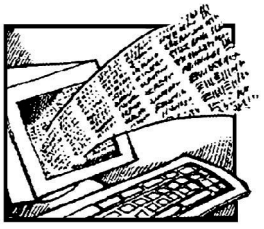


Interpolaatio, käyrän sovitus ja lukujonot

Heikki Apiola
Aalto-yliopisto, matematiikan ja systeemianalyysin laitos, lehtori emeritus
heikki.apiola@aalto.fi

Kirjoituksen lähtökohtana on Solmun numerossa 1/2019 julkaistu *Antti Laaksosen* artikkeli ”Kuinka löytää kaava lukujonolle”.

Laaksonen esittelee joitakin kokonaislukujonoja ja kysyy, miten voidaan löytää funktio, jonka arvoja jonon luvut ovat. Toki kysymys ei yleisessä muodossa ole mielekäs, onhan selvää, että ratkaisuja on ääretön määrä. Etsittävien funktioiden joukkoa tulee siis rajoittaa. Luonnollisin funktioluokka on varmasti *polynomit*, joihin kirjoittaja keskittyy.

Kysymys johtaa näin polynomi-interpolaatioon. Tuossa kirjoituksessa käsitellään asiaa laskemalla data-arvojen erotuksia. Itse asiassa tätä tekniikkaa käytetään *Newtonin interpolaatiomenetelmässä*, johon tämän kirjoituksen interpolaatiotekniikoiden esittely tietyllä tavalla huipentuu.

Noin 15 vuotta sitten kirjoitin Solmuun interpoloinnista varsin seikkaperäisesti (viite [s04]). Kertaan tiiviisti tuossa kirjoituksessa varsin yksityiskohtaisesti esitettyjä teemoja, ja täydennän esitystä menetelmillä, joita tuolloin en ottanut mukaan. Esitykseni ei toki edellytä aiemman kirjoitukseni lukemista/muistamista. (Voihan olla, ettet ollut edes syntynyt sen ilmestymisen aikaan.)

Tarkastelen lopuksi näiden tekniikoiden sovelluksena *Antti Laaksosen* kirjoituksen hengessä eräiden muidenkin lukujonojen, erityisesti sarjojen osasummien saattamista ns. ”suljettuun muotoon” interpolaatiotekniikkoja käyttäen. Tässä yhteydessä otan käyttöön symbolilaskennan ohjelmistoja. Esitän lopuksi esimerkin **tietokoneavusteisesta induktiotodistuksesta** tietyn osasummalausekkeen yleispätevyyden osoittamiseksi, ja ehdotan joitain tähän liittyviä harjoitustehtäviä lukijalle.

Varoitus: Teksti saattaa joillakin kohdin edetä hiukan vauhdikkaasti ja sisältää kenties lukijalle uusia käsitteitä ja merkintätapoja: *matriisi, summa- ja tulosymbolit:*

$$\sum_{k=1}^{n} x_k = x_1 + x_2 + \ldots + x_n, \quad \prod_{k=1}^{n} x_k = x_1 \cdot x_2 \cdots x_n.$$

Voit huoletta jatkaa lukemista, vaikka jokin yksityiskohta ei heti avautuisikaan.

Ohjelmointia: Kirjoituksen henki on sellainen, että esitetyt kaavat ja algoritmit pyritään saattamaan korkean tason ohjelmointikieliä käyttäen mahdollisimman läpinäkyvästi koneella suoritettavaan muotoon. Suurin osa ohjelmakoodeista on MATLAB-kieltä, lähes kaikki esimerkit voi toteuttaa OCTAVE:lla, joka on vapaasti saatava MATLAB-”klooni”. Lisäksi käytän symbolilaskentaohjelmia, vapaasti käytettävää MAXIMA:a, MATLAB:n symbolic toolboxia ja MAPLE:a. Kaksi viimeksi mainittua sisältyy yleensä oppilaitosten kampuslisensseihin.

Yllytän **vuorovaikutteiseen lukutapaan**. Vaikka et koodin yksityiskohtia heti ymmärrä, voit sijoittaa koodeja ohjelman komentoikkunaan tai editoriin ja kokeilla ja tehdä omia muunnelmia.

Ohjelmisto-ohjeita ja viitteitä

Kirjoituksessa esiintyy edellä mainittuihin ohjelmistoihin/kieliin liittyviä käyttöesimerkkejä. Suurin osa on MATLAB-kieltä, jota OCTAVE uskollisesti noudattaa.

1. - https://www.gnu.org/software/octave/
   - https://octave-online.net/ Ei tarvitse asentaa, rekisteröidy, niin voit käyttää ”skriptejä”, eli Octave-komentoja sisältäviä tekstitiedostoja, ns. m-tiedostoja.
   - LINUX (Ubuntu)-käyttäjälle asennus on vaivattominta: Komentoikkunaan kirjoitetaan:
   `sudo apt-get install octave`.
2. https://math.aalto.fi/∼apiola/matlab/opas/lyhyt Kaipaa päivittelyä, mutta päässee alkuun.
3. http://maxima.sourceforge.net/
4. http://www.wolfram.com/ Wolfram Alpha, MATHEMATICA:n tekijän, Steven Wolframin kehittynyt symbolilaskin.

Neljän kohdan pikaohje

OCTAVE:n käyttöliittymä: Kaksi pääikkunaa: *komentoikkuna* ja *editori*.

1. Voit kirjoittaa suoraan komentoikkunaan tyyliin: `>> komento` tai voit kirjoittaa editori-ikkunaan ja suorittaa komennot sieltä.
2. MATLAB-nimi viittaa termiin ”Matrix Laboratory”, siksi erityisesti kertolasku, jakolasku ja potenssiin korotus tarkoittavat matriisilaskutoimituksia. Niinpä vaakavektorin `v` ja samanpituisen pystyvektorin `p` tulo `v*p` tarkoittaa vektorien ”sisätuloa” eli vastinalkioiden tulojen summaa: $v_1p_1 + v_2p_2 + \ldots + v_np_n$.
3. *Alkioittaiset, pisteittäiset laskutoimitukset*: Usein tarvitaan esim. vektorien u ja v vastinalkioittaista tuloa `u.*v`, joka siis on vektori $[u_1v_1, u_2v_2, \ldots, u_nv_n]$. Vastaavasti pisteittäinen jakolasku ja potenssi saadaan liittämällä piste (.) operaatiomerkin eteen. Siis `.*  .^  ./`
4. Vektorin rakentaminen:
   vaakavektori: `>> v=[1 2 3]` tai `v=1:3`
   pystyvektori: `>> p=[1;2;3]` tai `p=v'`

Näillä ja esimerkkikoodien kommenteilla pääsee toivottavasti eteenpäin. **Huomaa:** Tässä tekstissä oleva heittomerkki `(')` ei ”copypastessa” tulkkaudu oikein, se pitää kirjoittaa itse, mikä muutenkin on suositeltavaa.

Jos et ole aiemmin ohjelmia käyttänyt, kohtaat todennäköisesti alkuunpääsyongelmia. Suositeltava foorumi kysymyksille on **Facebook-ryhmä ”Matematiikkalehti Solmu”**.

Liikkeellelähtö

Aloitetaan keväisellä kysymyksellä: Mikä on seuraava luku jonossa 1995, 2011, 2019?

Luonnollinen lähtökohta on etsiä polynomia, joka kulkee annettujen datapisteiden kautta, ts. *interpolaatiopolynomia* datapisteille $(0, 1995), (1, 2011), (2, 2019)$. Luonnollinen ratkaisu on kirjoittaa yhtälöryhmä polynomin $p(x) = a_0 + a_1x + a_2x^2$ tuntemattomien kertoimien a_0, a_1, a_2 määräämiseksi, kun vaaditaan ehdot: $p(0) = 1995, p(1) = 2011, p(2) = 2019$. Tässäpä sopiva pieni harjoitus lukijalle, kun vielä a_0 saadaan ilmaiseksi.

Esitän toisen laskutekniikan, jossa lasketaan erotuksilla. Tekniikka on helppo oppia mekaanisesti, palataan perusteluun *Newtonin interpolaatiomenetelmän* yhteydessä.

0	y_0		
1	y_1	$\frac{y_1-y_0}{1-0} = 16$	
2	y_2	$\frac{y_2-y_1}{2-1} = 8$	$\frac{8-16}{2-0} = -4$

Tämä merkitsee, että interpolaatiopolynomi on

$$p(x) = 1995 + 16\,x - 4x(x-1) = 1995 + 20x - 4x^2.$$

Kyseessä on alaspäin aukeava paraabeli, ja kaiken lisäksi $p(3) = 2019$, ja tuosta arvot vain pienenevät. Koska y-arvojen on oltava aidosti kasvava jono vuosilukuja, ei mallinnus interpolaatiopolynomilla toimi tälle datalle. Lasketaanpa nyt harjoittelun vuoksi OCTAVE:lla erotuksia:

```
>> yd=[1995 2011 2019];
>> d1=diff(yd) % yd-vektorin erotukset
d1 =
   16    8
>> d2=diff(d1) % Toiset erotukset
d2 = -8
```

`d2` edustaa diskreettiä 2. derivaattaa, sen negatiivisuus merkitsee siis ylöspäin kuperuutta.

Datan soveltumattomuus johtuu siis siitä, että 2. differenssi < 0, eli 1. differenssit pienenevät. Tässä erotukset sattuvat jopa puolittumaan aikaväleillä. Jos näin jatkuu, niin seuraava tapaus olisi jo 2023. Jääkiekkoväkeä tämä ehkä tyydyttäisi enemmän kuin interpolaatiomahdollisuus.

Voit ottaa ensituntumaa OCTAVE:en asentamalla ja kirjoittamalla suoraan komentoikkunaan. Tai kirjautumalla sivulle OCTAVE online, kuten yllä neuvotaan. Yllä käytetty komento `diff` on erittäin hyödyllinen työkalu, jolla voi myös laskea numeerisia derivaattoja.

Shakkiongelma

Antti Laaksonen käsittelee kysymystä: ”Miten monella tavalla voidaan $n \times n-$shakkilaudalla sijoittaa kaksi ratsua niin, etteivät ne uhkaa toisiaan?” Tapaukset $n = 1$ ja $n = 2$ antavat selvästi mahdollisuuksia 0 ja 6. Kirjoittaja perustelee tapauksen $n = 3$ luettelemalla systemaattisesti kaikki mahdollisuudet, josta nähdään tulos: 28. Loppuosan jonosta hän antaa viitaten tietokonelaskelmiin. Mahdollisuus tämän datan esittämiseen lukumääräänsä selvästi alempiasteisena polynomina antaa hypoteesin jonon jatkumisesta tätä polynomifunktiota noudattaen.

Laaksosen kirjoituksessa lasketaan datavektorin peräkkäisiä erotuksia. Katsotaanpa, miten toimittaisiin Octave:lla, jossa Matlab:n mallin mukaan funktiot operoivat yleensä kokonaisiin vektoreihin. Tässä tarjoutuu itsestään selvästi edellä esitelty funktio `diff`:

```
s=[0; 6; 28; 96; 252; 550; 1056; 1848];
d0=s;
d1=diff(d0);% y-datan 1. erotukset
d2=diff(d1);% 2. erotukset
d3=diff(d2);% 3. erotukset
d4=diff(d3) % 4. erotukset (näytetään)
d5=diff(d4) % 5. erotukset (tietysti)
```

Rakennettaessa erotusten taulukkoa tarvitaan täyttöalkioita, koska `diff` lyhentää vektoria 1:llä. Tällaiseksi tarjoutuu symboli `NaN`, ”Not a Number”. Octave:ssa laskutoimitus 0/0 ei aiheuta ohjelman keskeytymistä virheeseen, vaan tuottaa tuloksen `NaN`, jota voidaan mukavasti hyödyntää. Alla näkyy taulukon rakentaminen latomalla sarakkeet toistensa viereen.

```
taulukko=[d0,[NaN;d1],[NaN;NaN;d2],...
[NaN;NaN;NaN;d3],[NaN;NaN;NaN;NaN;d4],...
[NaN;NaN;NaN;NaN;NaN;d5]]
  d4 =  % Tilan säästämiseksi rivivektorina
    12    12    12    12
  d5 =    % ... samoin
     0     0     0
taulukko =
  0      NaN     NaN     NaN     NaN     NaN
  6        6     NaN     NaN     NaN     NaN
  28      22      16     NaN     NaN     NaN
  96      68      46      30     NaN     NaN
 252     156      88      42      12     NaN
 550     298     142      54      12       0
 1056    506     208      66      12       0
 1848    792     286      78      12       0
```

Siis 4. erotukset ovat vakioita, joten 5. erotukset = 0. Kuten Laaksonen toteaa, tästä voidaan päätellä, että koko annettu data voidaan esittää astetta 4 olevalla polynomilla.

Kirjoituksen lopulla palataan tähän differenssitaulukkoon.

Yleinen interpolaatiotehtävä

Olkoon annettu taulukko

x_0	x_1	x_2	$\dots$	x_n
y_0	y_1	y_2	$\dots$	y_n

Taulukko 1

Voidaan ajatella, että kyse on annetun funktion taulukoiduista arvoista pisteissä $x_0, x_1, \dots, x_n$. Toisaalta taulukko voi edustaa johonkin havaintoaineistoon tai kokeeseen liittyviä mittaustuloksia, kuten lämpötilaa mitattuna vaikkapa tunnin välein, ym.

Jos tiedetään, että luvut edustavat jonkin ”sileän”[1] funktion arvoja hyvällä tarkkuudella lasketuna/mitattuna, voi olla järkevää asettaa tehtäväksi määrittää johonkin sopivaan funktioluokkaan kuuluva funktio, jonka kuvaaja kulkee tarkalleen kaikkien annettujen pisteiden kautta. Tällöin puhutaan *interpolaatiosta.*

Toisaalta, jos mittaukset ovat epätarkkoja, tai kaikkien pisteiden kautta kulkemisen vaatimus on muuten tilanteeseen sopimaton, voidaan etsiä aineistoon liittyvää pääsuuntaa, ”trendiä” sopivan optimaalisuuskriteerin suhteen. Näistä eniten käytetty on ns. **pienimmän neliösumman** (PNS) approksimaatio. Tällöin puhutaan interpolaatiota yleisemmin käyrän sovittamisesta aineistoon.

Lasketaan esimerkki Matlabin/Octaven valmiilla työkaluilla

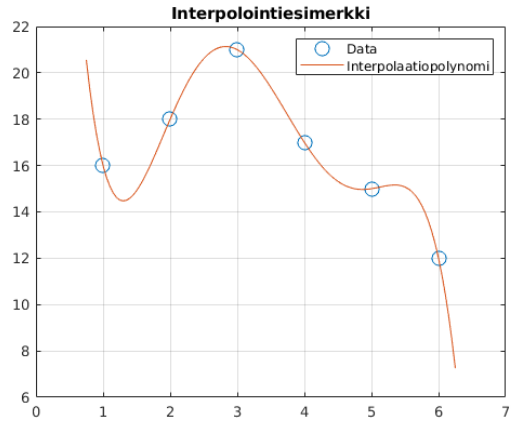
Matlab:n ja Octave:n käyttöympäristön pääosat ovat *komentoikkuna* ja *editori.* Komentoja voidaan syöttää suoraan komentoikkunaan tai ne voidaan kirjoittaa editorilla komento- eli skriptitiedostoon, vaikkapa `komennot.m` ja suorittaa kirjoittamalla komentoikkunaan: `>> komennot`

Seuraava istunto on tehty Octave:lla. Kehotetta (`>>`) seuraavat rivit ovat ohjelman komentoja ja muut komennon antamia tulosteita. Tuloste estetään puolipisteellä (;) komennon perässä.

```
>> xd=1:6  % xdata: [1,2,3,4,5,6]
xd =
   1   2   3   4   5   6
>> yd=[16 18 21 17 15 12]  % ydata
yd =
   16   18   21   17   15   12
>> N=length(xd);          % vektorin pituus
>> c=polyfit(xd,yd,N-1)
```

[1] Sileydellä tarkoitamme ao. sovelluksen kannalta riittävän monen derivaatan olemassaoloa.

```
   % Interpolaatiopolynomin kertoimet
c =
    -0.24167     4.33333   -28.95833
     87.66667   -115.80000    69.00000
>> x=.75:.05:6.25;  % Pisteet,joissa polynomin
                    % arvot lasketaan.
>> p=polyval(c,x);  % Polynomin arvot
                    % x-pisteissä (HUOM (;))
>> plot(xd,yd,'o',x,p); grid on
>> legend('Data','Interpolaatiopolynomi')
>> title('Interpolointiesimerkki')
```



Polynomi esitetään MATLAB:ssa kertoimien vektorina alkaen korkeimman asteen kertoimesta. Siten polynomimme on (desimaaleja karsien):

$$p(x) = -0.24x^5+4.33x^4-28.95x^3+87.66x^2-115.8x+69.$$

Huomaa, että komennossa `>> c=polyfit(xd,yd,N-1)` on `N-1` yhtä pienempi kuin datapisteiden lukumäärä, jolloin kyse on interpolaatiosta. (Kahden pisteen kautta suora, kolmen pisteen kautta paraabeli jne.)

Näemme, että tehtävä on kuvan tarkkuudella oikein ratkaistu, koska polynomimme kulkee kaikkien datapisteiden kautta.

Tarkkaavainen lukija saattaa ihmetellä, miksei `polyfit`-koodiin ole upotettu interpolaatiotehtävän datan määräämää parametriä `N-1` sisäisellä komennolla `N = length(xdata);` No, koska `polyfit` on kirjoitettu niin yleispäteväksi, että sillä voidaan laskea myös interpolaatiopolynomia alempiasteisia *pienimmän neliösumman (PNS)* polynomisovituksia kutsumalla funktiota `(N-1)`:tä pienemmillä arvoilla.

Interpolaatiopolynomin muodostaminen

Edellä laskettiin polynomin kertoimet ja arvot "mustalla laatikolla" `polyfit`. Käsittelen nyt erilaisia tapoja näiden laskentaan.

Lineaarinen yhtälöryhmä

Puetaan yleiseen muotoon alkuesimerkin yhtälöryhmän muodostus. Lähdetään liikkeelle taulukon 1 yleisestä datasta, jossa oletetaan vain, että x_k-pisteet ovat erillisiä. Etsitään polynomia:

$$p(x) = a_0 + a_1x + \ldots + a_nx^n.$$

Otetaan tässä yleisemmin kirjallisuudessa esiintyvä muoto kertoimien järjestyksen suhteen. Vaatimus:

$$p(x_0) = y_0, p(x_1) = y_1, \ldots, p(x_n) = y_n.$$

Toisin sanoen:

$$\begin{cases} a_0 + a_1x_0 + a_2x_0^2 + \ldots + a_nx_0^n = y_0 \\ a_0 + a_1x_1 + a_2x_1^2 + \ldots + a_nx_1^n = y_1 \\ \vdots \\ a_0 + a_1x_n + a_2x_n^2 + \ldots + a_nx_n^n = y_n \end{cases}$$

Ratkaistavana on siten $n+1$ yhtälöä ja $n+1$ tuntematonta (kertoimet $a_0, a_1, \ldots, a_n$) käsittävä lineaarinen yhtälöryhmä.

Matriisimuodossa:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \ddots & \vdots & \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$$

Jos matriisilaskenta ei ole tuttua, niin ylempi muoto näyttää, mitä matriisi kertaa pystyvektori tarkoittaa. Samoin kuin polynomin kerroinvektori määrää polynomin, määräävät matriisi ja oikean puolen y-vektori yhtälösysteemin.

Ratkaistaan nyt edellä käsitelty esimerkki. Tehtävä palautuu siis lineaarisen yhtälöryhmän ratkaisuun, johon OCTAVE:ssa on suora komento: `a=V\y`, missä `V`:llä merkitään yllä olevaa matriisia. Tätä muotoa olevaa matriisia kutsutaan *Vandermonden* matriisiksi, jonka määrää x-datavektori: $x = [x_0, x_1, \ldots, x_n]$. Yhtälöryhmän ratkaisukomennossa `a=V\y` voidaan ajatella, että takakeno (\) tarkoittaa jakoviivaa, nimittäjässä on matriisi `V`, ja siten vektori `y` "jaetaan matriisilla" `V`. Kootaan komennot skriptitiedostoon *vanderinterp.m* käyttäen OCTAVE:n editoria:

```
xd=(1:6)';  % xdata pystyvektorina
yd=[16;18;21;17;15;12];
              % ydata pystyvektorina
N=length(xd); % datapisteiden lukumäärä
ykkoset=ones(N,1); % Sama kuin [1 1 1 1 1 1]'
V=[ykkoset xd xd.^2 xd.^3 xd.^4 xd.^5];
a=V\yd; % Yhtälöryhmän ratkaisu
```

Muista: `xd.^k` tarkoittaa vektorin `xd` jokaisen komponentin korottamista potenssiin k, eli "pisteittäistä" potenssia.

Suositeltavaa on sijoittaa komennot tiedostoon `vanderinterp.m` ja ajaa `run`-painikkeella tai kirjoittamalla komentoikkunaan `vanderinterp` . Tällöin voit esim. editoida rivien loppujen puolipisteitä pois, jolloin näet välituloksia, joita emme näytä.

```
>> vanderinterp
a =
    69.00000
  -115.80000
    87.66667
   -28.95833
     4.33333
    -0.24167
```

Saadaan samat kuin edellä, mutta käännetyssä järjestyksessä, koska käsiteltiin polynomia kasvavien potenssien järjestyksessä. Kun haluamme laskea polynomin arvoja, käännetään se komennolla `flipud` (ud viittaa suuntaan "updown", vaakavektorin tapauksessa `fliplr`, "leftright"). Nyt voidaan laskea arvoja vaikkapa edellisen esimerkin vektorilla ja piirtää näillä komennoilla:

```
>> x=0.75:.05:6.25;
>> c=flipud(a);
>> p=polyval(c,x);plot(x,p);grid on
```

Huomaa, että monet komennot, kuten tässä `polyval, plot` toimivat yhtä hyvin vaaka- kuin pystyvektoreilla. Matriisialgebrassa näin ei yleensä ole.

Kysymys: Onko yhtälöryhmällä aina yksikäsitteinen ratkaisu? Jotkut lukijat saattavat tietää, että on, jos matriisin V determinantti $\neq 0$. Voidaan osoittaa, että Vandermonden matriisin determinantti on xdata-vektorin erotusten tulo, ja siis $\neq 0$, kun xdatapisteet ovat erilliset. Mutta verrattomasti helpommalla päästään puhtaasti polynomien perusominaisuuden avulla:

Jos kaksi korkeintaan astetta n olevaa polynomia yhtyy (n+1):ssä pisteessä, ne yhtyvät kaikkialla, eli ovat identtiset.

Tämä perustuu siihen yksinkertaiseen havaintoon, että jos polynomilla p on nollakohta x_0, niin $p(x)$ on jaollinen $(x - x_0)$:lla. Kts. [s04] Lause 1 ja Seuraus 1.

Pyydetään avuksi herroja *Lagrange* ja *Newton*

Edellä esitetty Vandermonden matriisiin perustuva menetelmä on samalla hyvä lähtökohta pienimmän neliösumman approksimaatiolle. Huonoa siinä on etenkin interpolaation kannalta matriisin "häiriöalttius", kun sen koko kasvaa. Numeerisen lineaarialgebran kannalta determinantin merkitys on vähäinen, tärkeää on "melkein singulaarisuus", josta determinantin "melkein nolla" ei kerro. Häiriöalttiutta arvioidaan erilaisella keinolla, jota ei tässä voida pitemmälle kehitellä. MATLAB/OCTAVE:ssa on funktiot `cond` ja `rcond` tähän tarkoitukseen.

Molempien menetelmien ydin on polynomien nokkelassa esitysmuodossa. Samalla, kun herrat konstruoivat ratkaisun, he saavat ratkaisun yleisen olemassaolotodistuksen tarvitsematta mitään muuta kuin edellä mainittua polynomien perusominaisuutta. Siis olemassaolotodistus ja ratkaisun laskennan kannalta tehokas ja vähemmän virhealtis konstruktio samassa paketissa.

1. Lagrangen menetelmä

Kirjoitin tämän auki jutussa [s04] hyvin konkreettisesti ja seikkaperäisesti. Mennään nyt suoraan yleiseen tapaukseen korostamalla menetelmän keksimisen ideaa.

Lähdetään hakemaan eräässä mielessä ortogonaalista polynomijoukkoa ("kantaa") $L_0(x), L_1(x), \ldots, L_n(x)$. Asetetaan kaksi vaatimusta:

1. Polynomit $L_k(x)$ ovat astetta n.
2. Ne toteuttavat "ortogonaalisuusehdon": $L_i(x_j) = \delta_{i,j} = 1$, kun $i = j$, ja $= 0$, kun $i \neq j$.

Jos tällaiset polynomit löydetään, voidaan ratkaisupolynomi p kirjoittaa suoraan:

$$p(x) = y_0 L_0(x) + y_1 L_1(x) + \ldots + y_n L_n(x),$$

kuten kohta osoitetaan.

Miten nämä kantapolynomit löydetään? Tarvitaan astetta n oleva polynomi $L_k(x)$, joka saa arvon 0 kaikissa pisteissä $x_j, j \neq k$. No mutta eihän ole muuta mahdollisuutta kuin valita:

$$L_k(x) = c(x-x_0)\ldots(x-x_{k-1})\ldots(x-x_{k+1})\ldots(x-x_n).$$

Tässä on yksi vapaasti valittava kerroin c, joka määrätään normeerausehdosta: $L_k(x_k) = 1$. Siis, jos merkitään

$$l_k(x) = (x-x_0)\ldots(x-x_{k-1})\ldots(x-x_{k+1})\ldots(x-x_n),$$

niin

$$c = \frac{1}{l_k(x_k)} \quad \text{ja siis:} \quad L_k(x) = \frac{l_k(x)}{l_k(x_k)} = \prod_{j\neq k} \frac{x - x_j}{x_k - x_j}.$$

Sanallisesti: Osoittajassa on termit $(x - x_j), j \neq k$, ja nimittäjässä $(x_k - x_j), j \neq k$. (Nimittäjästä puuttuu termi, joka tekisi sen nollaksi.)

Interpolaatioehto toteutuu:

$$p(x_k) = \underbrace{y_0 L_0(x_k)}_{=0} + \ldots + \underbrace{y_k L_k(x_k)}_{=y_k} + \\ + \underbrace{y_{k+1} L_{k+1}(x_k)}_{=0} + \ldots + \underbrace{y_n L_n(x_k)}_{=0} = y_k.$$

(Jos $k = 0$, saadaan $y_0 + 0 + \ldots + 0$.)

Huom: Lagrangen polynomit riippuvat pelkästään x-datasta.

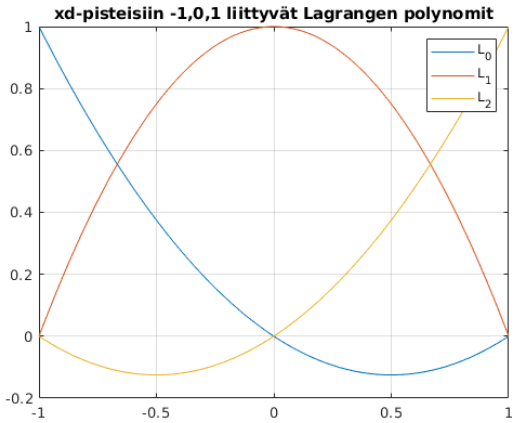
Kirjoitetaan Matlab-koodiksi (Octave:lla)

Ideoidaan ensin pienellä ”manuaalisella” esimerkillä:

```
% Tiedosto Lag2.m
%% Lagrangen 2. asteen polynomit
% Muodostetaan ''manuaalisesti'' xd-dataan
% liittyvät Lagrangen (2. asteen) polynomit
% Huom! Vektorin indeksit alkavat 1:stä..
xd=[-1 0 1];
x=linspace(-1,1,100);% Laskentapisteet 100 kpl
L0=(x-xd(2)).*(x-xd(3))./((xd(1)-xd(2))*...
 (xd(1)-xd(3)));
L1=(x-xd(1)).*(x-xd(3))./((xd(2)-xd(1))*...
(xd(2)-xd(3)));
L2=(x-xd(1)).*(x-xd(2))./((xd(3)-xd(1))*...
(xd(3)-xd(2)));
plot(x,L0,x,L1,x,L2); grid on
legend('L_0','L_1','L_2')
title('xd-pisteisiin liittyvät L-polynomit')
```

Huomaa, että `(x-xd(2)).*(x-xd(3))` on 100:n pituisten vektorien vastinalkioittainen tulo, samoin muut vastaavat. Nimittäjässä kerrotaan skalaareja, joten piste (.) ei ole tarpeen, mutta ei haittaisikaan.

Editoi komennot vaikka tiedostoksi `Lag2.m` ja kirjoita komentoikkunaan `Lag2` saadaksesi alla olevan, Lagrangen polynomien ominaisuuksia havainnollistavan kuvan.



Tältä pohjalta on helppo rakentaa funktio, joka suorittaa Lagrangen interpolaation:

```
function y=LagrangeInterp(xd,yd,x)
N=length(xd);
y=zeros(size(x));% Summavektorin alustus: 0
for i=1:N
  y=y+yd(i)*L(i,xd,x); % skal. kertaa vektori
end
% L_k-funktio alifunktiona:
function  y  = L(k,xdata,x)
% xdata-vektoriin liittyvä Lagrangen
% kantafunktio L_k laskettuna vektorilla x.
n=length(xdata);
y=ones(size(x));% Tulovektorin alustus: 1
for j=[1:k-1 k+1:n]
    y=y.*((x-xdata(j))./(xdata(k)-xdata(j)));
end
```

Huom1: Kuten MATLAB-ohjelmoinnissa hyviin tapoihin kuuluu, laskentapisteitä ”pitää sallia” ainakin kokonaisen vektorin verran. Tämä hoidetaan ”pisteittäisillä” laskutoimituksilla `(.*)` ja (./).

Huom2: Kutsu `L(1,...)` johtaa indeksivektoriin: `for j=[1:0 2:n]`, joka on $[2, \ldots, n]$, kuten pitää, koska `1:0` on tyhjä vektori, katsotaan:

```
>> 1:0
ans = 1×0 empty double row vector
```

Leikitäänpä vähän LagrangeInterp-työkalulla

Jotta voisit käyttää kirjoittamaamme funktiota, täytyy koodi kirjoittaa tekstitiedostoon `LagrangeInterp.m`, joka sijaitsee työhakemistossa tai polun (`path`) varrella. Funktiota kutsutaan aivan samoin kuin mitä tahansa OCTAVE:n sisäänrakennettua funktiota, siis komentoikkunassa kirjoitetaan esim. `>> y=LagrangeInterp(xd,yd,x)`, kun muuttujille `xd`, `yd`, `x` on annettu halutut arvot.

Kun halutaan muutella parametrejä ja tehdä erilaisia testiajoja, on suositeltavaa kirjoittaa tarvittavat parametrien muodostamiskomennot ja funktiokutsut skriptitiedostoon, vaikka `ajo.m` ja komentaa: `>> ajo` . Näillä komennoilla suoritettaisiin nyt edellä `vanderinterp`-tyylillä käsitelty esimerkki.

```
xd=1:6
yd=[16 18 21 17 15 12]
x=.75:.05:6.25;
p=LagrangeInterp(xd,yd,x);
plot(xd,yd,'o',x,p,'MarkerSize',10)
grid on
```

Huomattavaa on, että `LagrangeInterp`-funktiomme toimii myös symbolisella argumentilla x, jos käytössä on MATLAB, jossa on ”symbolic toolbox”. Palataan tähän loppupuolella.

2. Newtonin menetelmä

Ajatellaanpa, että olemme onnistuneet tavalla tai toisella muodostamaan interpolaatiopolynomin sanokaamme datalle

x_0	x_1	x_2
y_0	y_1	y_2

Halutkaamme lisätä datapiste (x_3, y_3) ja laajentaa polynomimme tälle uudelle datapisteistölle. Edellä esitellyillä menetelmillä ei auttaisi muu kuin laskea koko homma alusta uudelleen. Olkoon tuo edellä saatu polynomi $p_2(x)$. (Ankaraa mietintää...)

HEUREKA! Lisätään polynomiimme sellainen polynomi $q(x)$, joka saa arvon 0 kaikissa vanhoissa pisteissä x_0, x_1, x_2 ja on asteeltaan yhtä korkeampi. No mutta (ainoa) sellainenhan on:

$$q(x) = c(x - x_0)(x - x_1)(x - x_2).$$

Tämän polynomin lisääminen ei tuhoa interpolaatioehtoja vanhoissa pisteissä, ja siinä on yksi määrättävä vakio c, jonka avulla säädetään uusi ehto voimaan. Siispä $y_3 = p_3(x_3) = p_2(x_3) + q(x_3)$, joten

$$c = \frac{y_3 - p_2(x_3)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)}.$$

Jälleen päivänselvä oivallus, kunhan sen vain keksii! (Joillekin mm. Ar-, Ga-, La-, Ne- alkuisille näitä sattuu useammin kuin toisille!).

Newtonin muoto interpolaatiopolynomille voidaan kehittää aloittamalla astetta 0 olevasta polynomista $p_0 \equiv y_0$, sitten $p_1(x) = y_0 + c_1(x - x_0)$, $p_2(x) = p_1(x) + c_2(x - x_0)(x - x_1)$ jne. Vakioiden c_k määräämiseksi saadaan yhtälöt:

$$\begin{cases} y_0 = c_0 \\ y_1 = c_0 + c_1(x_1 - x_0) \\ y_2 = c_0 + c_1(x_2 - x_0) + c_2(x_2 - x_0)(x_2 - x_1) \\ \vdots \end{cases}$$

Taaskaan ei tarvita yhtälöryhmän yleistä ratkaisua, vaan voidaan edetä ylhäältä alas ratkaisemalla kullakin rivillä yksi ensimmäisen asteen yhtälö.

Näin päädytään yleiseen muotoon (NP):

$$p_n(x) = c_0 + c_1(x - x_0) + c_2(x - x_0)(x - x_1) + \ldots + c_n(x - x_0)(x - x_1)\cdots(x - x_n).$$

Tämä on **Newtonin interpolaatiopolynomi** ja sen konstruktiotapa on samalla aiemmista riippumaton todistus tehtävän yksikäsitteiselle ratkaisulle. Kaikki ratkaisutavat tuottavat siis saman polynomifunktion, mutta erilaisen, toisiinsa sievennettävissä olevan esitysmuodon.

Newtonin polynomi (NP) voidaan kirjoittaa lyhyesti tulo- ja summasymbolien avulla:

$$p_n(x) = \sum_{i=0}^{n} c_i \prod_{j=0}^{i-1}(x - x_j).$$

Pikkutehtävä: Olkoot pisteet $x_0 = 1, x_1 = 2, x_2 = 5$, $y_0 = 6$, $y_1 = 4$, $y_2 = 7$. Yhtälöryhmästä peräkkäisillä sijoituksilla ratkaisemalla saadaan: $c_0 = 6$, $c_1 = -2$, $c_2 = \frac{3}{4}$, joten

$$p(x) = 6 - 2(x - 1) + \frac{3}{4}(x - 1)(x - 2).$$

Vähän kookkaamman polynomin auki kertominen on viheliäistä puuhaa. Kannattaa käyttää ympärillä tarjoutuvia symbolilaskentamahdollisuuksia, vaikkapa *Maxima*-ohjelman `expand`-komentoa. MATLAB:n symbolic toolbox tarjoaa suoraan LaTeX-kaavan:

```
>> latex(expand(p))
\frac{3\,x^2}{4}-\frac{17\,x}{4}+\frac{19}{2}
```

Eli LaTeX-kääntäjän jäljiltä: $\frac{3\,x^2}{4} - \frac{17\,x}{4} + \frac{19}{2}$.

Kertoimet voidaan siis esimerkin tavoin laskea tuosta ”alakolmioyhtälöryhmästä”. Niille voidaan johtaa rekursiokaava ns. ”jaettujen erotusten” avulla, jolloin saadaan helposti muistettava menetelmä käsinlaskuun ja suorastaan riemastuttavan helposti kirjoitettava MATLAB-ohjelma. Lisäkehittelyn jälkeen saadaan ”tuotantokäyttöön” tehokas *Neville'n* algoritmi, mutta siihen ei nyt mennä.

Jaetut erotukset

Merkitään $y_k = f(x_k), k = 1, \ldots, n$, missä f tarkoittaa (x_k, y_k)-datapisteiden määräämää diskreettiä funktiota. Merkitään $f[x_k] = f(x_k)$, ja tarkoittakoon yleisesti $f[x_0, x_1, \ldots, x_k]$ dataan $((x_0, f(x_0)), \ldots, (x_k, f(x_k)))$ liittyvän interpolaatiopolynomin korkeimman asteen termin kerrointa, toisin sanoen edellä kirjoitetun Newtonin esitysmuodon kerrointa c_k. Näitä kutsutaan nimellä *jaetut erotukset* seuraavan lauseen perusteella.

Lause. Jaetut erotukset toteuttavat rekursiokaavan:

$$f[x_0, \ldots, x_k] = \frac{f[x_1, x_2, \ldots, x_k] - f[x_0, x_1, \ldots, x_{k-1}]}{x_k - x_0}.$$

Todistus. Jätän todistuksen viitteen [GC] s.188 *Theorem 8.3.1* varaan sekä ajan että tilan puutteen vuoksi. Todistus on yllättävän helppo ja elegantti yllä olevan määritelmän ansiosta ja Newtonin aste kerrallaan nousevasta esityksestä johtuen. □

Neljän datapisteen jaettujen erotusten taulukko:

$$
\begin{array}{llll}
f[x_0] & & & \\
f[x_1] & f[x_0,x_1] & & \\
f[x_2] & f[x_1,x_2] & f[x_0,x_1,x_2] & \\
f[x_3] & f[x_2,x_3] & f[x_1,x_2,x_3] & f[x_0,x_1,x_2,x_3]
\end{array}
$$

Taulukon lävistäjä

$$(f[x_0], f[x_0,x_1], f[x_0,x_1,x_2], f[x_0,x_1,x_2,x_3])$$

antaa siis Newtonin esitysmuodon kertoimet.

Pikkutehtävä uudestaan:

x_j	y_j		
1	6		
2	4	$\frac{4-6}{2-1}=-2$	
5	7	$\frac{7-4}{5-2}=1$	$\frac{1+2}{5-1}=\frac{3}{4}$

Saatiin siis samat kertoimet: $6, -2, \frac{3}{4}$. Matematiikassa on magiaa!

Ja nyt kaikki osaavat rakentaa Newtonin polynomin vaikka unissaan! Johan sitä alussakin harjoiteltiin.

Palataan shakkilautatehtävään

Koska x-data on tasavälinen, jopa niin, että askel = 1, ovat jakajana olevat erotukset $= 1, 2, 3, \ldots$, siis alussa esitettyyn erotuskaavioon tarvitsee vain lisätä koodissa näkyvät kertoimet $1/2, 1/3, 1/4, 1/5$.

```
xdata=0:7; % Tasavälinen askel=1
ydata=[0, 6, 28, 96, 252 550 1056 1848]';
ydiff1=[NaN;diff(ydata)];
ydiff2=[NaN;1/2*diff(ydiff1)];
ydiff3=[NaN;1/3*diff(ydiff2)];
ydiff4=[NaN;1/4*diff(ydiff3)];
ydiff5=[NaN;1/5*diff(ydiff4)];
erotustaulukko=[ydata ydiff1 ydiff2...
  ydiff3 ydiff4 ydiff5]
```

Tämä skripti tuottaa tulostuksen:

```
erotustaulukko =
 1.0e+03 *
     0      NaN      NaN      NaN      NaN
0.0060   0.0060      NaN      NaN      NaN
0.0280   0.0220   0.0080      NaN      NaN
0.0960   0.0680   0.0230   0.0050      NaN
0.2520   0.1560   0.0440   0.0070   0.0005
0.5500   0.2980   0.0710   0.0090   0.0005
1.0560   0.5060   0.1040   0.0110   0.0005
1.8480   0.7920   0.1430   0.0130   0.0005
```

Jätetään tilan puutteen takia viimeinen sarake näyttämättä, vakiosarakkeesta seuraava koostuu numeerisilta osiltaan tietysti kolmesta 0:sta.

Kuten *Newtonin* ja *Lagrangen* muodoista näkyy, ja *Vandermonde*-ratkaisustakin (Gaussin eliminaatio) selviää, kokonaislukudata johtaa aina rationaalikertoimiin. Komento `diag` poimii matriisin päälävistäjän.

```
format rat   % Rationaaliaritmetiikka
c=diag(erotustaulukko);
c=c(1:end-1)' % Jätetään viimeinen 0 pois
syms x     % Symbolinen x, vain Matlab:ssa
p=c(1)+c(2)*x+c(3)*x*(x-1)+...
 c(4)*x*(x-1)*(x-2)+c(5)*x*(x-1)*(x-2)*(x-3)
p=expand(p)
a=sym2poly(pe)  % Symbolinen muoto ...
                % kerroinvektoriksi
arvotxdatalla=polyval(a,xdata)% Tarkistus:
         % Lasketaan polynomi datapisteissä.
```

Viemällä nämä suoraan komentoikkunaan tai editoriin, josta skripti ajetaan, saadaan:

```
c =
    0     6     8     5     1/2
p =
    x^4/2 + 2*x^3 - (3*x^2)/2 + 5*x
a =
    1/2     2     -3/2     5     0
     arvotxdatalla =
      0     6     28     96     252 ...
       550          1056          1848
```

Kappas vaan, oikein meni!

Jos/kun operoit OCTAVE:lla, voit suoraan "leikkausliimata" OCTAVE:sta MAXIMA:an (sijoitusmerkki (:), pakollinen loppumerkki (;)):

```
(%i1) p:6*x+ 8*x*(x-1)+ 5*x*(x-1)*(x-2)...
 + (x*(x - 1)*(x - 2)*(x - 3))/2;
(%i2) expand(p);
                   4             2
                  x       3   3 x
(%o2)             -- + 2 x  - ---- + 5 x
                  2            2
```

Toki voit sijoittaa lausekkeen myös Wofram Alphaan, joka tekee sievennyksiä, grafiikkaa ym. ihan pyytämättäkin.

Miten jono jatkuu?

Saatiin siis polynomimalli $p(x) = \frac{x^4}{2} + 2x^3 - \frac{3x^2}{2} + 5x$, joka selittää 3 ylimääräistä data-arvoa: `[550 1056 1848]` x-arvoilla: `[5 6 7]`. Lasketaan lisäarvoja:

```
>> lisapisteita=8:12;
>> lisarvoja=polyval(a,lisapisteita);
>> [lisapisteita;lisarvoja]
ans =
   8     9     10     11     12
 3016   4662   6900    9856  13668
```

Pitääkö tämä yksinkertainen laskukaava yleisesti paikkansa? Jään odottamaan vastausta syvemmin kombinatoriikan menetelmiin perehtyneeltä matemaatikolta. Erityisen kiusallista on, että annettu data loppuu juuri oikean shakkilaudan kynnyksellä, joten jäämme tälle kynnykselle toivorikkaina luku 3016 taskussamme.

Symbolista summausta

Kaikkihan muistamme tarinan koulupojasta nimeltään *Gauss* ja laskutehtävästä: $s_n = \sum_{k=1}^{n} k$, $(n = 100)$. Gaussin oivallus oli vektoriyhteenlasku, joka voitaisiin ilmaista OCTAVE:lla: `>> (1:10)+(10:-1:1)`, jos otetaan $n = 10$. Idea johtaa heti kaavaan: $s_n = \frac{n(n+1)}{2}$.

Myös geometrisen jonon summakaava saadaan kätevästi ilman taulukkokirjaa tai symboliohjelmaa:

$$\begin{cases} s_n = 1 + q + q^2 + \ldots + q^n \\ qs_n = q + q^2 + \ldots + q^{n+1} \end{cases}$$

Vähentämällä ja ratkaisemalla saadaan: $s_n = \frac{1-q^{n+1}}{1-q}$.

Harvoille muille jonoille on näin yksinkertaisia summakaavoja näin helposti johdettavissa, jos ollenkaan. Kyse on siis lausekkeesta summalle

$$s_n = a_1 + a_2 + \ldots + a_n,$$

kun kerroinjono (a_k) on annettu. Tämä voidaan mieltää diskreettinä versiona annetun funktion integraalifunktion määräämistehtävästä. Symbolilaskentaohjelmat ovat varsin kehittyneitä tällä alueella. Toki yleisesti toimivaa algoritmia ei ole, kun ei "suljetun muodon"[2] ratkaisuakaan aina ole.

Edellisten lausekkeiden tapaan voisimme yrittää löytää polynomi- tai rationaalifunktiokaavan. Strategia voisi olla sellainen, että ensin kokeillaan, löytyisikö suhteellisen matalaa astetta oleva interpolaatiopolynomi, jonka asteluku ei kasva, kun jonon lukuja (dataa) lisätään. Näin saadaan hypoteesi, joka todistetaan induktiolla oikeaksi, jos pystytään. Jos polynomin asteluku on suurempi kuin esim. 3, joudutaan induktioaskeleessa varsin pitkiin sievennyksiin. Tässä kohden parhaat symboliohjelmat ovat luotettavampia kuin ihminen, ja tällaisella tietokoneavustuksella laadittu todistus on ilman muuta hyväksyttävä (eikö vain). Siis ihminen tietää, mihin sievennyksellä pyritään, ohjelma tekee, mitä käsketään nopeasti ja luotettavasti verrattuna hitaaseen ja epäluotettavaan ihmiseen.

Esimerkki: $s_n = \sum_{k=1}^{n} k^p$, missä p on kokonaisluku.

Tarkastellaan tapausta $p = 5$.

```
ind=(1:10)'; % Pystyvektori transponoimalla(')
jono=(ind.^5);
S=cumsum(jono); % Kumulatiiviset summat
indeksit_jono_ja_osasummat=[ind jono S]

indeksit_jono_ja_osasummat =
       1             1             1
       2            32            33
       3           243           276
       4          1024          1300
       5          3125          4425
       6          7776         12201
       7         16807         29008
       8         32768         61776
       9         59049        120825
      10        100000        220825
```

Osasummien jaetut erotukset:

```
format rat  % Pelkkiä rationaalisia laskuja
ydiff1=[NaN;diff(S)]; % Sama kuin alkup. jono
ydiff2=[NaN;1/2*diff(ydiff1)];
ydiff3=[NaN;1/3*diff(ydiff2)];
ydiff4=[NaN;1/4*diff(ydiff3)];
ydiff5=[NaN;1/5*diff(ydiff4)];
ydiff6=[NaN;1/6*diff(ydiff5)];
erotustaulukko=[S ydiff1 ydiff2 ydiff3 ydiff4...
 ydiff5 ydiff6];
c=(diag(erotustaulukko))'%  Newtonin polynomin
                         %  kertoimet
```

Tulostus:

```
c =
     1     32      211/2      95
    125/4            4             1/6
```

Siirrytään symbolinkäsittelyyn. Teen sen nyt MATLAB:n symbolic toolboxilla, mutta tein vastaavat operaatiot myös MAXIMA:lla ja MAPLE:llä ihan leikkausliimauksella ko. ohjelmiin.

```
>> syms n  % Julistetaan n symboliseksi.
>> p=c(1)+c(2)*(n-1)+c(3)*(n-1)*(n-2)+...+
        ...                % Piilotetaan osa
    c(7)*(n-1)*(n-2)*(n-3)*(n-4)*(n-5)*(n-6);
>> p_n=simplify(p)
  p_n =
   (n^2*(n + 1)^2*(2*n^2 + 2*n - 1))/12
>> p_nplus1=subs(p,n,n+1)
% Sijoitetaan p_n:n lausekkeessa
% n:n paikalle n+1
p_nplus1 =
((n+1)^2*(n+2)^2*(2*n + 2*(n + 1)^2 + 1))/12
% Induktioaskel:
>> erotus=p_nplus1-p_n
erotus =
((n+1)^2*(n+2)^2*(2*n + 2*(n+1)^2 + 1))/12-..
  (n^2*(n + 1)^2*(2*n^2 + 2*n - 1))/12
```

[2] Alkeisfunktioiden avulla n:n lausekkeena lausuttavissa oleva muoto.

```
>> simplify(erotus)
ans =
(n + 1)^5  % Mutta niinhän sen piti olla!
>> display('MOT')
MOT
```

Kaava on todettu jo useallakin n:n arvolla, ja nyt saatiin osoitetuksi yleinen askel, joten lauseke

$$s_n = p_n = (n^2(n+1)^2(2n^2+2n-1))/12$$

on voimassa kaikilla n.

Huomautuksia

1. Vähemmällä kaavan kirjoittamisella oltaisiin päästy valmiiksi ohjelmoimallamme `LagrangeInterp`-funktiolla. Tämä sisältyy esimerkkiskriptiin `SymsumEsim.m`, mutta soveltuu myös harjoitustehtäväksi, mikäli MATLAB ja symbolic toolbox ovat käytössä.

2. Kaikkein helpointa luulisi olevan MATLAB-funktion `polyfit` käyttö interpolointiin. Mutta se laskee liukuluvuilla, ja viimeisissä desimaaleissa esiintyvät pyöristysvirheet estävät muuntamisen oikeiksi rationaaliluvuiksi. Oma `LagrangeInterp`-funktio toimii myös symboliselle argumentille, samoin Newtonskriptit, joiden pohjalta annoin tehtävän kehittää vastaavasti `NewtonInterp`-funktion. Molemmathan suorittavat pelkkiä rationaalilaskuja. Symboliohjelmat, kuten MAPLE (ja varmaankin myös MAXIMA) sisältävät valmiit funktiot, jotka oletusarvoisesti laskevat rationaaleilla.

3. Symboliohjelmissa on hyvät välineet symbolisten summien laskentaan, niinpä tässäkin saadaan MATLAB-toolboxilla:

```
>> syms k n;symsum(k^5,[1,n])
ans =
(n^2*(n + 1)^2*(2*n^2 + 2*n - 1))/12
```

Tehtäviä

1. Olkoot `xd=[0:20:100]` ja
`yd=[76.0 105.7 131.7 179.3 226.5 281.4]`
`xd`-arvot edustavat vuosilukuja vuoden 1900 jälkeen ja `yd`:t USA:n väestölaskentadataa miljoonan asukkaan yksiköissä.
(a) Muodosta (käsin) *Lagrangen* menetelmän mukainen muoto 2. asteen polynomille, joka interpoloi vuosilukuihin 1900, 1920, 1940 liittyvää väestödataa.
(b) Muodosta *Newtonin menetelmällä* asteita 0, 1 ja 2 olevat interpolaatiopolynomit samoille 3:lle ensimmäiselle datapisteelle, ja näytä, että saat saman 2. asteen polynomin kuin (a)-kohdassa.
(c) Piirrä datapisteet. Sovita dataan interpolaatiopolynomi vaikkapa `LagrangeInterp`-funktion avulla. Piirrä polynomi samaan kuvaan. Mikä on "ennuste" vuosille 2020, 2030? Sovita myös `polyfit`:llä alemman asteisia PNS-polynomeja realistisemman kasvuennusteen saamiseksi.

2. Kirjoita funktio *NewtonInterp*, jonka kutsu on sama kuin funkiolla *LagrangeInterp*. Kehitä funktiokoodi shakkilautatehtävän alussa esiintyvän skriptin pohjalta.

3. Määritä tapauksen $p = 5$ käsittelyn tyylillä lauseke summalle $s_n = \sum_{k=1}^{n} k^p$ joillakin muilla p:n arvoilla. Kokeile ehkä ensin jotain arvoa $p < 5$, ja jos haluat lisää haastetta, niin kasvata p:tä. Voit tehdä interpolaatio-osan joko `LagrangeInterp`- tai edellisen tehtävän `NewtonInterp`- funktiolla. Induktiotodistus esim. Maximalla, tai mihin nyt pääset käsiksi.

4. Vastaava tehtävä jonolle:

$$a_k = (k-1)k(k+1), k = 1, 2, 3, \ldots$$

Vastaus:

```
>>S=symsum((k-1)*k*(k+1),k,[1 n])
>>expand(S)
n^4/4 + n^3/2 - n^2/4 - n/2
```

Viitteet

[AL] Antti Laaksonen: Kuinka löytää kaava lukujonolle, Solmu 1/2019.

[CM] https://se.mathworks.com/moler.html, MATLAB-ohjelman kehittäjän, *Cleve Molerin* kaksi erinomaista, vapaasti luettavaa verkkokirjaa: "Experiments" ja "Numerical computing" with Matlab. Erityisesti "Experiments" sisältää paljon koulumatematiikkataustaan sopivaa materiaalia.

[GC] *Anne Greenbaum, Timothy P. Chartier*: Numerical Methods. Design, Analysis and Computer Implementation of Algorithms, Princeton U.P. 2012.

[Samu] *Samuli Siltanen*: Astu matematiikan maailmaan, Otava 2019.
Sisältää runsaasti matemaattiseen mallintamiseen liittyvää ajattelua ja esimerkkejä lukijaystävällisessä muodossa.

[s04] https://matematiikkalehtisolmu.fi/2004/3/apiola.pdf

[skript] https://matematiikkalehtisolmu.fi/2019/2/skriptit/
Tähän kirjoitukseen liittyvät m-tiedostot, myöhemmin myös tehtävien ratkaisut.