näin jatkaen johdumme Markovin prosessiin

$$\boldsymbol{x}_{k+1} = A\,\boldsymbol{x}_k, k = 0, 1, 2, \ldots$$

Matriisin A "tempaamisesta" todettakoon, että siinä on otettu huomioon lävistäjävaltaisuus ja lisäksi tehty kokeiluja ja säätöjä aineistoon nojautuen. Matriisia ei voida ratkaista aineiston perusteella saatavissa olevasta 9×9- yhtälöryhmästä, tämä ei yleensä tuottaisi Markovin matriisia.

Vektoria $\boldsymbol{x}_0$ voimme kutsua todennäköisyysvektoriksi: alkiot ovat ei-negatiivisia ja niiden summa = 1.

Todetaanpa tärkeä perusominaisuus yleiselle $(n \times n)$ Markovin matriisille: Jos $\boldsymbol{x}_0$ on todennäköisyysvektori, niin samoin ovat $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$.
Perustelu:
1. Koska kukin $a_{ij} \geq 0$, ja kukin $x_i \geq 0$, lasketaan matriisi kertaa vektori-tulossa ei-negatiivisia termejä yhteen ja tulos ≥ 0.
2. Vektorin $\boldsymbol{v}$ alkioiden summa on sama kuin sisätulo (eli matriisitulo) $[1, 1, \ldots, 1]\,\boldsymbol{v}$ (Ajatellaan $\boldsymbol{v}$ pystyvektorina, kuten yleensäkin.) Markovin matriisin sarakesummat = 1, joten $[1, 1, \ldots, 1]\,A = [1, 1, \ldots, 1]$, ja niinpä $\boldsymbol{x}_1$:n alkioiden summa = $[1, 1, \ldots, 1]\,(A\,\boldsymbol{x}_0) = ([1, 1, \ldots, 1]\,A)\,\boldsymbol{x}_0 = [1, 1, \ldots, 1]\,\boldsymbol{x}_0 = 1$.
Tässä käytimme matriisitulon liitännäisyyttä.

Mitäpä tapahtuu kelpo puolueillemme, saavatko selvää toisistaan?

Lähdetään iteroimaan helmikuusta 2006, ts. valitaan $\boldsymbol{x}_0$ = G-matriisin ensimmäinen sarake: Iteroidaan ihan vaan käsin kirjoitellen ohjelmallemme:

```
--> A=[.93 .02 .04;.03 .93 .04;.04 .05 .92];
--> x0=G(:,1)  // G:n 1. sarake
--> x1=A*x0; x2=A*x1; x3=A*x2; x4=A*x3;
--> x5=A*x4;x6=A*x5;x7=A*x6;
--> [x0 x2 x4 x7]

    '2/06'   '6/06   '10/06'  '2/07'
   0.320    0.317    0.315    0.312  'Kok'
   0.313    0.317    0.321    0.325  'Kepu'
   0.367    0.365    0.364    0.363  'Sdp'
```

Kun verrataan viimeistä saraketta yllä annetun G-matriisin viimeiseen sarakkeeseen, nähdään, että Kokoomus on varsin tarkkaan kohdallaan, Kepu:n ja Sdp:n välillä mallinnuksemme on hiukan liikaa kallellaan jälkimmäiseen päin, tosin ajan kuluessa tuo näyttäisi hitaasti korjaantuvan. No, koska matrisiimme näinkin hyvin ennustaa "oikeita" ennusteita, jatkamme eteenpäin mallia (A-matriisia) muuttamatta.

Miten sitten käy vaaleissa? Tällä mallilla ja aineistolla on turha toivoa saatavan mitään parempaa kuin G-matriisin viimeinen sarake. Joudun tässä suhteessa tuottamaan lukijalle pettymyksen, nuo mainiot matriisitkaan eivät auta meitä ennustamaan tulosta. Sensijaan voidaan kysyä, mitä tapahtuu pitkällä aikavälillä. Jatketaanpa iterointia, itse asiassa se on hyvin helppoa, koska $\boldsymbol{x}_n = A^n\boldsymbol{x}_0$ ja matriisipotenssi ohjelmissamme on `A^n`

```
-->[x0 (A^50)*x0 (A^70)*x0 (A^80)*x0]
 ans  =
    0.3201    0.3027    0.3025    0.3025
    0.3129    0.3360    0.3361    0.3361
    0.367     0.3613    0.3613    0.3613
```

Huomataan, että viimeistään 70:n aikajakson jälkeen, eli n. 10 vuoden kuluttua prosessi on päätynyt kiintopisteeseen, josta se ei enää etene. Entä mitä vaikuttaa alkujakauma. Kokeillaan vaikka `x0=[.6 .3 .1]';`

```
-->[x0 (A^10)*x0 (A^30)*x0 (A^50)*x0 (A^80)*x0]
 ans =
    0.6    0.3989    0.3133    0.3038    0.3026
    0.3    0.3143    0.3318    0.3355    0.3361
    0.1    0.2868    0.3549    0.3607    0.3613


->[x0 (A^10)*x0 (A^30)*x0 (A^50)*x0 (A^100)*x0]
 ans =
  0.6    0.3989    0.3133    0.3038    0.3025
  0.3    0.3143    0.3318    0.3355    0.3361
  0.1    0.2868    0.3549    0.3607    0.3613
```

Huomataan, että aivan "vinossa olevasta" alkujakaumasta päädytään rajalla **samaan tasapainojakaumaan**. Toden totta, Markovin prosessi johtaa (tietyin yleisin lisäehdoin) **samaan kiintopisteeseen**, olipa

alkujakauma mikä tahansa. Mutta miksi? Perustelu voidaan esittää hyvin tyylipuhtaasti ominaisarvoteorian sovelluksena. Sen esittely on pakko jättää johonkin myöhempään yhteyteen. Aihetta voit opiskella alla mainituista lähteistä sekä Googlella hakusanoilla *ominaisarvot, eigenvalues, eigenvectors*. Kerromme kuitenkin tuloksen: Rajajakauma on suurinta ominaisarvoa (1) vastaava ominaisvektori todennäköisyysvektoriksi normeerattuna. Annan lopuksi vaihteeksi aidon MATLAB-ajon, jossa on tätä matriisiamme koskevat ominaisarvo/-vektoriloitsut, jotka paljastavat taikuuden:

```
>>  A=[.93 .02 .04;.03 .93 .04;.04 .05 .92];
>> [V,D]=eig(A);
>> lambda1=D(1,1);v1=V(:,1);
>> lambda1
lambda1 =
    1.0000
>> v1/sum(v1) % ominaisvektorin normeeraus.
    0.3025
    0.3361
    0.3613
```

Viitteet

1. David C. Lay *Linear Algebra and its applications*, 3^{rd} ed., Addison Wesley, 2003.

2. Gilbert Strang *Introduction to Linear Algebra*, 3^{rd} ed., Wellesley-Cambridge Press, 2003.
   Luentovideot MIT:n sivulla:
   `http://ocw.mit.edu/OcwWeb/Mathematics/18-06Spring-2005/VideoLectures/index.htm`

3. Wikipedia: Historiaa ja yleiskatsaus runsaine viitteineen
   `http://en.wikipedia.org/wiki/Linear_algebra`

4. Heikki Apiola
   Luentoja TKK:lla syksyllä 2006
   `http://math.tkk.fi/opetus/kp3-ii/06/L/`

5. Matlab/Scilab-kotisivut
   `http://www.mathworks.com/`
   `http://www.scilab.org/`

5. Heikki Apiola–Marko Laine
   Lyhyt Matlab-opas
   `http://math.tkk.fi/~apiola/matlab/opas/lyhyt/`

6. Google-hakusanoja:
   matriisi, matrix, vector, lineaarialgebra, linear algebra.