76315A ATKII - NUMEERINEN MALLINTAMINEN

Sisällysluettelo

Mathematica ja vinkkejä sen kanssa elämiseen

Lausekkeet, symbolit ja komennot

Unix-systeemissä: *Mathematica* käynnistyy komennolla mathematica, jolloin avautuu erillinen X-ikkuna, graaffinen käyttäjäliittymä (notebook), *Mathematica*' n komentoihin. Istunto lopetetaan valitsemalla FILE valikosta **Quit**. *Mathematica* on myös mahdollista käynnistää komennolla math, jolloin erillistä ikkunaa ei avaudu.

*Mathematica*n komennot kirjoitetaan ikkunaan ja ne suoritetaan painamalla SHIFT ja ENTER nappuloita yhtä aikaa. *Mathematica* numeroi suorittamansa käskyt **In[n]** ja niihin annetut vastaukset **Out[n]** juoksevin numeroin siten, että käskyllä ja vastauksella on sama numero.

```
In[1]:= 1 + 1
```

```
Out[1]= 2
```

Aiempiin tuloksiin voi viitata **%n**-merkinnällä, missä **n** on tulosterivin numero. Esimerkiksi

```
In[2]:= %1 + 4
```

```
Out[2]= 6
```

Toinen mahdollisuus on käyttää **%**-merkkiä viittaamaan viimeisimpään tulokseen, **%%**-merkkejä toiseksi viimeiseen tulokseen ja niin edelleen.

■ Lausekkeet ja symbolit

*Mathematica*ssa käytetään kirjaimia ja numeroita symbolien nimien esittämiseen, kuten matematiikassakin. Symbolien nimissä isot ja pienet kirjaimet tarkoittavat eri asiaa ja vain englannin kielen kirjaimet kelpaavat. Nimellä ei ole pituusrajoitusta, mutta se ei saa alkaa numerolla. Siten **x2** on hyväksyttävä symboli, mutta **2x** ei ole, vaan se tulkitaan kertolaskuksi $2 \cdot x$.

Yksinkertainen tapa antaa symbolille arvo on *sijoituslause*.

Sijoituslauseita on kahta tyyppiä

`symboli = lauseke` (*suora asetus*)

`symboli := lauseke` (*viivästetty asetus*)

Operaattori **=** laskee lausekkeen niin pitkälle kuin mahdollista ja määrittelee saadun tuloksen symboliksi. Tämä on ns. suora asetus.

Operaattori **:=** määrittelee lausekkeen sellaisenaan symboliksi, ja sen arvo lasketaan aina uudelleen, kun symbolia käytetään. Tätä kutsutaan viivästetyksi asetukseksi.

Esimerkiksi

```
In[3]:= a = 5
Out[3]= 5

In[4]:= b := a + 1

In[5]:= b
Out[5]= 6

In[6]:= a = 6
Out[6]= 6

In[7]:= b
Out[7]= 7

In[8]:= d = 3
Out[8]= 3

In[9]:= c = a + b + d
Out[9]= 16
```

```
In[10]:= c
Out[10]= 16

In[11]:= d = 1
Out[11]= 1

In[12]:= c
Out[12]= 16
```

Ensimmäisessä osassa **a**:n arvon muuttaminen muuttaa **b**:n arvon. Jälkimmäisessä tapauksessa näin ei tapahdu. Symbolin arvon voi tulostaa aina kirjoittamalla sen nimi tai käyttämällä **Print**-komentoa.

Esimerkiksi

```
In[13]:= Print[a]
6
```

Symbolin arvon voi hävittää merkinnällä

```
x=.
```

tai

```
Clear[x]
```

Sen jälkeen symbolia **x** voi käyttää esimerkiksi muuttujana.

Mathematicassa käytetään seuraavia merkintöjä peruslaskutoimituksille.

Kertolasku	**a*b**, **a b** (*Huomaa välilyönti!*)
Jakolasku	**a/b**
Yhteen- ja vähennyslasku	**a+b**, **a-b**
Potenssi	**a^b**

Kertolasku voidaan merkitä siis kahdella tavalla: **a*b** tai **a b**. Miksi tarvitaan välilyönti? Yksinkertaisesti siksi, että muuten *Mathematica* ei tiedä, tarkoitetaanko oliota nimeltä **ab** vai **a**:n ja **b**:n tuloa.

■ Komennot

Komennot *Mathematica*ssa ovat aina muotoa

`Komento[syötteet]`

Mathematica on symbolisen laskennan ohjelmisto. Symbolinen laskenta tarkoittaa sitä, että yhtälöitä voi esimerkiksi sieventää tai niistä voi yrittää eliminoida muuttujia. Symboliset ratkaisut ovat aina tarkkoja. Tutki seuraavaa esimerkkiä, jossa ratkaistaan toisen asteen yhtälö $2x^2 + 3x + 3 = 4$:

```
In[14]:=
Clear[x]
Solve[2 x^2 + 3 x + 3 == 4, x]
```

```
Out[15]= {{x → 1/4 (-3 - √17)}, {x → 1/4 (-3 + √17)}}
```

Yhtälöitä kirjoitettaessa käytetään **==** merkkiä yhtäsuuruuden ilmoittamiseen. Yhtälön ratkaisut esitetään sääntöinä, jotka ovat sijoitettu ryhmäsulkujen (**{}**) sisään. Ryhmäsulkujen avulla muodostettuja olioita kutsutaan listoiksi.

■ Listat

Lista on joukko alkioita, jotka ovat erotettu toisistaan pilkuilla ja kirjoitettu aaltosulkujen sisään.

Esimerkiksi

```
In[16]:=
lista1 = {9, 8, 7, 6, 5}
```

```
Out[16]= {9, 8, 7, 6, 5}
```

```
In[17]:=
matriisi = {{K, L, M}, {N, O, P}, {Q, R, S}}
```

```
Out[17]= {{K, L, M}, {N, O, P}, {Q, R, S}}
```

Symboli **`lista1`** on lista, jonka alkiot ovat lukuja. Matematiikassa sillä voidaan esittää pystyvektoria. Symboli **`matriisi`** viittaa listaan, jonka alkiot ovat listoja. Tässä tapauksessa kyseessä voisi olla 3×3 matriisi, jonka alkiot ovat lueteltu vaakariveittäin.

Listojen alkioihin viitataan kaksinkertaisilla hakasuluilla.

```
In[18]:=
lista1[[3]]
```

```
Out[18]= 7
```

```
In[19]:= matriisi[[2]]
Out[19]= {N, O, P}

In[20]:= matriisi[[1, 3]]
Out[20]= M
```

Merkintä **`matriisi[[1, 3]]`** viittaa siis ensimmäisen alkion kolmanteen jäseneen.

■ Säännöt

`Solve`-komennon tulos saatiin siis sääntöinä

```
In[21]:= Solve[2 x^2 + 3 x + 3 == 4, x]
Out[21]= {{x → 1/4 (-3 - √17)}, {x → 1/4 (-3 + √17)}}
```

Sääntö kertoo, että nuolen (→) vasemmalla puolella oleva olio **x** on korvattavissa nuolen oikealla puolella olevalla lausekkeella. Kun sääntöä sovellettaan, kohteena olevassa lausekkeessa korvataan **x**:n jokainen esiintymä säännön oikean puolen lausekkeella.

Esimerkkinä lasketaan lausekkeen $x^3 + 5$ arvo yllä saadun listan ensimmäistä sääntöä käyttäen:

```
In[22]:= x^3 + 5 /. %[[1]]
Out[22]= 5 + 1/64 (-3 - √17)^3
```

Lausekkeen ja säännön väliin kirjoitetaan sijoitusoperaattori

```
/.
```

Symboli **x** on lausekkeessa **x^3+5** muuttuja, jolle sijoitetaan arvo edellisen listan ensimmäistä sääntöä käyttäen.

Lausekkeet

```
In[23]:= x1 + x2 + x3 /. x1 -> 2
Out[23]= 2 + x2 + x3
```

```
In[24]:= A x1 + B x2 + C x3 /. {B → A, C → A}
Out[24]= A x1 + A x2 + A x3
```

ovat myös esimerkkejä lausekkeista, joissa esiintyy sääntö tai lista sääntöjä. Ensimmäisessä esimerkissä korvataan **x1** arvolla 2. Toisessa esimerkissä **{B->A,C->A}** on lista, jossa on kaksi sääntöä, ja tarkoittaa ``korvaa **B** **A**:lla, korvaa **C** **A**:lla".

■ Tehtäviä

1. Laske $3\sin(4.0)$. Lisää tulokseen 3. Jaa tulos, joka sinulla oli ennen 3:n summaamista, 2:lla.

2. Kokeile, mitä tekevät **a==a**, **2==1**, **a==2**.

3. Mieti, miten voit sijoittaa säännön **x→Sqrt[2]+5** mukaisen arvon muuttujalle **x** pysyväksi arvoksi.

4. Kokeile suoran ja epäsuoran sijoituksen eroja sijoittamalla **x=a**, **suora=2x**, **viive:=2x** ja **x=(k+1)**. Mitkä ovat nyt muuttujien **suora** ja **viive** saamat arvot?

5. Ratkaise yhtälö $y^3 + 4\,y - 9 = 0$.

Luonnonvakiot ja kompleksiluvut

Mathematica tuntee joukon luonnonvakioita ja erikoisempia matemaattisia olioita, kuten Neperin luvun e ja π:n. Ohessa on lista muutamista yleisimmistä vakioista:

```
In[25]:= E
Out[25]= ℯ

In[26]:= Pi
Out[26]= π

In[27]:= Infinity
Out[27]= ∞

In[28]:= I
Out[28]= ⅈ
```

Kompleksiluvut esitetään muodossa **a + b I**, missä **a** ja **b** ovat reaalilukuja.

Esimerkiksi lasketaan kahden kompleksiluvun tulo

```
In[29]:= a = 2 + 3 I
         b = 3 - I
         a b
```

```
Out[29]= 2 + 3 i
```

```
Out[30]= 3 - i
```

```
Out[31]= 9 + 7 i
```

Lausekkeen numeerinen arvo

Komennolla

N[lauseke]

tai

lauseke//N

lasketaan lausekkeen arvo desimaalilukuna, niiltä osin kuin se on mahdollista.

Esimerkiksi

```
In[32]:= Sqrt[2] + 5 / (Pi^2 + 10)
```

$$\text{Out[32]= } \sqrt{2} + \frac{5}{10 + \pi^2}$$

```
In[33]:= N[%]
```

```
Out[33]= 1.66585
```

```
In[34]:= E^3 / 5 // N
```

```
Out[34]= 4.01711
```

Lausekkeiden sieventäminen ja muokkaus

Lausekkeiden sieventämiseen ja muokkaukseen on runsaasti komentoja.

Seuraavassa on luettelo muutamista tärkeimmistä:

Expand	suorita kertolaskut ja potenssiin korottamiset
Factor	etsi yhteiset tekijät
Together	kirjoita lauseke yhteistä nimittäjää käyttäen
Apart	hajoita termit yksinkertaisia nimittäjiä käyttäen
Cancel	supista yhteiset tekijät osoittajasta ja nimittäjästä
Simplify	kokeile algebrallisia muunnoksia ja etsi niistä yksinkertaisin
Numerator	osoittaja
Denominator	nimittäjä

Lauseke, johon komento kohdistuu kirjoitetaan hakasulkeisiin tai käytetään merkintää **//**. Esimerkiksi

```
Expand[lauseke]
```

tai

```
lauseke//Expand
```

In[35]:=
```
e = (x - 1)^2 (2 + x) / ((1 + x) (x - 3)^2)
```

Out[35]=

$$\frac{(-1+x)^2\,(2+x)}{(-3+x)^2\,(1+x)}$$

In[36]:=
```
f = Expand[e]
```

Out[36]=

$$\frac{2}{(-3+x)^2\,(1+x)} - \frac{3x}{(-3+x)^2\,(1+x)} + \frac{x^3}{(-3+x)^2\,(1+x)}$$

In[37]:=
```
Together[f]
```

Out[37]=

$$\frac{2-3x+x^3}{(-3+x)^2\,(1+x)}$$

In[38]:=
```
Apart[%]
```

Out[38]=
$$1 + \frac{5}{(-3+x)^2} + \frac{19}{4(-3+x)} + \frac{1}{4(1+x)}$$

In[39]:=
```
Factor[%]
```

Out[39]=
$$\frac{(-1+x)^2(2+x)}{(-3+x)^2(1+x)}$$

In[40]:=
```
Numerator[%]
```

Out[40]=
$$(-1+x)^2(2+x)$$

In[41]:=
```
Expand[%]
```

Out[41]=
$$2 - 3x + x^3$$

In[42]:=
```
x / %
```

Out[42]=
$$\frac{x}{2 - 3x + x^3}$$

In[43]:=
```
Denominator[%]
```

Out[43]=
$$2 - 3x + x^3$$

Kuvat

Yksi *Mathematica*n vahvoja ominaisuuksia on sen sisäänrakennettu kyky tuottaa kuvia. Kaksi- ja kolmiulotteisten kuvien tekemiseen löytyy monia komentoja. Näistä yksinkertaisin, mutta hyvin käyttökelpoinen, on **`Plot`.**

Esimerkkinä harjoitus suhteellisuusteorian kurssista, jossa piti tutkia suureen

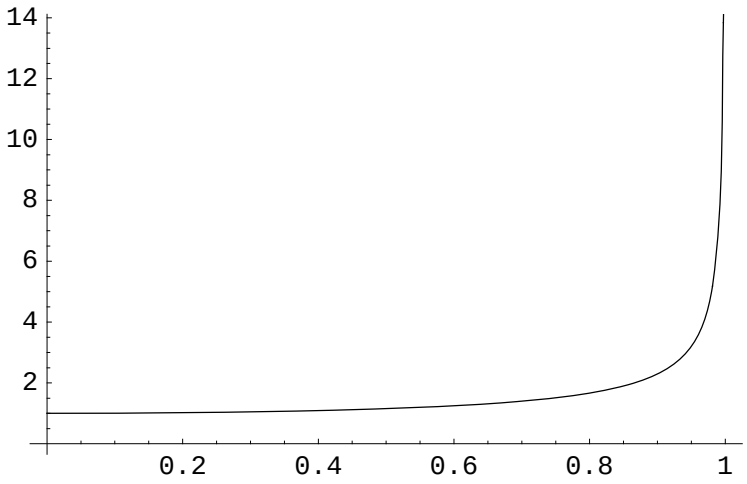
In[44]:=
```
c =.
v =.
```

In[46]:=
```
1 / Sqrt[1 - (v / c)^2]
```

Out[46]=
$$\frac{1}{\sqrt{1-\frac{v^2}{c^2}}}$$

käyttäytymistä suhteellisen nopeuden $a = v/c$ funktiona. Tämän teemme piirtämällä kuvan.

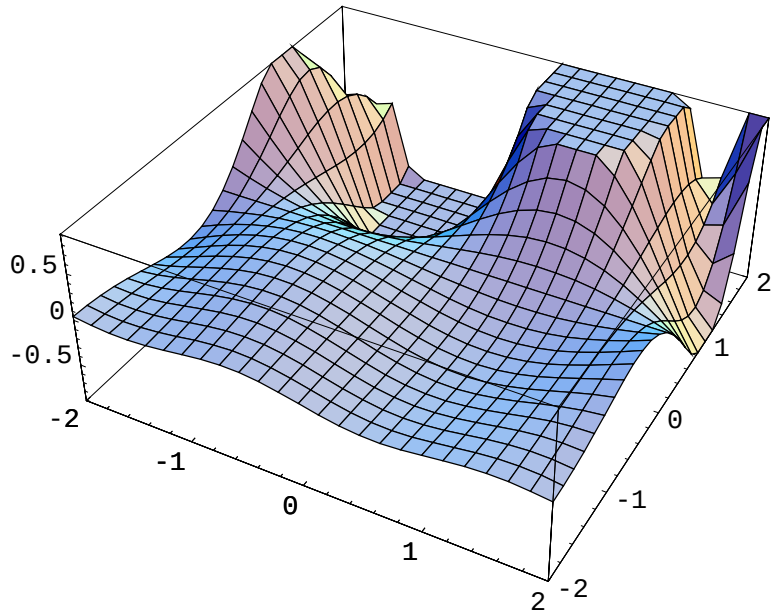
In[47]:=
```
Plot[1 / Sqrt[1 - a^2], {a, 0, 1}]
```



Out[47]=
```
- Graphics -
```

Muita kuvanpiirtokomentoja ovat **Plot3D,** joka piirtää kolmiulotteisen kuvan...

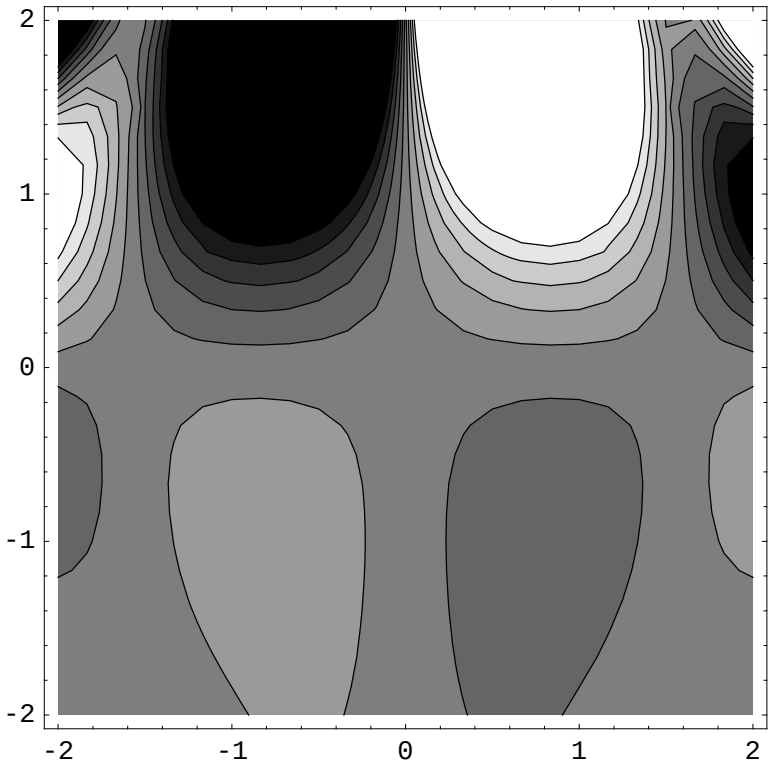
In[48]:=
```
Plot3D[Sin[x y] Cos[x] Exp[y], {x, -2, 2}, {y, -2, 2}]
```



Out[48]=
```
- SurfaceGraphics -
```

...sekä **ContourPlot**, joka piirtää tasa-arvokäyrän

```
In[49]:= ContourPlot[Sin[x y] Cos[x] Exp[y], {x, -2, 2}, {y, -2, 2}]
```




```
Out[49]= - ContourGraphics -
```

Apua

Apua lyötyy HELP valikosta. Koko *Mathematica* kirja on käytössä hypertekstien avulla.

Optiot

Monet *Mathematica*n komennoista hyväksyvät erilaisia optioita eli valitsimia, jotka määräävät esimerkiksi numeerisen tarkkuuden, kuvioiden piirtovärin ja -tarkkuuden sekä muita enemmän tai vähemmän hyödyllisiä asioita. Jonkin komennon optiot saa selville komennolla **`Options`**.

Esimerkiksi **`Plot3D`** komennon valitsimet listautuvat alla olevalla komennolla

```
In[50]:= Options[Plot3D]
```

```
Out[50]= {AmbientLight → GrayLevel[0], AspectRatio → Automatic, Axes → True,
 AxesEdge → Automatic, AxesLabel → None, AxesStyle → Automatic,
 Background → Automatic, Boxed → True, BoxRatios → {1, 1, 0.4},
 BoxStyle → Automatic, ClipFill → Automatic, ColorFunction → Automatic,
 ColorFunctionScaling → True, ColorOutput → Automatic,
 Compiled → True, DefaultColor → Automatic, DefaultFont :→ $DefaultFont,
 DisplayFunction :→ $DisplayFunction, Epilog → {}, FaceGrids → None,
 FormatType :→ $FormatType, HiddenSurface → True, ImageSize → Automatic,
 Lighting → True, LightSources → {{{1., 0., 1.}, RGBColor[1, 0, 0]},
   {{1., 1., 1.}, RGBColor[0, 1, 0]}, {{0., 1., 1.}, RGBColor[0, 0, 1]}},
 Mesh → True, MeshStyle → Automatic, Plot3Matrix → Automatic,
 PlotLabel → None, PlotPoints → 25, PlotRange → Automatic,
 PlotRegion → Automatic, Prolog → {}, Shading → True, SphericalRegion → False,
 TextStyle :→ $TextStyle, Ticks → Automatic, ViewCenter → Automatic,
 ViewPoint → {1.3, -2.4, 2.}, ViewVertical → {0., 0., 1.}}
```

■ Tehtäviä

6. Laske $\sin(\pi)$ ja $\cos(\pi)$.

7. Laske $\sqrt{-1}$, i^2, i^3, i^4.

8. Ratkaise yhtälö $z^4 = -1$.

9. Piirrä käyrä $y = x$ välillä [-10,2].

10. Hae näkyville **Sin**-komennon syntaksi.

11. Kokeile mitä **??** tekee! Käytä esimerkkinä vaikka **Plot**-komentoa.

12. Tutki, osaako Mathematica antaa tietoja jonkin komennon optioiden käytöstä **?**-komennon avulla.

Ohjelmat

Mathematica-ohjelmia käsitellään tarkemmin luvussa Ohjelmointi. Mainittakoon kuitenkin, että *Mathematica*-komentoja voi kirjoittaa tiedostoon, jonka *Mathematica* sitten suorittaa.

Funktiot

Valmiit funktiot

Mathematicassa ovat kaikki matematiikan standardifunktiot.

Abs[x], **Sqrt[x]**, **Exp[x]**, **Log[x]**, **Log[a, x]**

Sin[x], **Cos[x]**, **Tan[x]**, **Cot[x]**, **Sec[x]**, **Csc[x]**

ArcSin[x], **ArcCos[x]**, **ArcTan[x]**, **ArcCot[x]**, **ArcSec[x]**, **ArcCsc[x]**

Sinh[x], **Cosh[x]**, **Tanh[x]**, **Coth[x]**, ...

ArcSinh[x], **ArcCosh[x]**, **ArcTanh[x]**, **ArcCoth[x]**, ...

n!, **n!!**, **Binomial[n,m]**

Max[k, l ,m, ...], **Min[k, l, m, ...]**

Round[x], **Floor[x]**, **Ceiling[x]**, **Sign[x]**

Re[z], **Im[z]**, **Conjugate[z]**, **Abs[z]**, **Arg[z]**

Random[], **SeedRandom[]**

...

Käytettävissä on myös suuri joukko erikoisfunktioita

LegendreP[n, x]

SphericalHarmonicY[l, m, theta, phi]

ChebyshevT[n, x]

HermiteH[n, x]

LaguerreL[n, x]

JacobiP[n, a, b, x]

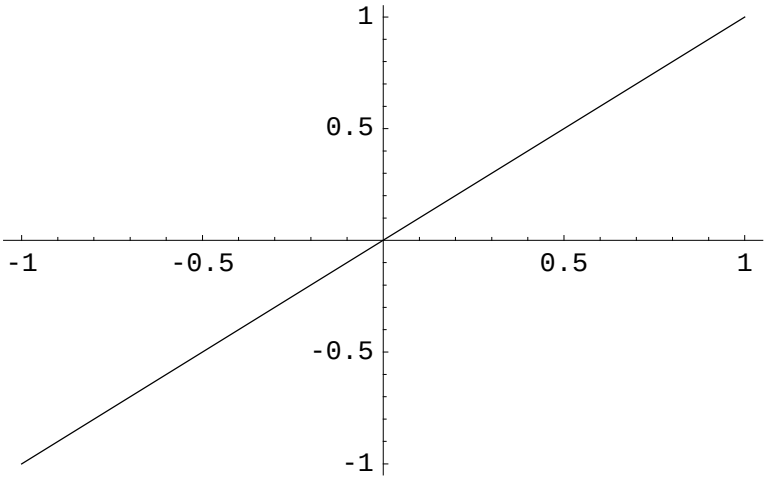
BesselI[n, z]

BesselJ[n, z]

```
Zeta[s, a]
```

Esimerkkinä piirretään Chebyshevin polynomi T_1

```
In[1]:= Plot[ChebyshevT[1, x], {x, -1, 1}]
```




```
Out[1]= - Graphics -
```

Omat funktiot

Mathematicassa määritellään funktio muodossa

```
nimi[x_, y_, z_, …] := lauseke
```

tai

```
nimi[x_, y_, z_, …] = lauseke
```

Sijoitusperaattorit **:=** ja **=** toimivat samoin kuin aikaisemmin kuvattiin. Edellinen on viivästetty sijoitus ja jälkimmäinen suora sijoitus. Funktion nimi on **nimi**, ja argumentit **x**, **y**, **z**,... luetellaan alaviivalla varustettuna. Alaviivaa käytetään muuttujien nimeämisessä, mutta sitä ei käytetä, kun lauseketta muodostetaan argumenttien avulla.

Esimerkkinä määritellään funktio

$$f(x, y) = x^3 + 4\sin(\pi / x) + y$$

ja lasketaan sen arvo pisteessä $(3, 1)$.

```
In[2]:= Clear[f]
```

```
In[3]:= f[x_, y_] := x^3 + 4 Sin[Pi / x] + y
```

```
In[4]:= f[3, 1]
```

```
Out[4]= 28 + 2 √3
```

Funktion arvon jollakin muutujan arvolla saa laskettua sijoittamalla haluttu arvo argumentiksi.

■ Tehtäviä

13. Määrittele funktio $f(x) = x^2 + 2x$. Laske funktion arvo kun $x = 3$.

14. Etsi edellä määritellyn funktion nollakohdat.

15. Määrittele $f(x) = x^4 - 3x + e^{-x}$ *Mathematica*ssa ja piirrä sen kuvaaja välillä [-3,3].

16. Määrittele $g(x) = x^2 - 3x$ ja ratkaise yhtälö $g(x) = 4$.

17. Määrittele funktio $z(x, y) = \sin(x y)\cos(x y)$ ja piirrä se 3D-kuvana ja tasa-arvokäyrinä.

■ Kertaustehtäviä

K1. Määrittele funktio $f(x) = x^5 - 3x^4 + 12x^2 + 3$.

K2. Piirrä ylläolevan funktion kuvaaja välillä [-5,3]

K3. Ratkaise funktion nollakohdat (Älä säikähdä omituisen näköistä vastausta. Se muuttuu mukavamman näköiseksi komennolla **N[%]**. Kts. helppi-tiedostosta lisätietoja **Solve**sta, jos haluat tietää mitä vastauksen merkinnät tarkoittivat). Tarkista lisäksi toteuttaako reaalinen ratkaisu yhtälön $f(x) = 0$(tästä huomaa, että komento **N[%]** todella ottaa numeerisen arvon nollakohdista).

Yhtälöt

Polynomiyhtälöt

Mathematica osaa ratkaista `Solve` komennolla *polynomiyhtälöitä*. Tarkat ratkaisut löytyvät kaikille astetta $k \leq 4$oleville polynomiyhtälöille, mutta joskus ratkaisut ovat hyvin monimutkaisen näköisiä. Numeerisesti samat yhtälöt voidaan ratkaista komennolla **NSolve**. **Solve**-komennossa annetaan ratkaistava yhtälö (tai joukko yhtälöitä pilkulla erotettuna aaltosuluissa), ja muuttuja (tai joukko muuttujia pilkulla erotettuna aaltosuluissa), jonka suhteen ratkaisu halutaan.

Esimerkiksi ratkaise yhtälöryhmä

$$x + y = 0$$
$$x - 2y = 3$$

```
In[1]:= Solve[{x + y == 0, x - 2 y == 3}, {x, y}]
```

```
Out[1]= {{x → 1, y → -1}}
```

In[2]:=
```
NSolve[{x + y == 0, x - 2 y == 3}, {x, y}]
```

Out[2]=
```
{{x → 1., y → -1.}}
```

Reduce-komento ratkaisee samat yhtälöt kuin **Solve,** mutta esittää tuloksen mahdollisimman yleisessä muodossa. Toisin kuin **Solve**, säännön sijasta **Reduce** antaa tuloksena uuden yhtälön.

Esimerkiksi ratkaistaan yleinen toisen asteen yhtälö $a\,x^2 + b\,x + c = 0$.

In[3]:=
```
a =.
b =.
c =.
```

In[6]:=
```
Solve[a x^2 + b x + c == 0, x]
```

Out[6]=
$$\left\{\left\{x \to \frac{-b - \sqrt{b^2 - 4\,a\,c}}{2\,a}\right\}, \left\{x \to \frac{-b + \sqrt{b^2 - 4\,a\,c}}{2\,a}\right\}\right\}$$

In[7]:=
```
Reduce[a x^2 + b x + c == 0, x]
```

Out[7]=
$$a \neq 0 \&\& \left(x == \frac{-b - \sqrt{b^2 - 4\,a\,c}}{2\,a} \,||\, x == \frac{-b + \sqrt{b^2 - 4\,a\,c}}{2\,a}\right) ||$$
$$a == 0 \&\& b \neq 0 \&\& x == -\frac{c}{b} \,||\, c == 0 \&\& b == 0 \&\& a == 0$$

Tuloksen esittämisessä **Reduce** käyttää *loogisia operaattoreita*, jotka ovat samoja kuin C-kielessä.

!	ei
||	tai
&&	ja

Algebrallisen yhtälön numeerinen ratkaiseminen

Yhtälöt, jotka eivät ole polynomiyhtälön muotoa, joudutaan yleensä ratkaisemaan numeerisesti. **NSolve**-komentoa käytetään tyypillisesti polynomiyhtälöiden numeeriseen ratkaisemiseen, kuten edellä esitettiin, mutta **FindRoot**-komennolla voidaan ratkaista yleisempiä yhtälöitä.

Muotoa

```
FindRoot[vasen==oikea, {x,x0}]
```

käytetään silloin, kun yhtälön voi derivoida symbolisessa muodossa (Newtonin menetelmä). Symboli **x0** on ensimmäinen arvaus ratkaisulle. Sen keksimisessä on suureksi avuksi, kun piirtää **Plot**-komennolla sekä vasemman että oikean puolen kuvaajan ja etsii niiden leikkauspisteitä. **FindRoot** löytää yleensä arvausta lähinnä olevan ratkaisun iteroimalla eli parantamalla arvausta kierros kierrokselta.

Ratkaistaan esimerkkinä yhtälö $\tan(x) = x$.

```
In[8]:= FindRoot[Tan[x] == x, {x, 0.5}]
Out[8]= {x → 1.24204 × 10^-8}
```

Tarkka ratkaisu on $x = 0$, joten tulos tuntuu epätarkalta, mutta oikean ja vasemman puolen erotus on kuitenkin hyvin pieni

```
In[9]:= Tan[x] - x /. %
Out[9]= 0.
```

Tarkkuutta voidaan lisätä kasvattamalla parametrin **AccuracyGoal** arvoa ja lisäämällä iteraatioiden lukumäärää **MaxIterations** parametrin avulla.

```
In[10]:= FindRoot[Tan[x] == x, {x, 0.5}, AccuracyGoal -> 15, MaxIterations -> 50]
Out[10]= {x → 1.24204 × 10^-8}

In[11]:= Tan[x] - x /. %
Out[11]= 0.
```

Toinen mahdollisuus on etsiä ratkaisua käyttämällä kahta lähtöpistettä ja niiden kautta piirrettyä suoraa (sekanttimenetelmä). Uusi parempi arvaus saadaan etsimällä suoran nollakohta ja käyttämällä sitä seuraavalla iterointikierroksella toisena lähtöpisteistä. Kaikki tämä tapahtuu tietysti **FindRoot**-komennon sisällä, ja ulos tulee vain ratkaisu. Näin täytyy menetellä silloin, kun yhtälön derivaattaa ei ole mahdollista laskea symbolisessa muodossa.

```
FindRoot[vasen==oikea, {x, x1, x2}]
```

Ratkaistaan sama yhtälö kuin edellä $\tan(x) = x$.

```
In[12]:= FindRoot[Tan[x] == x, {x, -0.1, 0.5}]
```

```
Out[12]= {x → 1.70591 × 10^-8}
```

Lineaarinen yhtälöryhmä

Lineaariset yhtälöryhmät muodostavat oman erikoisryhmänsä, joiden ratkaisemisessa kannattaa käyttää **LinearSolve**-komentoa.

```
LinearSolve[M, b]
```

Ratkaisuna saadaan vektori **x**, joka on matriisiyhtälön

$$M\ x = b$$

ratkaisu. Lineaarisen yhtälön kertoimet täytyy siis antaa matriisimuodossa ja yhtälön oikea puoli vektorimuodossa. Esimerkiksi ratkaistaan lineaarinen yhtälöryhmä

$$\begin{aligned} 3\,x_1 + x_2 &= 2 \\ x_2 + 2\,x_3 &= 2 \\ x_1 + x_3 &= 1 \end{aligned}$$

Tämä on saatettava matriisimuotoon, ennenkuin yhtälö voidaan ratkaista. Kunhan puuttuvat tekijät (nollat ja ykköset) ovat täytetty, kerroinmatriisi voidaan poimia tuntemattomien muuttujien kertoimista:

$$\begin{aligned} 3\,x_1 + 1\,x_2 + 0\,x_3 &= 2 \\ 0\,x_1 + 1\,x_2 + 2\,x_3 &= 2 \\ 1\,x_1 + 0\,x_2 + 1\,x_3 &= 1 \end{aligned} \quad \Longleftrightarrow \quad \underbrace{\begin{pmatrix} 3 & 1 & 0 \\ 0 & 1 & 2 \\ 1 & 0 & 1 \end{pmatrix}}_{M} \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}}_{b}$$

Yhtälön kerroinmatriisi syötetään Mathematicalle muodossa

```
In[13]:= M = {{3, 1, 0}, {0, 1, 2}, {1, 0, 1}}
          b = {2, 2, 1}
```

```
Out[13]= {{3, 1, 0}, {0, 1, 2}, {1, 0, 1}}
```

```
Out[14]= {2, 2, 1}
```

In[15]:=
```
MatrixForm[M]
```

Out[15]//MatrixForm=

$$\begin{pmatrix} 3 & 1 & 0 \\ 0 & 1 & 2 \\ 1 & 0 & 1 \end{pmatrix}$$

In[16]:=
```
MatrixForm[b]
```

Out[16]//MatrixForm=

$$\begin{pmatrix} 2 \\ 2 \\ 1 \end{pmatrix}$$

In[17]:=
```
LinearSolve[M, b]
```

Out[17]=

$$\left\{\frac{2}{5}, \frac{4}{5}, \frac{3}{5}\right\}$$

■ Tehtäviä

18. Ratkaise yleinen kolmannen asteen yhtälö $a\,x^3 + b\,x^2 + c\,x + d = 0$ käyttämällä **Reduce**-komentoa.

19. Etsi yhtälön $\cos(x) = x$ ratkaisu (Vinkki: piirrä ensin kuvaaja $\cos(x) - x$ vaikkapa välillä [-6,6], niin näet mihin nollakohta asettuu.).

20. Ratkaise seuraava yhtälöryhmä

$$3\,x + y = 2$$
$$y + 2\,z = 2$$
$$x + z = 1$$

symbolisesti ja numeerisesti.

21. Etsi yhtälöparille

$$x^2 + 3\,x - 1 = 0$$
$$3\,x^2 + 5\,x + 1 = 0$$

symboliset ja numeeriset ratkaisut.

22. Ratkaise yhtälöryhmä

$$x^2 + 5\,x - 4\,y = 0$$
$$2\,x - y = 3$$

23. Ratkaise lineaarinen yhtälöryhmä

$$3\,x_1 + x_2 + 4\,x_3 = 2$$
$$5\,x_1 + x_2 + 2\,x_3 = 3$$
$$x_1 - 3\,x_2 - 2\,x_3 = 1$$

Taulukoiden muodostaminen ja interpolointi

Fysiikassa ja kemiassa tulokset esitetään usein taulukon muodossa ja sen jälkeen taulukon arvoille yritetään tehdä analyysiä matemaattisia menetelmiä ja teorioita käyttäen. Tällöin joudutaan arvioimaan funktion käyttäytymistä pisteissä, joista ei ole mittaustuloksia. Mikäli kyseessä on mittauspisteiden välissä oleva piste, niin puhutaan interpoloinnista, muulloin suoritetaan ekstrapolointia.

Taulukon muodostaminen

Ajatellaan, että mittaus on tuottanut lukupareja (x_i, y_i), $i = 1, \ldots, 5$. Mathematicassa nämä parit kirjoitetaan listaksi

```
In[1]:= taulu = {{1, 3}, {2, 5}, {3, 7}, {4, 9}, {5, 12}}

Out[1]= {{1, 3}, {2, 5}, {3, 7}, {4, 9}, {5, 12}}
```

```
In[2]:= TableForm[taulu]
```

```
Out[2]//TableForm=
1    3
2    5
3    7
4    9
5    12
```

Listan voi tulostaa taulukkomuodossa komennolla **TableForm**. Listan alkiot voi kertoa ja jakaa luvulla. Niistä voidaan vähentää ja niihin lisätä lukuja, mutta nämä laskutoimitukset kohdistuvat kaikkiin alkioihin.

Esimerkiksi

```
In[3]:= taulu + 1
```

```
Out[3]= {{2, 4}, {3, 6}, {4, 8}, {5, 10}, {6, 13}}
```

```
In[4]:= taulu * 2
```

```
Out[4]= {{2, 6}, {4, 10}, {6, 14}, {8, 18}, {10, 24}}
```

Jos pelkät y_i:n arvot halutaan kertoa jollakin luvulla, niin silloin täytyy turvautua kahden listan kertolaskuun. Eli listan alkiot kerrotaan toisen listan vastinalkioilla. Esimerkkinä kerrotaan listassa **taulu** olevat y_i:n arvot π:llä.

```
In[5]:= kerroin = Table[{1, Pi}, {i, 5}]
```

```
Out[5]= {{1, π}, {1, π}, {1, π}, {1, π}, {1, π}}
```

```
In[6]:= kerroin * taulu
```

```
Out[6]= {{1, 3 π}, {2, 5 π}, {3, 7 π}, {4, 9 π}, {5, 12 π}}
```

Komennolla **Table** muodostetaan taulukko **kerroin**, jossa on viisi alkiota $\{1,\pi\}$. Yleisimmässä muodossa **Table** komennolla

```
Table[lauseke, {x, min, max, askel}]
```

muodostetaan taulukko, jossa **x** saa arvot parametrin **min** arvosta parametrin **max** arvoon saakka, kun askeleen pituus on parametrin **askel** arvo. Lausekkeen lauseke arvo laskataan kullakin **x**:n arvolla ja nämä arvot muodostavat listan alkiot.

Taulukon alkioita voidaan käsitellä myös alkio kerrallaan. Kuten edellä mainittiin, yksittäiseen alkioon viitataan kaksinkertaisia hakasulkeita käyttäen. Esimerkiksi muodossa

taulu[[i]]

Taulukon **taulu** y:n arvojen kertominen π:llä voisi tapahtua myös seuraavasti

```
In[7]:= maxalkio = Length[taulu]
        Table[{taulu[[i, 1]], Pi * taulu[[i, 2]]}, {i, maxalkio}]

Out[7]= 5

Out[8]= {{1, 3 π}, {2, 5 π}, {3, 7 π}, {4, 9 π}, {5, 12 π}}
```

Komento **Length[taulu]** kertoo kuinka monta alkiota taulukossa on.

Mathamaticassa voidaan dataa lukea myös tiedostosta **ReadList**-komennolla.

Komennolla

datat = ReadList["file", {Number, Number}]

luetaan tiedostosta **file** kahteen sarakkeeseen sijoitettuja lukuja taulukkoon **datat**. Lista **{Number,Number}** kertoo, missä muodossa taulukon sarakkeet ovat talletettu tiedostoon.

Taulukon alkioita voidaan lisätä, poistaa tai muuttaa seuraavilla komennoilla

Prepend[lista, alkio]	lisää alkio listan alkuun
Append[lista, alkio]	lisää alkio listan loppuun
Insert[lista, alkio, i]	lisää alkio listaan paikkaan i
Insert[lista, alkio, -i]	lisää alkio listaan paikkaan i lopusta lukien
Delete[lista, {i, j, …}]	poista alkiot i, j, … listasta
ReplacePart[lista, alkio, i]	korvaa paikassa i oleva alkio

Interpolointi

Kun funktion arvo tunnetaan joukossa pisteitä, $(y_i = f(x_i), i = 1, ..., n)$, ja halutaan laskea funktion arvo jossain muussa pisteessä x tarvitaan interpolointia.

■ Interpoloiva polynomi

Kun funktion arvo tunnetaan n:ssä pisteessä, voidaan sitä interpoloida $(n-1)$:n asteen polynomilla, joka kulkee kaikkien annettujen pisteiden kautta. Mathematicassa on käsky

InterpolatingPolynomial[datat, x],

jolla tämä tehdään.

Testataan interpoloinnin tarkkuutta laskemalla $\sin(x)$ arvot pisteissä $x = 0, 1, 2, \ldots, 10$ ja interpoloimalla saatua pistejoukkoa polynomilla.

In[9]:=
```
datat = Table[{x, Sin[x]}, {x, 0, 10}] // N
```

Out[9]=
```
{{0., 0.}, {1., 0.841471}, {2., 0.909297}, {3., 0.14112},
 {4., -0.756802}, {5., -0.958924}, {6., -0.279415},
 {7., 0.656987}, {8., 0.989358}, {9., 0.412118}, {10., -0.544021}}
```

In[10]:=
```
TableForm[%]
```

Out[10]//TableForm=

0.	0.
1.	0.841471
2.	0.909297
3.	0.14112
4.	−0.756802
5.	−0.958924
6.	−0.279415
7.	0.656987
8.	0.989358
9.	0.412118
10.	−0.544021

In[11]:=
```
poly = InterpolatingPolynomial[datat, x]
```

Out[11]=
```
(0.841471 + (-0.386822 +
      (-0.0103932 + (0.0320258 + (-0.00541109 + (-0.000152322 + (0.000138457 +
                      (-0.0000134113 + (-3.97978 × 10^-7 + 1.73593 × 10^-7 (-9. + x))
                          (-8. + x)) (-7. + x)) (-6. + x)) (-5. + x))
              (-4. + x)) (-3. + x)) (-2. + x)) (-1. + x)) (0. + x)
```

In[12]:=
```
Expand[%]
```

Out[12]=
$$0. + 0.992007\,x + 0.0277804\,x^2 - 0.207275\,x^3 + 0.0330457\,x^4 - 0.00829731\,x^5 + 0.00541063\,x^6 - 0.00134378\,x^7 + 0.000151942\,x^8 - 8.20968 \times 10^{-6}\,x^9 + 1.73593 \times 10^{-7}\,x^{10}$$

In[13]:=
```
poly /. x -> 1
```

Out[13]=
```
0.841471
```

In[14]:=
```
Sin[1.5] - poly /. x -> 1.5
```

Out[14]=
```
-0.0000120561
```

```
In[15]:= Sin[15] - poly /. x -> 15 // N
```

```
Out[15]= -632.37
```

Polynomi on siis tarkka tunnetuissa pisteissä ja interpoloi pistejoukkoa melko tarkasti, mutta ekstrapolointia ei kannata käyttää tässä tapauksessa.

■ Paloittain kuutiollinen sovitus

Usein käytännössä joudutaan interpolaatiota kuitenkin tekemään suurelle joukolle pisteitä, jolloin olisi käytettävä korkean asteen interpolaatiopolynomia. Laskennallisten ja laskentatarkkuuteen liittyvien ongelmien vuoksi on parempi käyttää paloittaista alemman kertaluvun interpolaatiopolynomia. Paloittain kuutiollinen sovitus saadaan komennolla **`Interpolation[data]`**.

```
In[16]:= inter = Interpolation[datat]
```

```
Out[16]= InterpolatingFunction[{{0., 10.}}, <>]
```

Polynomin astetta voi muuttaa parametrilla **`InterpolationOrder → n`**.

```
In[17]:= inter = Interpolation[datat, InterpolationOrder -> 2]
```

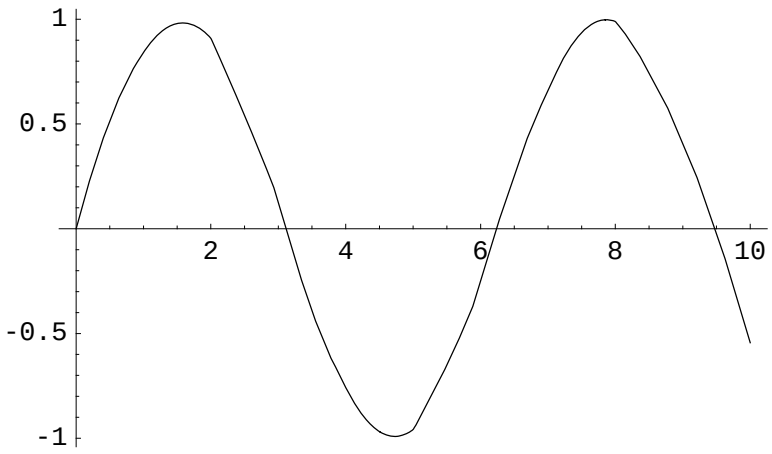
```
Out[17]= InterpolatingFunction[{{0., 10.}}, <>]
```

Tuloksena syntyi funktio, **`InterpolatingFunction[{{0,10.}},"<>"]`**, jonka määrittelyalue on **`{0, 10.}`**. Se sisältää riittävät tiedot interpoloivan funktion muodostamiseen kaikissa tarpeellisissa tilanteissa, kuten arvojen laskemisessa, piirtämisessä, derivoinnissa jne.

```
In[18]:= inter[1.5]
```

```
Out[18]= 0.979885
```

```
In[19]:= Plot[inter[x], {x, 0, 10}]
```




```
Out[19]= - Graphics -
```

```
In[20]:= Sin[1.5] - inter[1.5] // N
```

```
Out[20]= 0.0176103
```

Huomaat, että **Interpolation** ja **InterpolatingPolynomial** eivät anna yhtä tarkkaa tulosta pisteessä 1.5.

■ Spline sovitus

Usein käytetty interpolointimenetelmä on sovittaa kolmannen asteen spline (cubic spline) datajoukkoon. Se määritellään polynomina

$$s(x) = y_i + b_i\,(x - x_i) + c_i\,(x - x_i)^2 + d_i(x - x_i)^3$$

kun $x_i < x < x_{i+1}$

- Derivaatat $s'(x)$ ja $s''(x)$ ovat jatkuvia kaikissa pisteissä x_i

- $s(x_i) = y_i$

- $s(x_{i+1}) = y_{i+1}$

- Funktio on mahdollisimman sileä, eli integraalilla

$$\int_{x_1}^{x_n} (s''(x))^2\,dx$$

on minimi.

Spline-interpolaatiota ei ole Mathematican peruskäskyjen joukossa, mutta se löytyy paketista **NumericalMath`SplineFit`**. Lisäpaketeissa olevat käskyt saadaan käyttöön komennolla

<<

```
In[21]:= << NumericalMath`SplineFit`
```

```
In[22]:= spline = SplineFit[datat, Cubic]
Out[22]= SplineFunction[Cubic, {0., 10.}, <>]
In[23]:= spline[1.5]
Out[23]= {1.5, 0.994193}
```

Sovituksen tuloksena on funktio **SplineFunction[Cubic, {0.,10.}, <>]**, jonka parametreina ovat pisteet numeroituna järjestyksessään nollasta alkaen. Funktion arvo esitetään (x, y) lukuparina.

■ Tehtäviä

24. Piirrä edellä muodostettua datajoukkoa **datat** interpoloivanpolynomin ja paloittain interpoloivan neliöllisen polynomin kuvaajat.

25. Muodosta erotusfunktio **Sin[x]-IntepolatingPolynomial** ja piirrä sen kuvaaja.

■ Kertaustehtäviä

4. Ratkaise yhtälö $\ln(x) = x - 2$.

5. Ratkaise lineaarinen yhtälöryhmä

$$2x + 3y - 5z = 10$$
$$13x - z = 1$$
$$2y + z = 3$$

6. Taulukoi lukuparit $(x, \ln x)$, kun $x = 1, 2, \ldots, 10$. Muodosta paloittain interpoloiva toisen asteen polynomi ja piirrä sen kuvaaja.

Mittaustulosten käsittelyä

■ Pienimmän neliösumman sovitus

Erilaisia funktioita voidaan sovittaa mittaustuloksiin **Fit**-komennolla.

```
Fit[data, {f1, f2, …}, x]
```

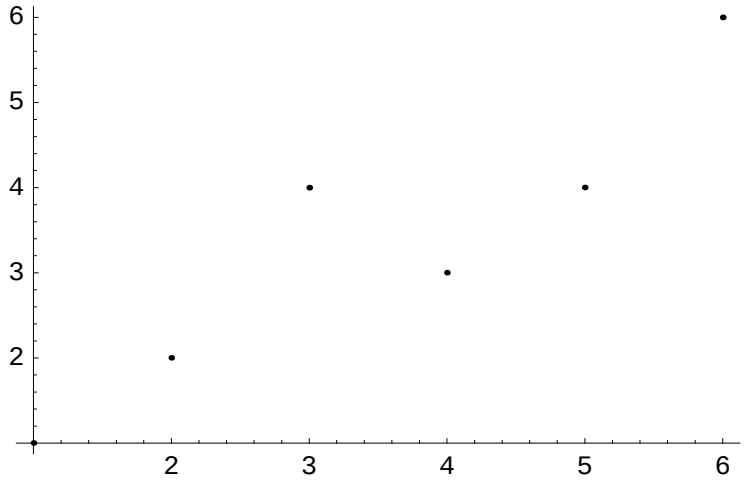
Yllä taulukko **data** pitää sisällään mittauspisteet. Listassa **{f1, f2, …}** annetaan ne funktiot, joiden lineaariyhdistelmänä sovitusta haetaan. Esimerkiksi **{1, x}** tarkoittaa, että haetaan sovitusta muodossa $y = a \cdot 1 + b \cdot x$, toisin sanoen, etsitään suoraa. Vastaavasti antamalla **{1, x, x^2}** saataisiin sovitus paraabelinä, $y = a + b\,x + c\,x^2$.

Esimerkkinä tehdään yksinkertainen pienimmän neliösumman suoran sovitus.

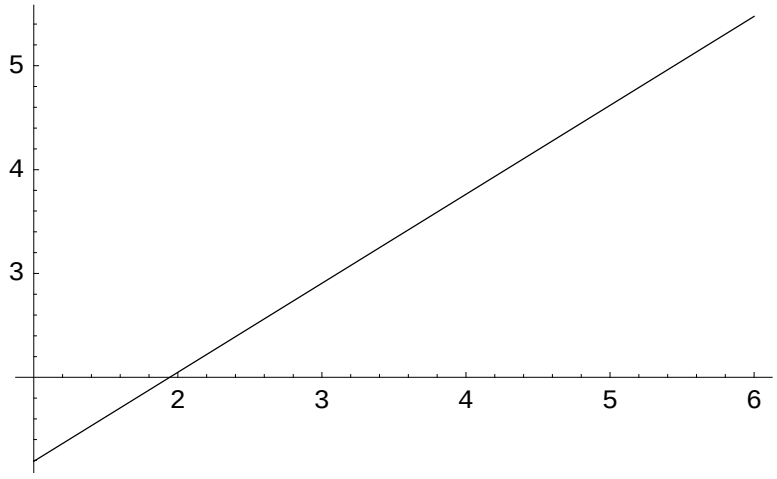
```
In[1]:= arvot = {{1, 1}, {2, 2}, {3, 4}, {4, 3}, {5, 4}, {6, 6}}
Out[1]= {{1, 1}, {2, 2}, {3, 4}, {4, 3}, {5, 4}, {6, 6}}

In[2]:= sovitus = Fit[arvot, {1, x}, x]
Out[2]= 0.333333 + 0.857143 x
```

Piirretään nyt kuvat.

```
In[3]:= ListPlot[arvot]
```



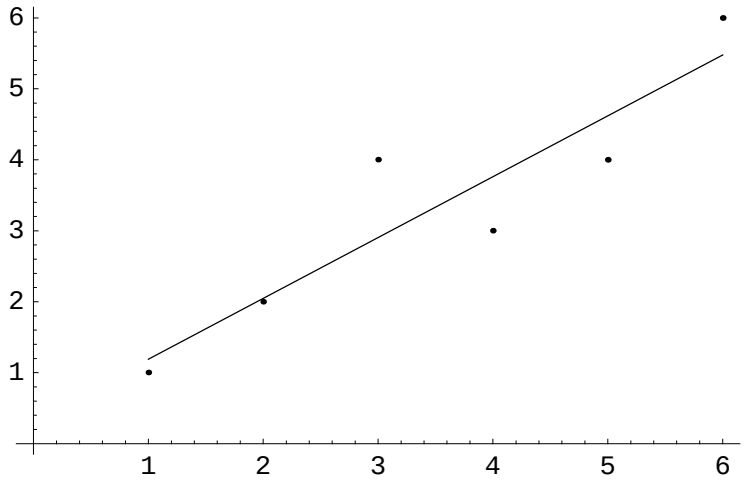

```
Out[3]= - Graphics -

In[4]:= Plot[sovitus, {x, 1, 6}]
```




```
Out[4]= - Graphics -
```

Piirretään nyt kummatkin kuvat samaan koordinaatistoon.

```
In[5]:= Show[%, %%]
```




```
Out[5]= - Graphics -
```

Myös muita funktioita voidaan sovittaa mittaustuloksiin.

Tilastollisen analyysin **Fit**-funktiolla tehdyn sovituksen hyvyydestä saa **Regress**-funktiolla. Komento löytyy **Statistics`LinearRegression`** paketista.

Esimerkiksi

```
In[6]:= << Statistics`LinearRegression`
```

```
In[7]:= arvot = {{1, 1}, {2, 2}, {3, 2}, {4, 3}, {5, 5}, {6, 6}}
```

```
Out[7]= {{1, 1}, {2, 2}, {3, 2}, {4, 3}, {5, 5}, {6, 6}}
```

```
In[8]:= Regress[arvot, {1, x}, x]
```

```
Out[8]= {ParameterTable → 1      Estimate      SE           TStat         PValue
                          x      -0.333333     0.537484     -0.620174     0.568728  ,
                                 1.            0.138013     7.24569       0.00192576

 RSquared → 0.929204, AdjustedRSquared → 0.911504,
 EstimatedVariance → 0.333333, ANOVATable →
              DF     SumOfSq     MeanSq       FRatio     PValue
  Model       1      17.5        17.5         52.5       0.00192576}
  Error       4      1.33333     0.333333
  Total       5      18.8333
```

antaa edellä käytettyihin datoihin **arvot** suoritetun lineaarisen sovituksen tilastollisen analyysin.

■ **Tehtäviä**

26. Sovita seuraavat mittausarvot toisen asteen polynomilla:

    (-2, 6), (-1, 1), (0, 0.5), (1, 2), (2, 4)

    Piirrä lisäksi mittausarvot ja sovitus samaan kuvaan.

27. Tee äskeisen neliöllisen sovituksen tilastollinen analyysi **`Regress`**-funktiolla.

28. Kokeile sovittaa esimerkin suoraa toisen asteen polynomilla. Selitä mitä tapahtui.

Derivointi ja integrointi

Derivointi

Lausekkeen derivaatta saadaan komennolla

```
D[lauseke, muuttuja]
```

tai

```
D[lauseke, {muuttuja, aste}]
```

Derivoitaessa usean muuttujan suhteen voidaan käyttää merkintöjä **`D[lauseke, x1, x2, …]`** tai **`D[lauseke, {x1, aste1}, {x2, aste2}, …]`**.

Esimerkiksi

```
In[1]:= D[x^3 + 3 x^2 - 4 x + 3, {x, 2}]
```

```
Out[1]= 6 + 6 x
```

laskee lausekkeen $x^3 + 3\,x^2 - 4\,x + 3$ toisen derivaatan x:n suhteen. Nopeus v on paikan x aikaderivaatta, $v = \dot{x} = dx/dt$. Kiihtyvyys a puolestaan on nopeuden aikaderivaatta, eli paikan toinen aikaderivaatta, $a = \ddot{x} = \dot{v} = dv/dt$. Yläpisteellä merkitään aikaderivaattaa, $\dot{x} = dx/dt$.

■ Tehtäviä

29. Derivoi $x^2 + 4\,x$.

30. Derivoi $x^4 - 3\,x - \sin(x)$ kaksi kertaa.

31. Kappaleen paikka saadaan lausekkeesta $x = 40 - 5\,t - 5\,t^2$. Mikä on kappaleen kiihtyvyys hetkellä $t = 2$?

32. Piirrä edellisen tehtävän paikka ja nopeus ajan funktiona.

33. Laske funktion $f(x, y) = x^3\,y^4$ osittaisderivaatat

$$\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y},$$

$$\frac{\partial^2 f}{\partial x^2}, \frac{\partial^2 f}{\partial y^2},$$

$$\frac{\partial^2 f}{\partial x\,\partial y}, \frac{\partial^2 f}{\partial y\,\partial x}$$

Kokonaisdifferentiaali ja virheenarviointi

Mathematica osaa laskea myös kokonaisdifferentiaalit näppärästi komennolla **`Dt`**.

`Dt[f]`	laskee kokonaisdifferentiaalin $d\,f$.
`Dt[f, x]`	laskee kokonaisderivaatan $d\,f\,/\,d\,x$.
`Dt[f, {x, n}]`	laskee astetta **`n`** olevan kokonaisderivaatan

Tutkitaan lausekkeen

$$f(a, b, c) = \frac{\pi\,a^2\,b^3}{\sqrt{c}}$$

derivaattoja ja kokonaisdifferentiaaleja.

```
In[2]:=
Clear[a, b, c]
f = Pi a^2 b^3 / Sqrt[c]
```

Out[3]= $\frac{a^2\,b^3\,\pi}{\sqrt{c}}$

```
In[4]:=
D[f, a]
```

Out[4]= $\frac{2\,a\,b^3\,\pi}{\sqrt{c}}$

In[5]:=
```
D[f, {a, 2}]
```

Out[5]=
$$\frac{2\,b^3\,\pi}{\sqrt{c}}$$

In[6]:=
```
Dt[f]
```

Out[6]=
$$\frac{2\,a\,b^3\,\pi\,\mathrm{Dt}[a]}{\sqrt{c}} + \frac{3\,a^2\,b^2\,\pi\,\mathrm{Dt}[b]}{\sqrt{c}} - \frac{a^2\,b^3\,\pi\,\mathrm{Dt}[c]}{2\,c^{3/2}}$$

In[7]:=
```
Dt[f, a]
```

Out[7]=
$$\frac{2\,a\,b^3\,\pi}{\sqrt{c}} + \frac{3\,a^2\,b^2\,\pi\,\mathrm{Dt}[b, a]}{\sqrt{c}} - \frac{a^2\,b^3\,\pi\,\mathrm{Dt}[c, a]}{2\,c^{3/2}}$$

In[8]:=
```
Dt[f, {b, 2}]
```

Out[8]=
$$\frac{\pi\,(6\,a^2\,b + 12\,a\,b^2\,\mathrm{Dt}[a, b] + b^3\,(2\,\mathrm{Dt}[a, b]^2 + 2\,a\,\mathrm{Dt}[a, \{b, 2\}]))}{\sqrt{c}} - \frac{\pi\,(3\,a^2\,b^2 + 2\,a\,b^3\,\mathrm{Dt}[a, b])\,\mathrm{Dt}[c, b]}{c^{3/2}} + a^2\,b^3\,\pi\left(\frac{3\,\mathrm{Dt}[c, b]^2}{4\,c^{5/2}} - \frac{\mathrm{Dt}[c, \{b, 2\}]}{2\,c^{3/2}}\right)$$

Laboratoriotöissä joudutaan usein tekemään virheenarviointeja. Tähän tehtävään kokonaisdifferentiaalia voidaan pienellä vaivalla soveltaa. Esimerkkinä olkoon mielivaltainen suure $F(x, y, z)$, jossa x, y ja z ovat kokeellisesti mitattuja muuttujia. F:n kokonaisdifferentiaali on

$$dF = \frac{\partial F}{\partial x}\,dx + \frac{\partial F}{\partial y}\,dy + \frac{\partial F}{\partial z}\,dz.$$

Tutkittavan suureen muutokselle saadaan arvio, kun korvataan sen kokonaisdifferentiaalissa esiintyvät differentiaalit kyseisten muuttujien pienten muutosten suuruuksilla. Esimerkiksi jos x, y ja z muuttuvat Δx, Δy ja Δz:n verran, muuttuu F noin ΔF:n verran:

$$\Delta F \simeq \frac{\partial F}{\partial x}\,\Delta x + \frac{\partial F}{\partial y}\,\Delta y + \frac{\partial F}{\partial z}\,\Delta z$$

Tässä tapauksessa muuttujien poikkeamat tulkitaan virheinä niiden mittauksissa. Kun jokaisen muuttujan antama osuus lasketaan differentiaalin lausekkeessa positiivisena, saadaan arvio virheen ylärajalle:

$$\left|\Delta F\right| \lesssim \left|\frac{\partial F}{\partial x}\,\Delta x\right| + \left|\frac{\partial F}{\partial y}\,\Delta y\right| + \left|\frac{\partial F}{\partial z}\,\Delta z\right|$$

Tehdään Mathematicalla käytännön esimerkki:

■ Esimerkki

Halutaan tietää sylinterin tilavuus. Mitataan sylinterin halkaisija ja korkeus työntömitalla, kukin kolme kertaa. Tulokset olivat

Mittaus	Halkaisija [cm]	Korkeus [cm]
1	12,00	15,00
2	12,01	14,96
3	12,00	15,05

Mittausten keskiarvot saadaan laskettua käyttämällä **Mean**-komentoa, joka löytyy tilastofunktioiden paketista.

```
In[9]:= << Statistics`DescriptiveStatistics`

In[10]:= data1 = {12.00, 12.01, 12.00}

Out[10]= {12., 12.01, 12.}

In[11]:= halkaisija = Mean[data1]

Out[11]= 12.0033

In[12]:= data2 = {15.00, 14.95, 15.05}

Out[12]= {15., 14.95, 15.05}

In[13]:= korkeus = Mean[data2]

Out[13]= 15.
```

d = halkaisija [cm]	h = korkeus [cm]
12,0033	15,00

Sylinterin tilavuus on

$$V = h\,\pi\,(\frac{d}{2})^2,$$

ja mittausten antama tilavuuden virheen arvio saadaan laskemalla halkaisijan ja korkeuden mittausvirheistä aiheutuneiden virheiden itseisarvojen summa,

$$|\Delta V| = \left|\frac{\partial V}{\partial h}\Delta h\right| + \left|\frac{\partial V}{\partial d}\Delta d\right|.$$

Tilavuuden kokonaisdifferentiaali lasketaan seuraavasti:

```
In[14]:=
Clear[h, d, v]
v[d_, h_] := h Pi (d / 2)^2
diffV = Dt[v[d, h]]
```

```
Out[16]=
1/2 d h π Dt[d] + 1/4 d^2 π Dt[h]
```

```
In[17]:=
virhe = Abs[diffV[[1]]] + Abs[diffV[[2]]]
```

```
Out[17]=
1/2 π Abs[d h Dt[d]] + 1/4 π Abs[d^2 Dt[h]]
```

Sijoitetaan halkaisijan ja korkeuden arvoiksi mittausten keskiarvot ja differentiaalien paikalle niiden virheet.

```
In[18]:=
N[virhe /. {Dt[d] -> 0.01, Dt[h] -> 0.05, d -> 12.0033, h -> 15}]
```

```
Out[18]=
8.48619
```

Arvioksi virheelle saamme siten $\Delta V \lesssim \pm 8.49\ \mathrm{cm}^3$.

Tilavuus saadaan käyttämällä määriteltyä funktiota.

```
In[19]:=
N[v[12.0033, 15]]
```

```
Out[19]=
1697.39
```

Nyt voimme kirjoittaa tuloksen $V = 1697 \pm 8\ \mathrm{cm}^3$.

Edellä differentiaalin numeroarvon laskemisessa käytettiin sijoitusoperaattoria `/.` , jolla saatiin differentiaalilausekkeessa esiintyvät muuttujat korvattua numeroarvoilla.

■ Tehtäviä

34. Oletetaan, että eläintarhan apinoiden lukumäärä riippuu banaanien lukumäärästä ja yhden apinan painosta, kuten

```
m[banana_, paino_] = 3*banana/paino
```

Laske apinoiden lukumäärä virherajoineen, kun banaaneita on 100 ± 5 kg ja yhden apinan paino on 7.3 ± 1.2 kg.

Integrointi

Mathematica osaa integroida melko hyvin jopa symbolisesti, mutta on muistettava että jos Mathematica ei osaa laskea esimerkiksi jotain integraalia symbolisesti, se ei tarkoita, etteikö integraalia voisi laskea käsin kohtuullisella vaivannäöllä. Komento

```
Integrate[lauseke, x]
```

integroi annetun lausekkeen **x**:n suhteen. Komento

```
Integrate[f, {x, a, b}]
```

laskee **f**:n määrätyn integraalin $\int_a^b f(x)\,dx$ ja komento

```
Integrate[f, {x, xmin, xmax}, {y, ymin, ymax}]
```

laskee kaksinkertaisen integraalin. Numeerinen integrointi tapahtuu komennolla

```
NIntegrate[f, {x, xmin, xmax}]
```

Numeerinen integrointi onnistuu paljon useammin kuin symbolinen integrointi, mutta se ei tietystikään ole aivan tarkkaa. Useimmissa tapauksissa Mathematica osaa kertoa, jos integroinnilla ei saavuteta haluttua tarkkuutta tai jos näyttää siltä, että integroitava funktio on liian hankala numeerisillekin menetelmille.

■ Tehtäviä

35. Integroi sin(x), kun $0 < x < 2\pi$. Kokeile myös numeerista integrointia.

36. Integroi $\int_a^b \int_d^c \frac{1}{x\,y}\,dx\,dy$.

37. Laske integraali $\int (4x^4 + 4x^3 - 9x^2 - x + 2)\,dx$ ja tarkista saamasi tulos derivoimalla se.

38. Ylinopeutta ajava auto ajaa nopeudella $v = 15\,m/s$. Poliisi ryhtyy ottamaan ajajaa kiinni. Poliisiauto lähtee levosta ja kiihdyttää $a = 10\,m/s^2$. Milloin poliisi saa kaaharin kiinni? Piirrä kummankin auton paikka ajan funktiona.

39. Kappaleen kiihtyvyys a(t) ajan funktiona muuttuu seuraavasti

$a(t) = \arctan(t)$,

kun aika ilmoitetaan sekunteina. Laske, mikä on kappaleen keskimääräinen nopeus 12 ensimmäisen sekunnin aikana, kun liikkeelle lähdetään levosta, ja mikä on sen nopeus 20 sekunnin kuluttua lähdöstä. Piirrä kiihtyvyyden ja nopeuden kuvaajat samaan kuvaan.

Raja-arvot ja sarjat

Raja-arvot

Raja-arvojen laskeminen Mathematicalla tapahtuu komennolla

```
Limit[f[x], x->x0]
```

Tässä lasketaan raja-arvo $\lim_{x \to x_0} f(x)$.

Esimerkiksi funktion $\sin(x)/x$ raja-arvo nollassa:

```
In[1]:= f = Sin[x] / x
```

$$\text{Out[1]=} \quad \frac{\text{Sin}[x]}{x}$$

```
In[2]:= Limit[f, x -> 0]
```

```
Out[2]= 1
```

Taylorin ja Laurentin sarjat

Mathematica osaa kehittää funktioita potenssisarjoiksi. Taylorin ja Laurentin sarjakehitelmät saadaan **`Series`**-komennolla.

Esimerkiksi sinin sarjakehitelmä muuttujan x suhteen pisteen $x = 0$ ympäristössä potenssiin yhdeksän saakka saadaan komennolla:

```
In[3]:= Series[Sin[x], {x, 0, 9}]
```

$$\text{Out[3]=} \quad x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + \frac{x^9}{362880} + O[x]^{10}$$

```
In[4]:= Normal[%]
```

$$\text{Out[4]=} \quad x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + \frac{x^9}{362880}$$

Viimeinen **`O[x]`**^{10}-termi kuvaa yhdeksättä potenssia korkeampia termejä. Ennen kuin Mathematican tekemää sarjakehitelmää voidaan käyttää edelleen (esimerkiksi kuvaajan piirtoon), pitää se muuttaa tavalliseksi Mathematican lausekkeeksi **`Normal`**-komennolla.

Laurentin sarjassa on termejä, jotka ovat äärettömiä siinä pisteessä, jonka ympäristössä kehitelmä on tehty. Esimerkiksi funktion $\frac{e^x}{1-x^2}$ Laurentin sarjakehitelmä pisteen $x = 1$ ympäristössä on

```
In[5]:= Series[E^x / (1 - x^2), {x, 1, 3}]
```

$$\text{Out[5]}= -\frac{e}{2\,(x-1)} - \frac{e}{4} - \frac{1}{8}\,e\,(x-1) - \frac{1}{48}\,e\,(x-1)^2 - \frac{1}{96}\,e\,(x-1)^3 + O[x-1]^4$$

■ Tehtäviä

40. Kehitä kosini sarjaksi yhdeksänteen asteeseen asti. Vertaa sitä esimerkin sinin sarjakehitelmään.

41. Laske yhteen sinin ja kosinin sarjakehitelmien neliöt. Mitä pitäisi tulla tulokseksi ja mitä saat?

42. Piirrä sinin kolmannen, viiden, seitsemännen ja yhdeksännen asteen sarjakehitelmien kuvaajat ja vertaa niitä tarkkaan kuvaajaan.

43. Kehitä sini sarjakehitelmäksi pisteen $x = 1$ ympäristössä kuudennen asteen termejä myöten. Esitä tulos polynomina käyttäen **Expand**-komentoa.

44. Tutki myös jonkin muun funktion sarjakehitelmien tarkkuutta muunnellen astelukua.

45. Kokeile kehittää jokin määrittelemätön funktio **f[x]** sarjakehitelmäksi. Mitä saat tulokseksi?

46. Esimerkkinä singulaarisista eli ns.Laurentin sarjakehitelmistä kehitä funktio $e^x/(x-1)$ sarjaksi pisteen $x = 1$ ympäristössä. Mathematican komento on sama kuin Taylorin sarjakehitelmän tapauksessa.

47. Muunna kehitelmä **Normal**-funktion avulla normaaliin lausekkeen muotoon kuvan piirtämistä

varten ja piirrä kuva.

48. Osoita,että suhteellisuusteorian mukaisesta energian lausekkeesta $E = m\,c^2$ saadaan ei-relativistinen liike-energian lauseke

$$E = m_0\,c^2 + \tfrac{1}{2}\,m_0\,v^2,$$

kun liikemassan m ja lepomassan m_0 välille voidaan johtaa relaatio $m = \frac{m_0}{\sqrt{1-\frac{v^2}{c^2}}}$.

Differentiaaliyhtälöitä *Mathematica*lla

Differentiaaliyhtälöt

Differentiaaliyhtälöt ovat yhtälöitä, joissa esiintyy tuntematon funktio, esimerkiksi $y(x)$, ja sen derivaattoja.

Yksinkertainen esimerkki differentiaaliyhtälöstä on

$$\frac{dy(x)}{dx} = 3\,y(x).$$

Differentiaaliyhtälön ratkaisuksi sanotaan analyyttistä lauseketta tai muuta esitystä y(x):lle. Yhtälön ratkaisu on $y(x) = e^{3x}$, sillä

$$\frac{dy(x)}{dx} = \frac{d}{dx}\,(e^{3x}) = 3\,e^{3x} = 3\,y(x).$$

Differentiaaliyhtälön kertaluku on korkein yhtälössä esiintyvä derivaatan kertaluku. Yo. yhtälö oli ensimmäisen kertaluvun differentiaaliyhtälö. Esimerkki toisen kertaluvun differentiaaliyhtälöstä on

$$\frac{d^2u(x)}{dx^2} + 16\,u(x) = 0,$$

sillä siinä esiintyy toista derivaattaa. Tämän yhtälön ratkaisu on mm. $u(x) = \cos(4\,x)$.

Reunaehdot ja reuna-arvotehtävät

Tarkastele seuraavaa differentiaaliyhtälöä, joka kuvaa populaation määrää x ajanhetkellä t.

$$\dot{x} = \frac{dx}{dt} = x$$

Tämän yhtälön ratkaisu on $x(t) = k\,e^t$. Koska ratkaisussa esiintyy mielivaltainen vakio k, kutsutaan tällaista ratkaisua yleiseksi ratkaisuksi. Jos toisaalta tiedämme, että ajanhetkellä $t = 0$ populaatio on $x(0) = 10$, voimme sijoittaa nämä tiedot yleiseen ratkaisuun ja saamme k:lle arvon $k = 10$. Olemme juuri ratkaisseet reuna-arvotehtävän.

Ratkaisut

Jos differentiaaliyhtälö ratkaistaan ilman reunaehtoja, saadaan ratkaisuun yhtälön kertaluvun osoittama määrä tuntemattomia vakioita. Ratkaisu on siis käyräparvi, jonka parametrien määrä on yhtälön kertaluku.

Lineaarisia differentiaaliyhtälötyyppejä

Lineaarisuus tarkoittaa, että yhtälössä ei esiinny tuntemattoman funktion $y(x)$ tai sen derivaattojen potensseja. Esittelemme muutamia differentiaaliyhtälötyyppejä, joihin tulet törmäämään fysiikassa.

■ Ensimmäisen kertaluvun vakiokertoimiset differentiaaliyhtälöt

Ensimmäisen kertaluvun vakiokertoimiset differentiaaliyhtälöt ovat muotoa

$$y'(x) + a\,y(x) = b,$$

missä a ja c ovat vakioita. Jos $c = 0$ sanotaan yhtälöä *homogeeniseksi*.

■ Ensimmäisen kertaluvun funktiokertoimiset differentiaaliyhtälöt

Muotoa

$$y'(x) + p(x)\,y(x) = q(x)$$

olevat differentiaaliyhtälöt ovat ensimmäisen kertaluvun funktiokertoimisia differentiaaliyhtälöitä. Jos $q(x) = 0$ kaikilla x:n arvoilla on yhtälö *homogeeninen*, jos taas sekä $q(x)$ että $p(x)$ ovat nollasta eroavia, sanotaan yhtälöä *täydelliseksi*.

■ Toisen kertaluvun vakiokertoimiset differentiaaliyhtälöt

Toisen kertaluvun differentiaaliyhtälöissä esiintyy jo tuntemattoman funktion $y(x)$ toista derivaattaa $y''(x)$. Vakiokertoiminen, täydellinen yhtälö on muotoa

$$y''(x) + a\,y'(x) + b\,y(x) = c.$$

Vastaavasti kuin ensimmäisen kertaluvun tapauksessa sanotaan yhtälöä homogeeniseksi, jos $c = 0$.

■ Toisen kertaluvun funktiokertoimiset differentiaaliyhtälöt

Yleinen muoto toisen kertaluvun funktiokertoimiselle differentiaaliyhtälölle on

$$y''(x) + p(x)\,y'(x) + r(x)\,y(x) = q(x).$$

Differentiaaliyhtälöiden ratkaiseminen Mathematicalla

Mathematica on näppärä apuneuvo differentiaaliyhtälöiden ratkaisuun.

Peruskomento on **DSolve**.

```
DSolve[yhtälö, y[x], x]
```

Ratkaisemme esimerkiksi yhtälön

```
In[1]:= DSolve[y'[x] == 3 y[x], y[x], x]
```

```
Out[1]= {{y[x] → ℯ^(3 x) C[1]}}
```

Tulosta pitää osata hieman tulkita, mutta kun hoksaa että siinä esiintyvä **C[1]** on vakio, jota olimme aiemmassa esimerkissä merkinneet k:lla, lauseke näyttää jo tutulta. Huomaa, että jos erehdyt käyttämään **=**-merkkiä **==**-merkin asemesta, määrittelee *Mathematica* **=**-merkin vasemmalla puolella olevan asian oikealla puolella olevan asian kanssa yhtäsuureksi. Jos näin tapahtuu esimerkiksi ratkottaessa differentiaaliyhtälöitä, pitää nämä määrittelyt purkaa **Clear**-komennolla, muuten voi syntyä ihmeellisiä virheitä.

■ Numeerinen ratkaisu

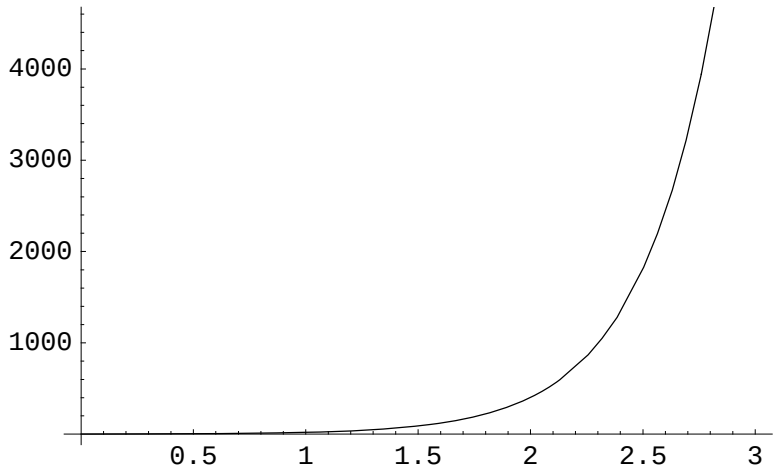
Jos differentiaaliyhtälö on hankala, ei Mathematica välttämättä osaa ratkaista sitä symbolisesti. Voit kuitenkin käyttää komentoa **NDSolve**, joka ratkaisee differentiaaliyhtälön numeerisesti. **NDSolve**lle pitää antaa myös vapaan muuttujan väli, jolle ratkaisu etsitään. Tulos on voimassa vain tällä välillä. Lisäksi **NDSolve** vaatii riittävästi reunaehtoja kyetäkseen ratkaisemaan kaikki vakiot, muuten se ei pysty laskemaan numeerista ratkaisua.

Esimerkiksi edellisen esimerkin yhtälön tapauksessa **NDSolve** ei antaisi tulosta, mutta jos sovimme reunaehdoksi vaikkapa $y(0) = 1$ saamme,

```
In[2]:= ratkaisu = NDSolve[{y'[x] == 3 y[x], y[0] == 1}, y[x], {x, 0, 10}]

Out[2]= {{y[x] → InterpolatingFunction[{{0., 10.}}, <>][x]}}
```

Nyt voimme esimerkiksi piirtää funktion $y(x)$ kuvaajan.

```
In[3]:= Plot[y[x] /. ratkaisu, {x, 0, 3}]
```




```
Out[3]= - Graphics -
```

Kuvaaja voidaan piirtää vain välillä [0, 10]. Huomaa sijoitusoperaattorin **/.** käyttö!

■ Tehtäviä

49. Ratkaise 1:n kertaluvun lineaarinen vakiokertoiminen differentiaaliyhtälö $y'(x) - 3\,y(x) = 4$.

50. Ratkaise 2:n kertaluvun lineaarinen vakiokertoiminen differentiaaliyhtälö $4\,y''(x) - 3\,y'(x) + 5\,y(x) = 2$. Määrää mieleisesi reunaehdot ja piirrä ratkaisun reaaliosan kuvaaja.

■ **Kertaustehtäviä**

7. Integroi $\cos(x)$ välillä $x \in (-10, 10)$ symbolisesti ja numeerisesti.

8. Kirjoita $\tan(x)$:n sarjakehitelmä 5:n ympäristössä. Ota kehitelmään 5 ensimmäistä termiä.

9. Ratkaise differentiaaliyhtälö $12\,x''(t) - 9\,x'(t) + 16\,x(t) - 9 = 0$ alkuarvoilla $x(0) = 2$ ja $x'(0) = 0$. Piirrä ratkaisun kuvaaja, kun t saa arvot $0 \ldots 10$.

10. Mieti, miten saisit näppärästi määriteltyä edellisen tehtävän ratkaisua vastaavan funktion.

Radioaktiivinen hajoaminen

Radioaktiivista hajoamista voidaan kuvata yhtälöllä

$$\frac{dN}{dt} = -\lambda\, N,$$

missä λ on hajoamisvakio.

Hetkellä $t = 0$ meillä on N_0 ydintä. Hetkellä t jäljellä olevien ydinten määrä saadaan selville ratkaisemalla yo. yhtälö alkuehdolla $N(t = 0) = N_0$.

```
In[4]:= ratk = DSolve[{n'[t] == -lambda n[t], n[0] == n0}, n[t], t]
```

```
Out[4]= {{n[t] → ℯ^(-lambda t) n0}}
```

joka voidaan kirjoittaa tutumpaan muotoon

$$N(t) = N_0\, e^{-\lambda t}.$$

Lasketaan vielä puoliintumisaika. Puoliintumisaika on aika, jonka kuluttua puolet ytimistä on hajonnut.

```
In[5]:= Solve[n[t] == n0 / 2 /. ratk, t]
```

```
Out[5]= {{t → Log[2]/lambda}}
```

Voimme myös havainnollistaa ratkaisua piirtämällä siitä kuvan. Ensin annamme λ:lle ja N_0:lle sopivat numeroarvot.

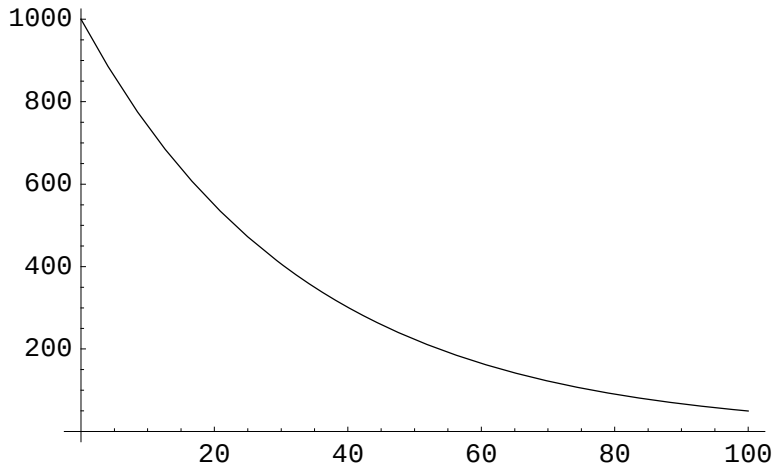
```
In[6]:= n0 = 1000
```

```
Out[6]= 1000
```

```
In[7]:= lambda = 0.03
```

```
Out[7]= 0.03
```

```
In[8]:= Plot[n[t] /. ratk, {t, 0, 100}]
```




```
Out[8]= - Graphics -
```

Huomaa, miten ratkaisuna saamaamme funktioon **n[t]** viitataan sekä **Solve**- että **Plot**-komennoissa.

Ylöspäin heitetty kappale

Haluamme saada matemaattisen kuvauksen suoraan ylöspäin heitetylle pesäpallolle. Tiedämme, että pallon kiihtyvyys on gravitaatiokiihtyvyys. Toisaalta kiihtyvyys on paikan toinen aikaderivaatta,

$$a = g = \frac{d^2y}{dt^2} = \ddot{y}.$$

Lisäksi voimme asettaa koordinaatiston niin, että pallo on lähtöhetkellä origossa. Matemaattisesti tämä voidaan esittää yhtälöparilla

$$a = \ddot{y} \qquad .$$
$$y(t = 0) = 0$$

Katsotaan mitä Mathematica saa näistä aikaiseksi.

```
In[9]:= DSolve[{a == y''[t], y[0] == 0}, y[t], t]
```

$$\left\{\left\{y[t] \to \frac{1}{2}\,(a\,t^2 + 2\,t\,C[2])\right\}\right\}$$

Huomaa, että olimme merkinneet $\ddot{y}$:n t:n funktioksi, vaikka kyseessä oikeastaan olikin vakio a. Huomaa myös, että tuloksessa on ainoastaan yksi tuntematon vakio, **C[2]**, vaikka kyseessä oli toisen kertaluvun differentiaaliyhtälö. Tämä johtuu siitä, että annoimme yhtälön ratkaisun yhteydessä reunaehdon, $y(0) = 0$. Pääsisimme eroon myös **C[2]**:sta, jos meillä olisi vielä yksi lisäehto, esimerkiksi tieto siitä, että pesäpallon alkunopeus oli $10\,m/s$.

■ Tehtäviä

51. Eliminoi esimerkin tehtävästä vakio **C[2]** käyttämällä oletusta, jonka mukaan pesäpallon alkunopeus oli $10\,m/s$ ylöspäin.

52. Mikä on pesäpallon saavuttama maksimikorkeus ja missä ajassa? Milloin pesäpallo palaa heittokorkeudelle?

Putoamisliike vedessä

Tutkitaan nyt hieman monimutkaistettua tilannetta, jossa tiputamme tiiliskiven veneestä. Tiiliskiveen vaikuttaa maan vetovoima $F_G = m\,g$, mutta myös veden aiheuttama liikevastus $F_D = -k\,\dot{y}$. Muotoillaan asia matemaattisesti.

$$F = m\,g - k\,\dot{y} = m\,\ddot{y} = m\,a.$$

Mathematica sanoo

```
In[10]:= Clear[y, m, g, t]
         DSolve[m g - k y'[t] == m y''[t], y[t], t]
```

$$\text{Out[11]= } \left\{\left\{y[t] \to \frac{g\,m\,t}{k} - \frac{e^{-\frac{k\,t}{m}}\,m\,C[1]}{k} + C[2]\right\}\right\}$$

Kun otetaan huomioon järkevät reunaehdot, saadaan tulokseksi

```
In[12]:= DSolve[{m g - k y'[t] == m y''[t], y[0] == 0, y'[0] == 0}, y[t], t]
```

$$\text{Out[12]= } \left\{\left\{y[t] \to \frac{e^{-\frac{k\,t}{m}}\,g\,m\left(m - e^{\frac{k\,t}{m}}\,m + e^{\frac{k\,t}{m}}\,k\,t\right)}{k^2}\right\}\right\}$$

■ Tehtäviä

53. Päättele veneestä pudotetun tiiliskiven tapauksessa järkevät reunaehdot, ratkaise yhtälö *Mathematica*lla, anna vakioille numeroarvot ja piirrä ratkaisusta kuva. Mitä huomaat? Miten tutkisit asiaa edelleen? Tutki myös mitä tapahtuu, jos muutat tiiliskiven massaa ilman että muutat vakiota k.

54. Laske tiiliskiviesimerkki numeerisesti. Vertaa tuloksien yhtäpitävyytta piirtämällä kummastakin kuvaajat päällekäin, tai vaikka laskemalla niiden arvot muutamassa pisteessä.

Raketin lento

Tarkastellaan rakettia, joka polttaa massaansa tuottaakseen työntövoimaa. Rakettiin vaikuttaa vain maan vetovoima $-m\,(t)\,g$, joka oletetaan vakioksi, ja raketin moottorin aiheuttama työntövoima F_{moot}, joka myös oletetaan vakioksi.

Liikeyhtälön muodostaa yhtälöpari

$$F_{\text{moot}} - m(t)\,g = m(t)\,\ddot{x}(t),$$
$$m(t) = m_0 - a\,t$$

niin kauan kuin poltettavaa massaa riittää. Kun polttoaine on kulutettu, muuttuu $m(t)$:n lauseke. Tutkitaan nyt kuitenkin rakettimatkan alkuosan $t < 10\,s$ ratkaisua seuraavalla ohjelmanpätkällä.

In[13]:=
```
Clear[m]
g = 9.81
m0 = 100
a = 5
f = 3000
m[t_] = m0 - a t
ratkaisu = NDSolve[{f - m[t] g == m[t] x''[t], x[0] == 0, x'[0] == 0},
  x[t], {t, 0, 10}]
```

Out[14]=
```
9.81
```

Out[15]=
```
100
```

Out[16]=
```
5
```

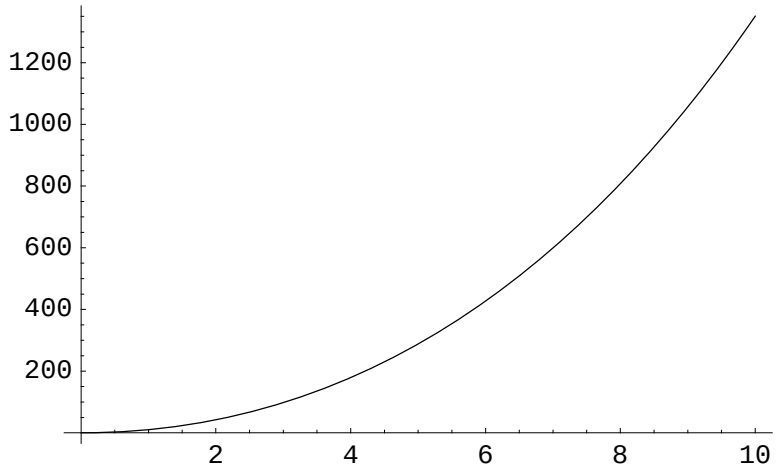
Out[17]=
```
3000
```

Out[18]=
```
100 - 5 t
```

Out[19]=
```
{{x[t] → InterpolatingFunction[{{0., 10.}}, <>][t]}}
```

Koska tulos ei ole kovin selväkielinen, tutkimme asiaa piirtämällä ratkaisusta, eli raketin korkeudesta ajan funktiona, kuvan.

```
In[20]:= Plot[x[t] /. ratkaisu, {t, 0, 10}]
```




```
Out[20]= - Graphics -
```

■ **Tehtäviä**

55. Laske rakettiesimerkki ja piirrä sen ratkaisu, mutta lisää ilmanvastuksen aiheuttama voima, $F_{\text{ilma}} = -k\,v^2 = -k\,\dot{x}(t)^2$. Käytä k:lle vaikka arvoa 0,3. Vieläkö ratkaistava yhtälö on lineaarinen? Vertaa raketin lentorataa esimerkin lentorataan.

Mekaniikkaa

Kitkaton värähtelijä

Tutkitaan kitkattoman jousen päähän kiinnitetyn kappaleen käyttäytymistä. Jousen jousivakio on k ja kappaleen massa m. Systeemi oletetaan kitkattomaksi. Kappaletta poikkeutetaan matkan s verran tasapainoasemastaan ja päästetään vapaaksi. Haluamme tietää, mikä on punnuksen poikkeama x tasapainoasemastaan kullakin ajanhetkellä. Kirjoitetaan liikeyhtälö:

$$F = m\,a = -k\,x$$

eli

$$m\,\ddot{x} = -k\,x.$$

Lisäksi tiedämme, että punnus oli hetkellä $t = 0$ poikkeutettuna matkan s tasapainoasemastaan, $x(0) = s$ ja sen nopeus oli nolla,

$$\dot{x}(0) = v(0) = 0.$$

Voimme käyttää näitä tietoja reunaehtoina.

Syötämme tietomme *Mathematica*an:

```
In[1]:=
Clear[m, x, k, s]
DSolve[{m x''[t] == -k x[t], x[0] == s, x'[0] == 0}, x[t], t]
```

```
Out[2]=
{{x[t] → s Cos[ √k t / √m ]}}
```

■ **Tehtäviä**

56. Laske esimerkin jouseen kiinnitetyn kappaleen hetkellinen nopeus.

57. Kirjoita tiedostoon *Mathematica*-ohjelma, joka antaa s:lle, k:lle ja m:lle järkevät arvot, ratkaisee yhtälön kuten edellä ja piirtää kuvaajan kappaleen poikkeamasta ajan funktiona.

58. Lisää ohjelmaan pätkä, joka laskee kappaleen hetkellisen nopeuden ja piirtää sen samaan kuvaan poikkeaman kanssa.Tulkitse kuvaa. (Huomaa, ettet voi viitata tuloksiin **%n**-menetelmällä ohjelman sisällä!)

Vaimennettu värähtely

(Katso Benson, University Physics s. 309, 15.5)

Todellisuudessa kuvan kaltaisessa systeemissä esiintyy erilaisia kitkavoimia. Kun jouseen kiinnitetyn kappaleen nopeus on pieni ja systeemi upotetaan nesteeseen, on kitkavoima verrannollinen nopeuteen,

$$F_{\text{kitka}} = -\gamma v.$$

Suuretta γ sanotaan vaimennusvakioksi. Kirjoitamme uudelleen liikeyhtälön ja otamme huomioon myös kitkan

$$m\ddot{x} + \gamma \dot{x} + k x = 0.$$

■ **Tehtäviä**

59. Ratkaise jousen paikka ajan funktiona.

60. Kuten kitkattomassa tapauksessa, piirrä paikan ja nopeuden aikakehitys samaan kuvaajaan. Mitä eroja havaitset?

Sähköoppia

LC-piirit

Tarkastellaan suljettua virtapiiriä, joka koostuu kondensaattorista, kapasitanssi C ja kelasta, induktanssi L. Kirchhoffin lain mukaan kierrettäessä virtapiiri, tulee jännitteiden (potentiaalierojen) summan olla nolla, $V_C + V_L = 0$. Sähköopista tiedetään että virta on sähkövarauksen aikaderivaatta, $I = \partial Q / \partial t$. Keloille pätee $V_L = L \, \partial I / \partial t$, eli $V_L = L \, \partial^2 Q / \partial t^2$. Kondensaattorin napojen välillä vallitsee jännite $V_C = Q / C$.

$$V_L = L \frac{d^2 Q}{dt^2}$$
$$V_C = \frac{Q}{C}$$
$$V_C + V_L = 0$$
$$\Longrightarrow \frac{Q}{C} + L \frac{d^2 Q}{dt^2} = 0$$

Yllättäen edessämme on toisen kertaluvun differentiaaliyhtälö. Ei kun ratkaisemaan! Mathematica sanoo

```
In[1]:=
Clear[c, l, q]
DSolve[q[t] / c + l q''[t] == 0, q[t], t]
```

$$\text{Out[2]= } \left\{\left\{q[t] \to C[1] \, \text{Cos}\left[\frac{t}{\sqrt{c}\,\sqrt{l}}\right] + C[2] \, \text{Sin}\left[\frac{t}{\sqrt{c}\,\sqrt{l}}\right]\right\}\right\}$$

Saimme harmonisen oskillaattorin lausekkeen.

■ Tehtäviä

61. Eliminoi vakiot asettamalla varaukselle Q ja virralle $\partial Q / \partial t$ sopivat alkuehdot, ja tutki saamaasi lopputulosta kuvin. Vaihtele komponenttien arvoja ja tutki systeemin käyttäytymistä.

LCR-piirit

LRC-piiri on huomattavasti lähempänä todellista tilannetta. Siinä on kytketty sarjaan kondensaattorin ja kelan kanssa myös vastus, joka aiheuttaa sähköenergian muuttumisen lämpöenergiaksi ja häviämisen systeemistä. Kirchhoffin säännön perusteella voimme kirjoittaa yhtälön

$$\frac{Q}{C} + I R + L \frac{dI}{dt} = 0$$

Käyttämällä LC-piirien yhteydessä tekemäämme havaintoa $I = \partial Q / \partial t$ voimme kirjoittaa saman muodossa

$$L \ddot{Q} + R \dot{Q} + \frac{Q}{C} = 0.$$

Verrattuna edelliseen tehtävään olemme saaneet yhden termin lisää. Mathematica osaa ratkaista tämän yhtälön analyyttisesti, mutta tulos on melko sekavan näköinen

```
In[3]:= DSolve[{l q''[t] + r q'[t] + q[t] / c == 0}, q[t], t]
```

$$\text{Out[3]=}\quad \left\{\left\{q[t] \to e^{\frac{\left(-c\,r-\sqrt{c}\,\sqrt{-4\,l+c\,r^2}\right)t}{2\,c\,l}}\,C[1] + e^{\frac{\left(-c\,r+\sqrt{c}\,\sqrt{-4\,l+c\,r^2}\right)t}{2\,c\,l}}\,C[2]\right\}\right\}$$

```
In[4]:= Simplify[%]
```

$$\text{Out[4]=}\quad \left\{\left\{q[t] \to e^{-\frac{\left(r+\frac{\sqrt{-4\,l+c\,r^2}}{\sqrt{c}}\right)t}{2\,l}}\,C[1] + e^{-\frac{\left(r-\frac{\sqrt{-4\,l+c\,r^2}}{\sqrt{c}}\right)t}{2\,l}}\,C[2]\right\}\right\}$$

■ **Tehtäviä**

62. Eliminoi vakiot kuten LC-piirien tapauksessa, ja tutki saamaasi lopputulosta kuvin. Vaihtele komponenttien arvoja ja tutki systeemin käyttäytymistä.

Sähköopin ja mekaniikan suureiden analogiaa

Mekaniikka	x	v	m	$\frac{1}{2}\,m\,v^2$	k	$\frac{1}{2}\,k\,x^2$	F
Sähköoppi	Q	I	L	$\frac{1}{2}\,L\,I^2$	$\frac{1}{C}$	$\frac{1}{2}\,\frac{Q^2}{C}$	V

Populaatioiden kilpailu

Epälineaarisia differentiaaliyhtälöitä

Edellä on tutkittu lineaarisia differentiaaliyhtälöitä, joilla on paljon sovellutuksia eri fysiikan aloilla. Seuraavassa tarkastellaan eräitä epälineaarisia differentiaaliyhtälötyyppejä, joilla voidaan kuvata toisiinsa kytkettyjen systeemien kehittymistä. Otamme tutkimuksen kohteeksi eläin- ja kasvimaailman eliöstöjen lisääntymisen ongelmat.

Perusyhtälö lukumäärän $N(t)$ kehittymiselle ajan t funktiona saadaan, jos tunnetaan lisääntymisnopeus ϵ,

$$\frac{dN(t)}{dt} = \epsilon\, N(t).$$

Yhtälö on muodoltaan sama kuin radioaktiivista hajoamista kuvaava yhtälö. Se on lineaarinen yhtälö ja tuloksena on eksponentiaalinen kasvu ilman rajoja. Biologisen systeemin käyttäytyminen tällä tavoin on mahdollista vain tiettyyn rajaan saakka, sillä kasvua rajoittavat tilan, ruuan, valon yms. puute, eli yksilöt tulevat tietoisiksi ympäristöstään ja lajitovereidensa olemassaolosta. Tämä näkyy lukumäärän kehittymistä kuvaavassa differentiaaliyhtälössä epälineaarisena terminä, jonka edessä kertoimena on erilaisia puutteita kuvaava vakio β.

$$\frac{dN(t)}{dt} = \epsilon\, N(t) - \beta\, N(t)^2.$$

In[1]:=
```
e = 4
b = 2
ra = DSolve[{n'[t] == e n[t] - b n[t]^2, n[0] == 1}, n[t], t]
```

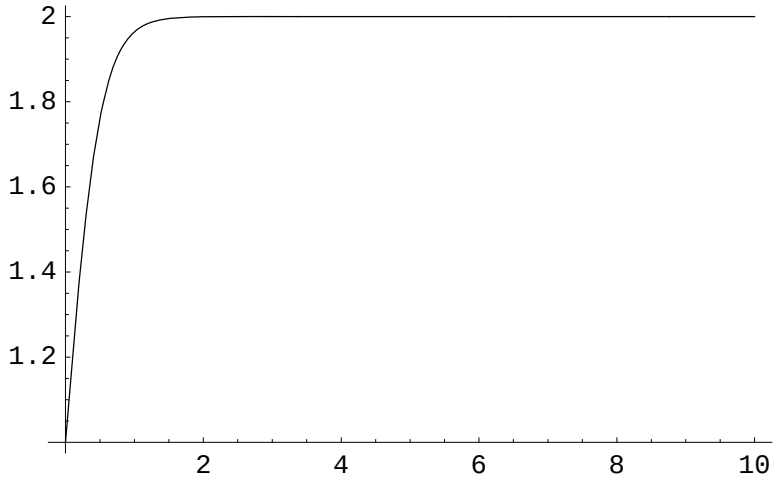
Out[1]=
```
4
```

Out[2]=
```
2
```

```
Solve::ifun :
 Inverse functions are being used by Solve, so some solutions may not
   be found; use Reduce for complete solution information. More…
```

Out[3]=
$$\left\{\left\{n[t] \to \frac{2\, e^{4\,t}}{1 + e^{4\,t}}\right\}\right\}$$

In[4]:=
```
Plot[n[t] /. ra, {t, 0, 10}, PlotRange -> All]
```



Out[4]=
```
- Graphics -
```

Toisenlainen epälineaarinen kytkentä aiheutuu toisten lajien olemassaolosta. Peto-saalis suhteessa saaliin määrä lisääntyy eksponentiaalisesti, ellei petoja ole olemassa. Toisaalta petojen lisääntyminen riippuu täysin saaliin lukumäärästä ja ne kuolevat pois eksponentiaalisesti, mikäli saalis hävitetään. Tällaista tilannetta kuvaamaan tarvitaan kaksi kytkettyä differentiaaliyhtälöä. Olkoon esimerkkinä kettujen ja jänisten lukumäärien kehittyminen,

$$\frac{dN_j(t)}{dt} = [\epsilon_j - \gamma_j\, N_k(t)]\, N_j(t)$$
$$\frac{dN_k(t)}{dt} = [-\epsilon_k - \gamma_k\, N_j(t)]\, N_k(t)$$

Yhtälöissä $N_j(t)$ kuvaa jänisten ja $N_k(t)$ kettujen lukumäärää. Vakio ϵ_j kertoo jänispopulaation kasvunopeuden ilman kettuja. Se riippuu ruuasta, ilmastosta jne. Vakio γ_j kuvaa sitä, kuinka paljon jäniksiä joutuu kettujen suihin, ja γ_k puolestaan sitä, miten kettujen määrä lisääntyy tämän saalistuksen seurauksena. Ketut eivät elä ilman jäniksiä, joten ϵ_k antaa nopeuden niiden sukupuuttoon kuolemiselle. Kokeilemalla erilaisia parametrien arvoja yhtälön ratkaisemiselle voit todeta, että luonnon tasapaino on hyvin herkkä ja sen järkkyminen johtaa aivan uudenlaiseen ratkaisuun.

*Mathematica*ssa nämä yhtälöt saavat muodon,

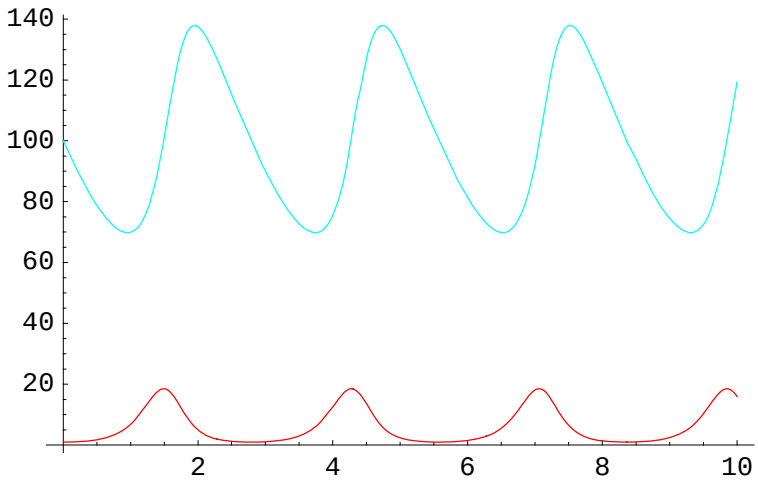
```
In[5]:= e1 = 10
        g1 = 0.1
        e2 = 0.6
        g2 = 0.1

Out[5]= 10

Out[6]= 0.1

Out[7]= 0.6

Out[8]= 0.1

In[9]:= rat = NDSolve[{j'[t] == (e1 - g1 k[t]) j[t], k'[t] == (-e2 + g2 j[t]) k[t],
            j[0] == 1, k[0] == 100}, {j[t], k[t]}, {t, 0, 10}]

Out[9]= {{j[t] → InterpolatingFunction[{{0., 10.}}, <>][t],
          k[t] → InterpolatingFunction[{{0., 10.}}, <>][t]}}
```

Kuva lukumäärän kehittymisestä ajan funktiona saadaan helposti piirtämällä **`Plot`**-funktiolla yhtälöiden ratkaisut. Toinen mielenkiintoinen tapa yhtälöryhmän ratkaisujen tutkimisessa on esittää ratkaisut toistensa funktiona, eli kuinka kettujen lukumäärä muuttuu jänisten lukumäärän funktiona. Tähän tarkoitukseen on käytettävissä **`ParametricPlot`**-funktio. Se on muotoa

```
ParametricPlot[{fx, fy}, {t, tmin, tmax}]
```

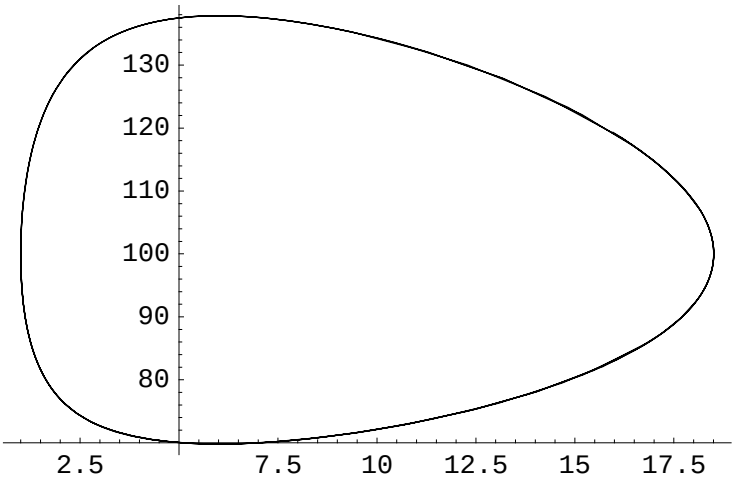
Tässä lausekkeet **`fx`** ja **`fy`** antavat käyrän *x* ja *y* komponentit ajan **`t`** funktiona.

```
In[10]:= Plot[{j[t] /. rat, k[t] /. rat}, {t, 0, 10}, PlotRange -> All,
          PlotStyle → {Hue[0], Hue[0.5]}]
```




```
Out[10]= - Graphics -
```

```
In[11]:= ParametricPlot[{j[t] /. rat[[1]], k[t] /. rat[[1]]}, {t, 0, 10},
          PlotRange -> All]
```




```
Out[11]= - Graphics -
```

Yllä esitetyt mallit ovat tietysti hyvin idealisoituja ja luonnossa monet tekijät yhdessä vaikuttavat lukumäärien kehittymiseen. Esimerkkinä voisi olla mäntyjen ja kuusien kilpailu elintilasta. Sitä voidaan kuvata differentialliyhtälöryhmällä, jossa molemmat edellä esitetyt epälineaariset termit on otettu mukaan.

$$\frac{dN_m}{dt} = [\epsilon_m - \beta_m N_m(t) - \gamma_m N_k(t)] N_m(t)$$
$$\frac{dN_k}{dt} = [\epsilon_k - \beta_k N_k(t) - \gamma_k N_m(t)] N_k(t)$$

Yhtälöissä $N_m(t)$ kuvaa mäntyjen ja $N_k(t)$ kuusien lukumäärää. Vakiot ϵ_m ja ϵ_k kertovat puiden lukumäärän lisääntymisnopeuden. Lukumäärän kasvaessa alkavat toiset puut varjostaa ja estää toisten kasvua, joten lukumäärä ei pääse kasvamaan rajatta. Vakiot β_m ja β_k kuvaavat saman lajin sekä γ_m ja γ_k eri lajin aiheuttamaa varjostusta.

In[12]:=
```
e1 = 1
g1 = 0.1
b1 = 0.1
e2 = 0.6
g2 = 0.1
b2 = 0.1
```

Out[12]=
```
1
```

Out[13]=
```
0.1
```

Out[14]=
```
0.1
```

Out[15]=
```
0.6
```

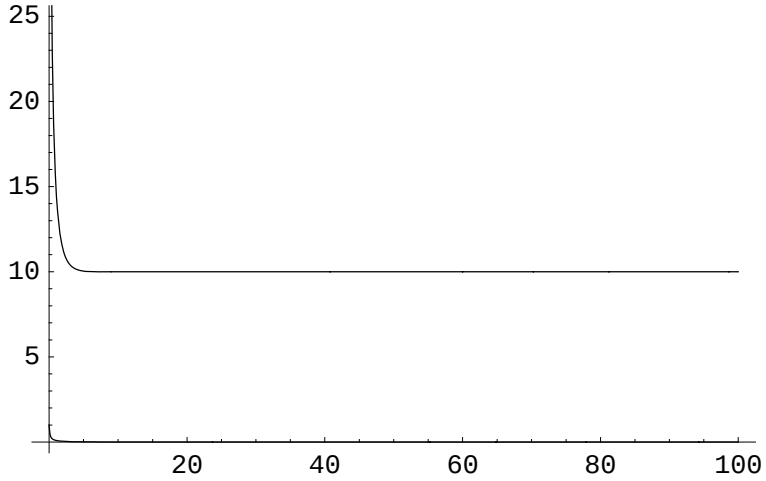
Out[16]=
```
0.1
```

Out[17]=
```
0.1
```

In[18]:=
```
dy1 := m'[t] == (e1 - b1 m[t] - g1 k[t]) m[t]
dy2 := k'[t] == (e2 - b2 k[t] - g2 m[t]) k[t]
rat = NDSolve[{dy1, dy2, m[0] == 100, k[0] == 1}, {m[t], k[t]}, {t, 0, 100}]
```

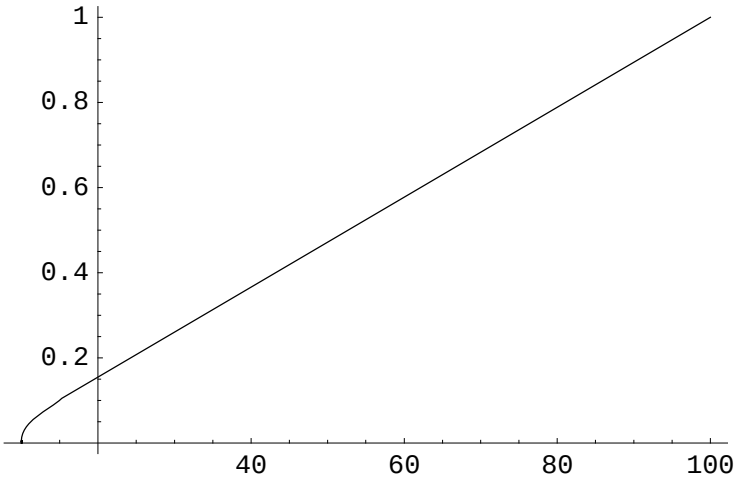
Out[20]=
```
{{m[t] → InterpolatingFunction[{{0., 100.}}, <>][t],
  k[t] → InterpolatingFunction[{{0., 100.}}, <>][t]}}
```

In[21]:=
```
Plot[{m[t] /. rat, k[t] /. rat}, {t, 0, 100}]
```



Out[21]= - Graphics -

In[22]:=
```
ParametricPlot[{m[t] /. rat[[1]], k[t] /. rat[[1]]}, {t, 0, 100},
 PlotRange -> All]
```



Out[22]= - Graphics -

■ Tehtäviä

63. Tutki yhtälön $dN(t)/dt = \epsilon N(t) - \beta N(t)^2$ käyttäytymistä eri parametrien arvoilla ja antamalla lukumäärälle eri lähtöarvoja ajan hetkellä $t = 0$. Piirrä lukumäärän kuvaaja ajan funktiona.

64. Tutki sekä jänisten ja kettujen että kuusien ja mäntyjen elämää kuvaavien yhtälöiden käyttäytymistä eri parametrien arvoilla ja eri lähtöarvoja käyttäen. Piirrä kettujen lukumäärä jänisten lukumäärän funktiona ja kuusien lukumäärä mäntyjen lukumäärän funktiona. Etsi kuvaajien avulla stabiilit ratkaisut, joissa lajien lukumäärät pysyvät ajan funktiona vakioina.

Ohjelmointi *Mathematica*lla

Ohjelma ja sen suoritus

*Mathematica*n kieli soveltuu myös ohjelmointiin. Se tuntee yleisimmät toistorakenteet ja ehtolauseet. *Mathematica*-ohjelmat kirjoitetaan tavalliseen tekstitiedostoon, jonka tunnisteeksi on sovittu .m. Esimerkiksi `ohjelma.m` on hyvä tiedostonimi.

Mathematican kieli on tulkittu kieli, toisin kuin esimerkiksi C-kieli Atk I-kurssilla. Tämä tarkoittaa sitä, että Mathematica lukee ohjelmaa ja käy sitä suoraan läpi käsky käskyltä, kääntämättä sitä ensin mihinkään muuhun - kuten esimerkiksi assembly - muotoon.

Kun ohjelma on kirjoitettu tiedostoonsa, käsketään Mathematicaa suorittamaan se. Mathematica lukee ohjelman käsky käskyltä ja yrittää suorittaa sen. Kaikkien grafiikkakäskyjen tulosteet avautuvat myös grafiikkanäytölle.

Ennen ohjelman ajoa on syytä varmistaa, että **`$Path`**-määrittelyissä on mukana hakemisto, jossa ohjelmatiedosto on. Hakemisto liitetään määrittelyihin komennolla **`SetDirectory["hakemisto"]`**.

Ohjelma luetaan Mathematicaan **<<** -käskyllä, esimerkiksi

In[1]:=
```
Clear[a, b, c, f]
<< ohjelma.m
```

Out[2]=
```
{{x → 0.666667 - 1.24722 i}, {x → 0.666667 + 1.24722 i}}
```

Ohjelma oli

In[3]:=
```
Clear[f, x]
a = 1.5
b = -2
c = 3
f[x_] := a x^2 + b x + c
Solve[f[x] == 0, x]
```

Out[4]=
```
1.5
```

Out[5]=
```
-2
```

Out[6]=
```
3
```

Out[8]=
```
{{x → 0.666667 - 1.24722 i}, {x → 0.666667 + 1.24722 i}}
```

■ Tehtäviä

65. Tutki esimerkin ohjelmaa `ohjelma.m`. Käy se läpi käsky käskyltä, ja selitä, mitä se tekee.

66. Kirjoita itse tiedostoon ohjelma, jossa määrittelet funktion $f(x) = x^3 - 4x + 12$ ja piirrät sen kuvaajan.

Ehtolauseet

Valinta eri käskyjen suorittamisen välillä tehdään **`If`**-komennolla

```
If[ehto, käskyt1, käskyt2]
```

Siinä annetaan ehto, jonka perusteella valinta suoritetaan. Ehto voi saada kaksi arvoa *tosi* ja *epätosi*, *Mathematica*ssa **`True`** tai **`False`**. Ehdon jälkeen seuraa kaksi käskysarjaa pilkuilla erotettuna. Sarja **`käskyt1`** suoritetaan, jos ehto on tosi ja **`käskyt2`** jos ehto on epätosi. On myös mahdollista antaa kolme lauseketta, joista viimeinen suoritetaan, jos ehto ei ole sen paremmin tosi kuin epätosikaan.

```
In[9]:= If[1 < 2, Print["Tosi"], Print["Vaarin"]]
```

```
Tosi
```

Ehtorakenne, joka usein tulee vastaan matemaattisia funktioita käsiteltäessä, liittyy funktion määrittelyalueen rajaamiseen. Funktioiden ei tarvitse olla edes jatkuvia, vaan ne voidaan määritellä paloittain.

```
funktio[x_]:= lauseke1/;määrittelyalue1
funktio[x_]:= lauseke2/;määrittelyalue2
```

Merkintää **`/;`** käytetään siis määrittelyalueen rajaamiseen.

■ Vertailuoperaattorit

Ehtojen muodostamisessa tarvitaan vertailuopraattoreita.

`x==y`	yhtä suuri
`x!=y`	eri suuri
`x>y`	suurempi kuin
`x<y`	pienempi kuin
`x>=y`	suurempi tai yhtä suuri kuin
`x<=y`	pienempi tai yhtä suuri kuin
`x===y`	identtisesti yhtä suuri

```
In[10]:= 1 < 2 < 3
```

```
Out[10]= True
```

```
In[11]:=
1 < 3 < 2
```

```
Out[11]=
False
```

■ **Loogiset operaattorit**

Loogisia operaatioita tarvitaan yhdistettäessä erilaisia ehtolausekkeita.

Not[x] tai **!**	ei
x&&y	ja
x||y	tai
Xor[x,y,…]	joko ... tai

```
In[12]:=
Not[True]
```

```
Out[12]=
False
```

Raketin lentorata

Tarkastellaan raketin lentorataa sellaisessa tapauksessa, että raketin polttama polttoaine on merkittvä osa kokonaismassasta, joten lennon aikana raketin massan muutos täytyy huomioida.

Massa ajan funktiona **m[t_]:=If[c t<p, m0-c t, m0-p]**.

Toinen tapa määritellä massa ajan funktiona:

```
In[13]:=
Clear[m]
m[t_] := m0 - c t /; c t < p
m[t_] := m0 - p /; c t >= p
```

Nousun differentiaaliyhtälö

```
In[16]:=
dy := m[t] x''[t] == fm[t] - m[t] g
```

Työntövoima ajan funktiona

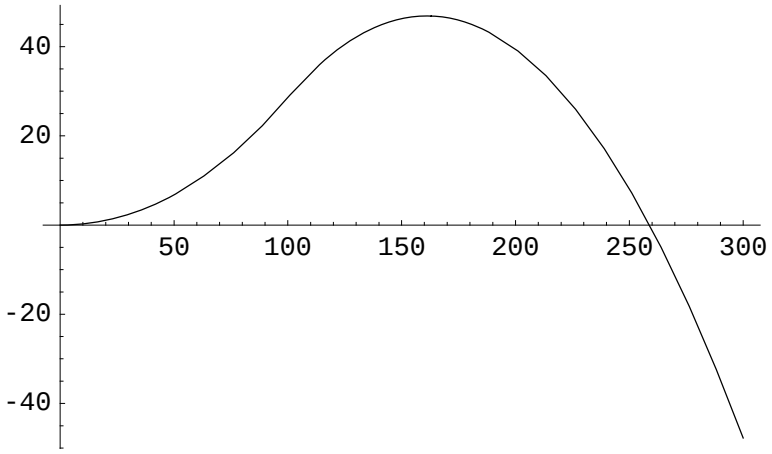
```
In[17]:=
fm[t_] := f /; c t < p
fm[t_] := 0 /; c t >= p
```

Vakiot yksiköissä kilogramma, kilometri ja sekunti $g = 9.81\,m/s^2$, $c = 1\,\mathrm{kg}/s$, $m_0 = 1000\,\mathrm{kg}$, $p = 100\,\mathrm{kg}$, $f = 15000\,N$

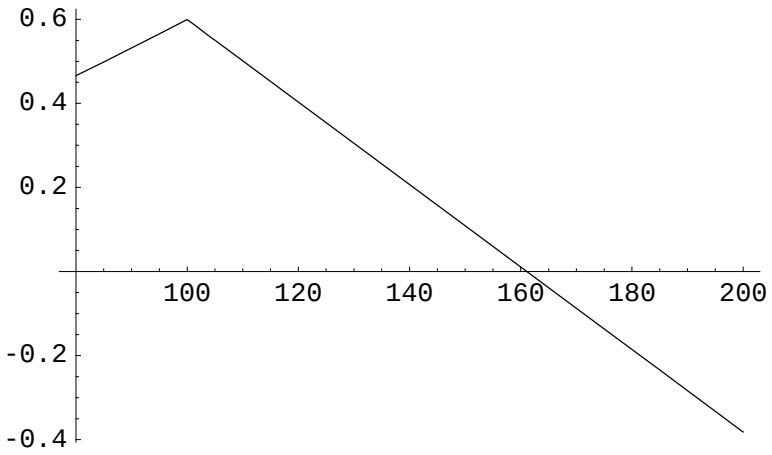
In[19]:=
```
g = 9.81 * 10^-3;
c = 1;
m0 = 1000;
p = 100;
f = 15;
```

Differentiaaliyhtälön ratkaisuna saadaan paikka x ajan t funktiona

In[24]:=
```
rat1 = NDSolve[{dy, x[0] == 0, x'[0] == 0}, x[t], {t, 0, 1000}, AccuracyGoal → 15];
paikka = x[t] /. rat1[[1]];
nopeus = D[paikka, t];
Plot[paikka, {t, 0, 300}]
Plot[nopeus, {t, 80, 200}]
```



Out[27]=
```
- Graphics -
```



Out[28]=
```
- Graphics -
```

Lasketaan hetki ja nopeus, kun korkeus on 40km

In[29]:=
```
tloppu = FindRoot[paikka == 40, {t, 180, 200}]
tl = t /. tloppu
loppunop = nopeus /. t → tl
```

Out[29]=
```
{t → 198.596}
```

Out[30]=
```
198.596
```

Out[31]=
```
-0.367816
```

ja radan maksimi

In[32]:=
```
tmax = FindRoot[nopeus == 0, {t, 150, 200}]
tm = t /. tmax
xmax = paikka /. t → tm
```

Out[32]=
```
{t → 161.102}
```

Out[33]=
```
161.102
```

Out[34]=
```
46.8955
```

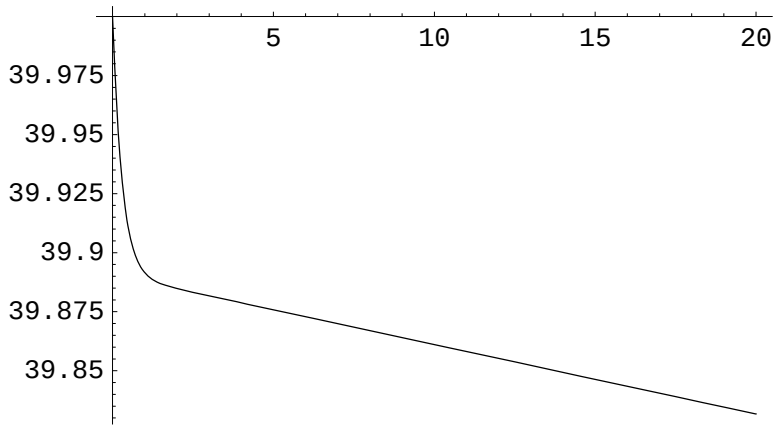
Polttoaineen loputtua raketti avaa laskuvarjon 40 km korkeudessa. Laskeutumista kuvataan differrentiaaliyhtälöllä, jossa huomioidaan myös ilmanvastus, ilmanvastuskertomeksi valitaan γ=3000 kg/s

In[35]:=
```
difflasku := (m0 - p) y''[t] + kitka y'[t] == -(m0 - p) g
kitka = 3000
```

Out[36]=
```
3000
```

Laskeutumisen ratkaisu

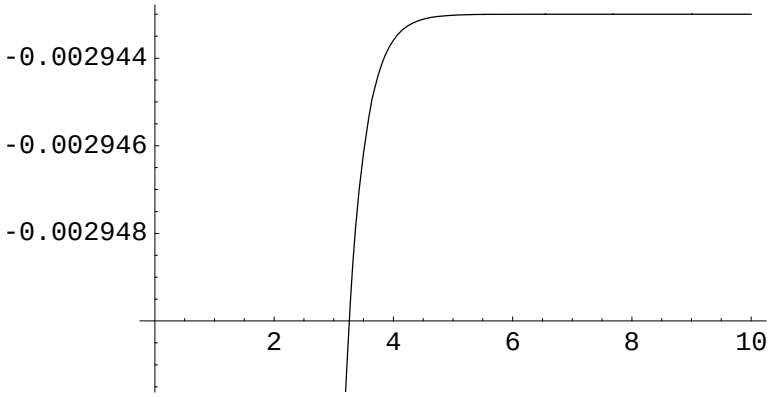
In[37]:=
```
rat2 = NDSolve[{difflasku, y[0] == 40, y'[0] == loppunop}, y[t], {t, 0, 20000}];
paikka2 = y[t] /. rat2[[1]];
Plot[paikka2, {t, 0, 20}, PlotRange → All]
```



Out[39]=
```
- Graphics -
```

Nopeus maahan tultaessa

In[40]:=
```
nopeus2 = D[paikka2, t];
Plot[nopeus2, {t, 0, 10}];
tmaa = FindRoot[paikka2 == 0, {t, 10000, 15000}];
tm2 = t /. tmaa
nopeusmaa = nopeus2 /. t → tm2
kmt = 3600 nopeusmaa km / t
```



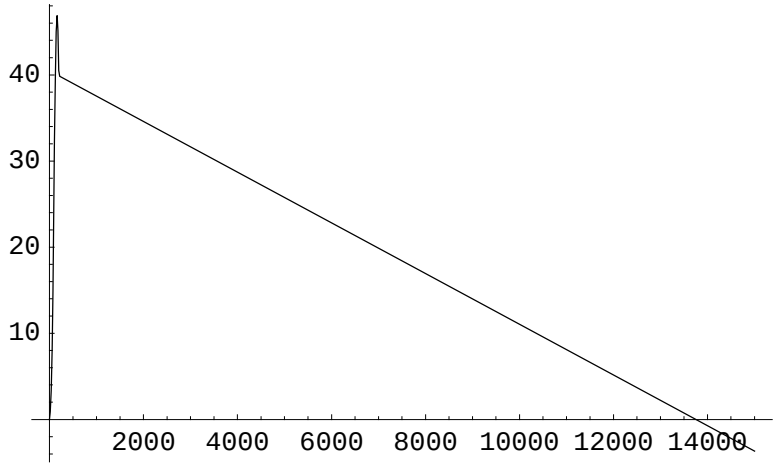
Out[43]=
```
13554.4
```

Out[44]=
```
-0.002943
```

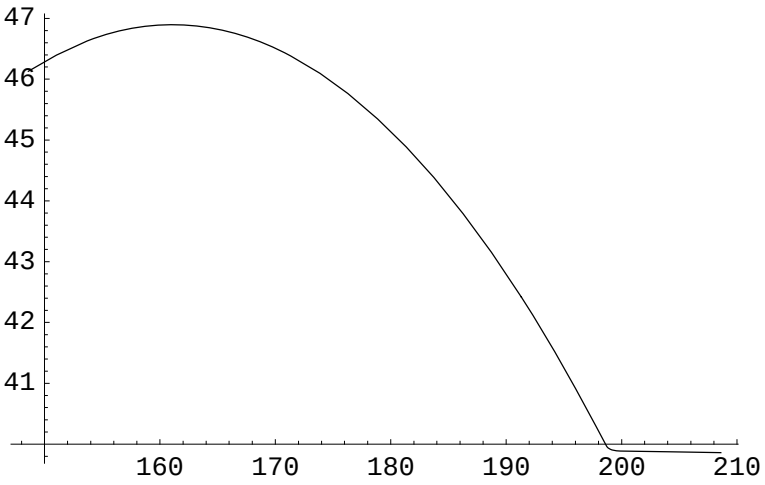
Out[45]=
$$-\frac{10.5948\ \text{km}}{t}$$

Piirretään koko lentorata

In[46]:=
```
lentorata[tt_] := If[tt ≤ tl, paikka /. t → tt, paikka2 /. t → tt - tl]
Plot[lentorata[tt], {tt, 0, 15000}]
Plot[lentorata[tt], {tt, tl - 50, tl + 10}, PlotRange → All]
```



Out[47]= - Graphics -



Out[48]= - Graphics -

■ Oikea maan vetovoima

Todellisessa tilanteessa maan vetovoima ei ole vakio vaan kääntäen verrannollinen etäisyyden neliöön

In[49]:=
```
<< Miscellaneous`PhysicalConstants`
rmaa = EarthRadius / Meter * 10^-3;
Print[rmaa, " km"]
```

$\frac{318907}{50}$ km

Korkeudella $z(t)$ gravitaatiovoiman aiheuttama kiihtyvyys $a_{\text{grav}} = g\left(\frac{R_{\text{maa}}}{z(t)+R_{\text{maa}}}\right)^2$. Se esitetään Mathematicassa seuraavasti:

In[52]:=
```
grav := g (rmaa / (z[t] + rmaa))^2
```

Raketin differentiaaliyhtälö $m\,a = F_{\text{moot}} - m(t)\,a_{\text{grav}}$ voidaan nyt kirjoittaa seuraavasti:

In[53]:=
```
diff3 := m[t] z''[t] == fm[t] - m[t] grav
```

■ Ratkaisun ja rajanopeuden etsiminen oikealle gravitaatiovoimalle

Jos raketin kineettinen energia, $\frac{1}{2}\,m\,v_p{}^2$, silloin, kun polttoaine loppuu on suurempi kuin sen potentiaalienergia, $m\,g\,R^2/(R+z)$, niin se vapautuu maan vetovoimasta. Rajanopeudeksi saadaan $v_{\text{raja}} = \sqrt{2\,g}\;R\,/\sqrt{R+z}$.

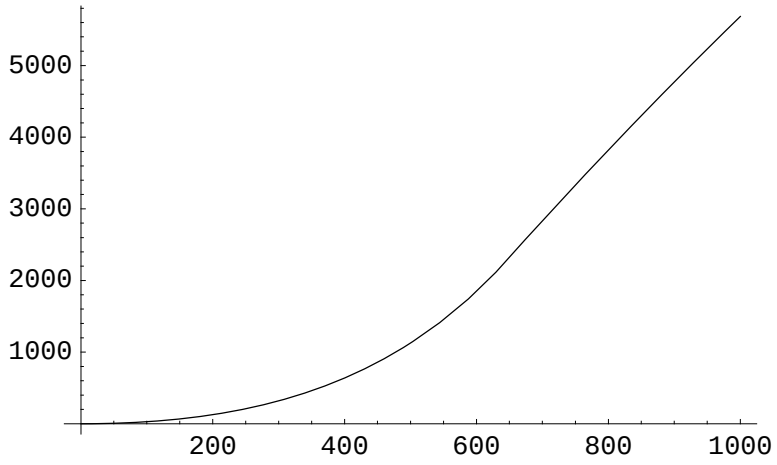
Polttoaineen loppumisen jälkeen raketin liikeitä kuvaa yhtälö

$$m\,a = F_{\text{grav}} \;\Rightarrow\; \ddot{z}(t) = a_{\text{grav}},$$

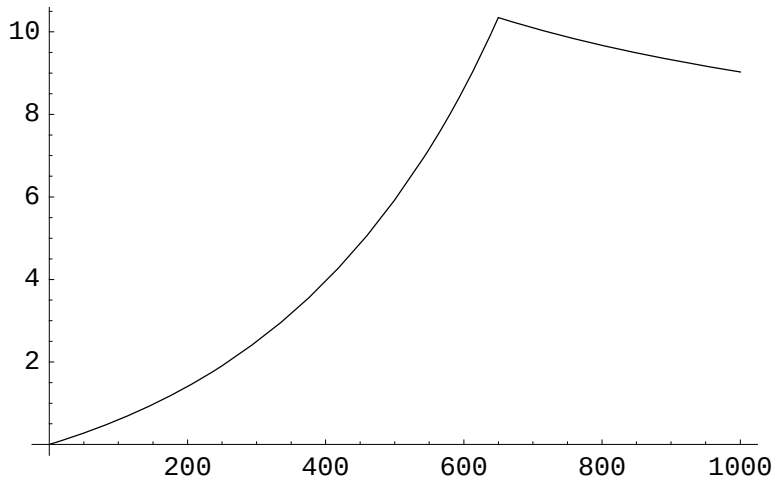
jonka ratkaisuna saadaan raketin nopeus $v^2 = \frac{2\,g\,R^2}{R+z} + v_\infty{}^2$. Hyvin kaukana maasta raketti matkaa siis nopeudella v_∞. Jotta rajanopeus saavutettaisiin, täytyy työntövoimaa kasvattaa edellisestä huomattavasti ja lisätä polttoainetta rakettiin.

In[54]:=
```
f = 15;
p = 650;
tpo = p / c;
max = 1000;
rat3 = NDSolve[{diff3, z[0] == 0, z'[0] == 0}, z[t], {t, 0, max},
   AccuracyGoal → 17];
paik[t_] = z[t] /. rat3[[1]];
vraja = rmaa * Sqrt[2 * g / (rmaa + paik[tpo])];
Print["rajanopeus = ", vraja, " km/s korkeudella ", paik[tpo], " km"]
nop[t_] = D[paik[t], t];
Print["v_p = ", nop[tpo], " km/s"];
vaareton = Sqrt[nop[tpo]^2 - vraja^2];
Print["nopeus kaukana on ", vaareton, " km/s"]
Plot[paik[t], {t, 0, max}]
Plot[nop[t], {t, 0, max}]
```

```
rajanopeus = 9.57858 km/s korkeudella 2321.17 km
v_p = 10.3486 km/s
nopeus kaukana on 3.91712 km/s
```



Out[66]= - Graphics -



Out[67]= - Graphics -

■ Tehtäviä

67. Tutustu rakettiesimerkkiin ja kokeile mitä tapahtuu, jos muutat vakioiden, kuten alussa olevan polttoaineen arvoja.

68. Määrittele funktio **f[x]**, joka saa arvon 1, kun $x \leq 0$ ja arvon 0, kun $x > 0$. Piirrä funktion kuvaaja.

■ Switch-lause

*Mathematica*sta löytyy C-kielen switch-rakenteen tapainen valintarakenne.

```
Switch[expr, form1, value1, form2, value2, …, _, other value]
```

Switch-lause laskee lausekkeen **expr** arvon ja vertaa tulosta jokaiseen **formi** arvoon. Jos **expr** == **formi** niin **Switch**-komennon tuloksena saadaan **valuei**. Kun mikään arvo ei käy, niin jätetään **Switch**-lause suorittamatta, tai jos alaviiva vaihtoehto on lisätty, niin suoritetaan alaviivan jälkeen tulevat käskyt.

Toistorakenteet

Mathematica tuntee **For**, **Do** ja **While** -komennot.

■ For-silmukka

For-komento on muotoa

For[alkuasetukset, ehto, laskuri, komennot],

(alkuasetukset: puolipisteillä erotettuna! komennot: puolipisteillä erotettuna!)

missä siis alkuasetukset ja komennot ovat esitettu puolipisteillä (**;**) erotettuina, jos niitä on useampi kuin yksi.

Alkuasetukset suoritetaan vain kerran, ennen muita toimia. Ehto tarkistetaan joka kierroksella, ja jos se on epätosi, poistutaan silmukasta, muuten jatketaan silmukan suorittamista järjestyksessä:

ehto

komennot

laskuri,

ehto

jne.

Laskuri ja komento ovat tavallisia Mathematica-komentoja. Jos halutaan useampi komento, voidaan ne erottaa puolipisteellä tai kirjoittaa listaksi.

Mathematica tuntee C-kielen tyyliset lyhennysmerkinnät:

Lyhennys	Merkitys
a += b	**a = a+b**
a -= b	**a = a-b**
a /= b	**a = a/b**
a *= b	**a = a*b**
++a, **a++**	**a = a+1**
--a, **a--**	**a = a-1**

Esimerkki:

```
In[68]:= For[a = 0; i = 0, i < N[Pi], i += N[Pi / 5], a += Sin[i]; Print[a]]
0
0.587785
1.53884
2.4899
3.07768
```

■ Do -silmukka

Do-komento on muotoa

```
Do[komennot, {laskuri, alkuarvo, loppuarvo, askel}]
```

(komennot: puolipisteillä erotettuna!)

Jos **askel** puuttuu käytetään oletusarvoa **askel=1**. Jos sekä askel että alkuarvo puuttuvat, niin käytetään molemmille oletusarvoa 1.

Esimerkki:

In[69]:=

```
a = 0
Do[a += Sin[i]; Print[a], {i, 0, N[Pi], N[Pi / 5]}]
```

Out[69]=

```
0
```

```
0
0.587785
1.53884
2.4899
3.07768
3.07768
```

■ While-silmukka

While-komento on muotoa

```
While[ehto, komennot]
```

(komennot: puolipisteillä erotettuna!)

Komentojen joukossa täytyy olla lauseke, joka muuttaa ehdon totuusarvoa silmukkaa suoritettaessa.

Esimerkki:

```
In[71]:= a = 0
         i = 0
         While[i < N[Pi], i += N[Pi / 5]; a += Sin[i]; Print[a]]
```

```
Out[71]= 0
```

```
Out[72]= 0
```

```
0.587785
1.53884
2.4899
3.07768
3.07768
```

■ Nest ja FixedPoint

Lisäksi Mathematica osaa **`Nest`** ja **`FixedPoint`**-toistorakenteet, joihin ei tässä sen tarkemmin puututa.

■ Tehtäviä

69. Kuten melkein sanasta voi päätellä, eivät edellä esitetyt esimerkit tee aivan täsmälleen samaa asiaa. Mitä ne tekevät eri tavalla?

70. Tutki, mitä eroa on lausekkeilla **`a++`** ja **`++a`**?

Vuorovaikutus ohjelman ajon aikana

Usein on tarve pyytää käyttäjää syöttämään esimerkiksi jokin luku. Tämän voi tehdä **`Input`**-komennolla.

Esimerkiksi

```
In[74]:= 2 + Input["Anna luku! "]
```

```
Out[74]= 5
```

Kuten aikaisemmin oli jo esillä, Mathematicassa voidaan datoja lukea myös tiedostosta **`ReadList`**-komennolla.

Komennolla

```
datat = ReadList["file", {Number,Number}]
```

luetaan tiedostosta **`file`** kahteen sarakkeeseen sijoitettuja lukuja taulukkoon **`datat`**.

Tiedostoon kirjoittamista varten täytyy avata kirjoituskanava käskyllä

```
In[75]:= kanava = OpenWrite["tiedosto", FormatType → OutputForm]
Out[75]= OutputStream[tiedosto, 7]
```

Tiedostoon kirjoitetaan kanavan kautta.

```
In[76]:= x = 2
         y = 4
         Write[kanava, x, "  ", y]
Out[76]= 2
Out[77]= 4
```

Lopuksi kirjoituskanava suljetaan komennolla **Close**:

```
In[79]:= Close[kanava]
Out[79]= tiedosto
```

■ Tehtäviä

71. Kirjoita *Mathematica* ohjelma, joka laskee n ensimmäistä Fibonaccin lukua. Fibonaccin luvut saadaan rekursiokaavalla

$$x_i = x_{i-1} + x_{i-2},\ i \geq 2$$
$$x_0 = 0$$
$$x_1 = 1$$

eli jokainen luku on kahden edeltäjänsä summa. Lukumäärä n kysytään käyttäjältä. Lukujen tulostukseen voit käyttää Print-komentoa.

■ Kertaustehtäviä

11. Määrittele paloittain funktio, joka saa arvon x^2, kun $x < 3$, arvon $x + 5$ kun $3 \leq x < 7$ ja arvon 14, kun $x \geq 7$. Piirrä f:n kuvaaja. Mieti, miten saisit funktiosta jatkuvan.

12. Laske summa $\sum_{i=0}^{10} i^2$ käyttäen **Do**-ja **While**-silmukoita.

13. Laske rekursiokaavan

$$y_i = 3\,y_{i-1} + 4\,y_{i-2},\ i \geq 2$$
$$y_0 = 1$$
$$y_1 = 9$$

mukainen y_i:n arvo, kun $i = 7$.

Schrödingerin yhtälö

Atomien ja molekyylien ominaisuuksia voidaan kuvata lineaarisella toisen kertaluvun differentiaaliyhtälöllä, jossa on ominaisarvo. Yhtälöä kutsutaan *Schrödingerin yhtälöksi*. Se voidaan kirjoittaa seuraavaan muotoon

$$y''(r) - V(r)\,y(r) = E\,y(r).$$

Funktio $V(r)$ kuvaa kahden mikromaailman hiukkasen (atomi, elektroni, protoni, jne.) välistä vuorovaikutusta ja E on yhtälön ominaisarvo, jonka arvo määräytyy ratkaisulle $y(r)$ vaadituista rajaehdoista, kun $r \to \infty$ ja/tai $r \to 0$.

Esimerkiksi vetyatomin tapauksessa Schrödingerin yhtälön ratkaisuna saadaan aaltofunktio $y(r)$, jonka neliö kuvaa elektronin todennäköisyysjakautumaa protonin ympäristössä, sekä ominaisarvo E eli energia, jolla protoni sitoo elektronin itseensä.

Yhtälön ratkaiseminen tapahtuu siten, että ensin arvataan ominaisarvo ja sitä käyttäen ratkaistaan differentialiyhtälö (**`NDSolve`**-komento). Jos ratkaisu ei toteuta haluttuja rajaehtoja, niin suoritetaan parempi arvaus ominaisarvolle ja yritetään uudelleen.

Mathematica ohjelma vetyatomin perustilalle

Edellä kuvattua yksinkertaista periaatetta voidaan kehittää siten että numeerinen tarkkuus paranee ja ratkaisusta saadaan stabiili. Tällöin pisteissä $r \to 0$ ja $r \to \infty$ annettuja alkuarvoja käyttäen lasketut ratkaisut sovitetaan yhteen pisteessä $r = r_{\text{väli}}$ siten, että sekä $y(r)$ että sen derivaatta $y'(r)$ ovat jatkuvia tässä pisteessä. Koska yhtälö on lineaarinen, niin ominaisarvon määräämiseen riittää, että suhde $y'(r)/y(r)$ on jatkuva kohtauspisteessä $r = r_{\text{väli}}$. Lineaarisen differentiaaliyhtälön ratkaisunhan voi vapaasti kertoa vakiolla ja tuloksena saadaan yhtälölle uusi ratkaisu. Lopuksi ratkaisu normitetaan siten, että

$$\int_0^\infty y(r)^2\,dr = 1.$$

Ominaisarvon arvausta parannetaan binaarihaun avulla. Haku toimii silloin, kun on löydetty kaksi ominaisarvoa $E = E_1$ ja $E = E_2$, jotka antavat eri merkin erotukselle

$$\Delta = \frac{y_0'(r_{\text{väli}})}{y_0(r_{\text{väli}})} - \frac{y_\infty'(r_{\text{väli}})}{y_\infty(r_{\text{väli}})}.$$

Merkinnät $y_0(r_{\text{väli}})$ ja $y_\infty(r_{\text{väli}})$ tarkoittavat origosta ja äärettömyydestä lähteviä ratkaisuja.

Schrödingerin yhtälö

In[1]:=

```
Clear[e, y]
yhtalo := y''[r] - (Vcoul[r] - e) y[r] == 0
```

Coulombin potentiaali, $V_{\text{Coul}} = -\frac{2}{r} + \frac{L(L+1)}{r^2}$,

In[3]:=

```
Vcoul[r_] := -2 / r + L (L + 1) / r^2
```

Alkuarvoissa asetetaan

- laskentatarkkuutta kuvaava parametri, **`tark`**

- piste, josta diff. yhtälön ratkaiseminen alkaa lähellä origoa, **`ralku`**

- piste, johon diff. yhtälön ratkaiseminen päättyy, **`rvali`**

- piste, josta diff. yhtälön ratkaiseminen alkaa kohti origoa, **rloppu**

- ensimmäinen arvaus ominaisarvolle, **e = e1**, ja sitä vastaava Δ, **Δ(e1)=ero1.**

- toinen arvaus ominaisarvolle, **e=e2**, ja sitä vastaava **Δ(e2)=ero2.**

In[4]:=
```
L = 0;
tark = 10^(-12) ;
ralku = 10^(-8) ;
rvali = 1;
rloppu = 8;
e1 = -1.1;
e2 = -0.9;
```

Ratkaisufunktion asymptoottinen käyttäytyminen alkupisteessä $y(r \to 0) = C_1\, r$ ja loppupisteessä $y(r \to \infty) = C_2\, e^{-\sqrt{-E}\, r}$. Lasketaan myös funktion derivaatat alku- ja loppupisteissä.

In[11]:=
```
Clear[r, e, yal, dyal, ylo, dylo]
c1 = 1
c2 = 10^-4
yal[r_] = c1 * r
dyal[r_] = D[yal[r], r];
ylo[r_] = c2 * E^(-Sqrt[-e] * r)
dylo[r_] = D[ylo[r], r];
```

Out[12]=
```
1
```

Out[13]=
$$\frac{1}{10000}$$

Out[14]=
```
r
```

Out[16]=
$$\frac{e^{-\sqrt{-e}\, r}}{10000}$$

Lasketaan logaritmisten derivaattojen erotus, **ero1,** sovituspisteessä, **rvali,** sen jälkeen arvataan uusi ominaisarvo, **e2**, ja lasketaan sitä vastaava logaritmisten derivaattojen erotus, **ero2**. Mikäli **ero1** ja **ero2** ovat erimerkkiset, niin oikean ominaisarvon täytyy olla välillä **e1 < e < e2**. Parannetaan arvausta puolittamalla väli jokaisella kierroksella ja huolehtimalla siitä, että **ero1** ja **ero2** ovat aina erimerkkiset.

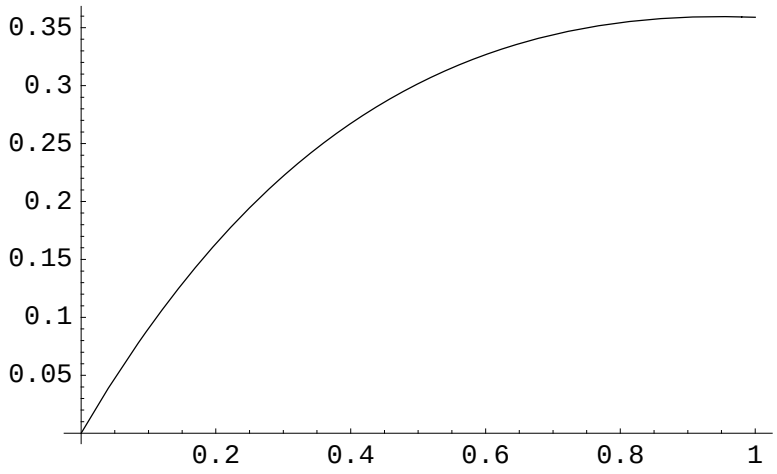
■ Alkuarvojen etsintä ominaisarvolle (e1 ja e2)

In[18]:=
```
Clear[y, r]
e = e2
rata = NDSolve[{yhtalo, y[ralku] == yal[ralku], y'[ralku] == dyal[ralku]},
       y[r], {r, ralku, rvali}, AccuracyGoal → 15]
```

Out[19]=
```
-0.9
```

Out[20]=
```
{{y[r] → InterpolatingFunction[{{1. × 10^-8, 1.}}, <>][r]}}
```

In[21]:=
```
kuvaalku = Plot[y[r] /. rata, {r, ralku, rvali}, PlotRange -> All]
```



Out[21]=
```
- Graphics -
```

In[22]:=
```
Clear[alk]
alk[r_] = y[r] /. rata[[1]]
logalk[r_] = D[alk[r], r] / alk[r]
```

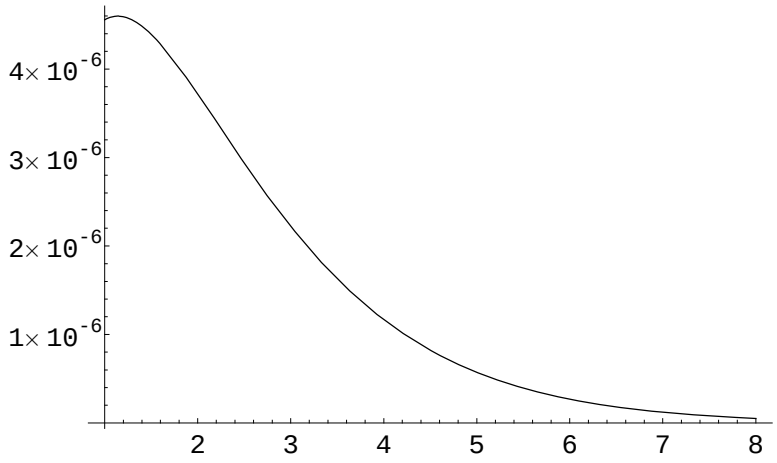
Out[23]=
```
InterpolatingFunction[{{1. × 10^-8, 1.}}, <>][r]
```

Out[24]=
$$\frac{\text{InterpolatingFunction}[\{\{1.\times 10^{-8}, 1.\}\}, <>][r]}{\text{InterpolatingFunction}[\{\{1.\times 10^{-8}, 1.\}\}, <>][r]}$$

In[25]:=
```
Clear[lop]
ratl = NDSolve[{yhtalo, y[rloppu] == ylo[rloppu], y'[rloppu] == dylo[rloppu]},
  y[r], {r, rloppu, rvali}, AccuracyGoal → 15]
```

Out[26]=
```
{{y[r] → InterpolatingFunction[{{1., 8.}}, <>][r]}}
```

In[27]:=
```
kuvaloppu = Plot[y[r] /. ratl, {r, rvali, rloppu}, PlotRange -> All]
```



Out[27]=
```
- Graphics -
```

In[28]:=
```
lop[r_] = y[r] /. ratl[[1]]
loglop[r_] = D[lop[r], r] / lop[r]
```

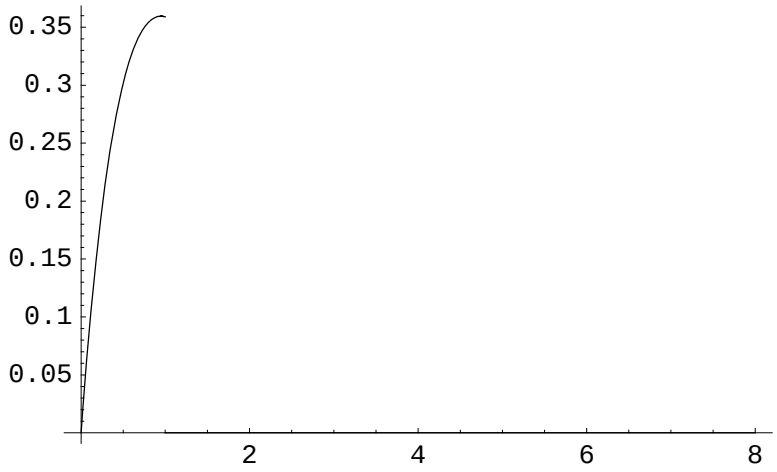
Out[28]=
```
InterpolatingFunction[{{1., 8.}}, <>][r]
```

Out[29]=
$$\frac{\text{InterpolatingFunction}[\{\{1., 8.\}\}, <>][r]}{\text{InterpolatingFunction}[\{\{1., 8.\}\}, <>][r]}$$

In[30]:=
```
ero = logalk[rvali] - loglop[rvali]
```

Out[30]=
```
-0.197543
```

```
In[31]:= Show[kuvaalku, kuvaloppu]
```




```
Out[31]= - Graphics -
```

Logaritmisten derivaattojen erotus arvoilla **e1** (**ero1**) ja **e2** (**ero2**)

```
In[32]:= ero1 = 0.174;
         ero2 = -0.197;
```

■ Varsinainen iterointi

In[34]:=
```
While[Abs[e1 - e2] > tark,
 e = (e1 + e2) / 2;

(* alkuosa *)
 rata = NDSolve[{yhtalo, y[ralku] == yal[ralku], y'[ralku] == dyal[ralku]},
   y[r], {r, ralku, rvali}, AccuracyGoal → 15];
 alku[r_] = y[r] /. rata[[1]];
 logalk[r_] = D[alku[r], r] / alku[r];

(* loppuosa *)
 ratl = NDSolve[{yhtalo, y[rloppu] == ylo[rloppu], y'[rloppu] == dylo[rloppu]},
   y[r], {r, rloppu, rvali}, AccuracyGoal → 15];
 loppu[r_] = y[r] /. ratl[[1]];
 loglop[r_] = D[loppu[r], r] / loppu[r];

 (* alku- ja loppuosan yhteensopivuus *)
 ero = logalk[rvali] - loglop[rvali];
 If[ero ero1 >= 0,
  e1 = e; ero1 = ero,
  e2 = e; ero2 = ero;];
] (* While-silmukka loppuu *)

Print["ominaisarvo = ", e]
Print["logaritmisten derivaattojen erotus = ", ero]
```

```
ominaisarvo = -0.999997
```

```
logaritmisten derivaattojen erotus = 1.44922×10^-12
```

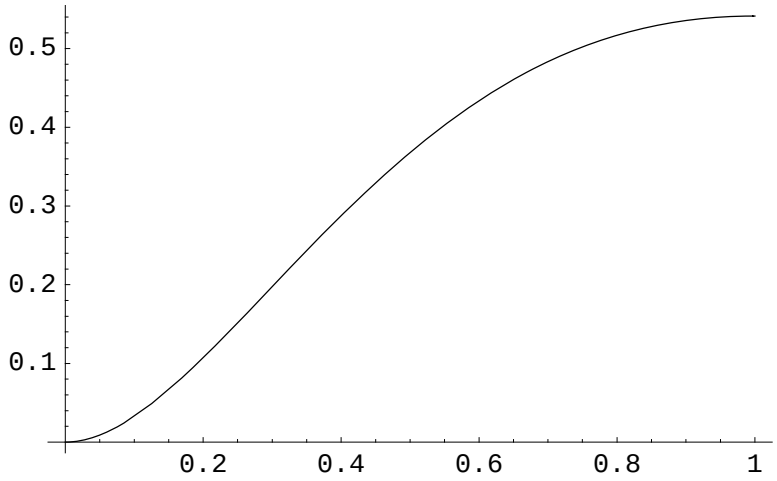
Sovitetaan ratkaisut yhteen ja normitetaan. Nämä ehdot määräävät vakiot **c1** ja **c2**.

In[37]:=
```
(* ratkaisujen sovittaminen yhteen *)
c2 = alku[rvali] / loppu[rvali];

(* ratkaisun normittaminen *)
norm = NIntegrate[(alku[r])^2, {r, ralku, rvali}] +
  c2^2 NIntegrate[(loppu[r])^2, {r, rvali, rloppu}]
```

```
General::spell1 : Possible spelling error: new
   symbol name "norm" is similar to existing symbol "Norm". More…
```

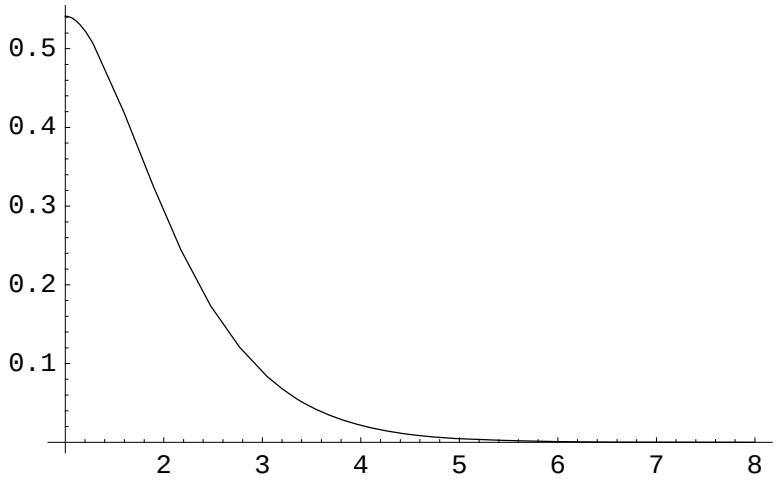
Out[38]=
```
0.249989
```

In[39]:=
```
(* kuvaajat: *)
(* Todennäköisyystiheydet *)
kuvaalku = Plot[(alku[r])^2 / norm, {r, ralku, rvali}, PlotRange -> All]
```



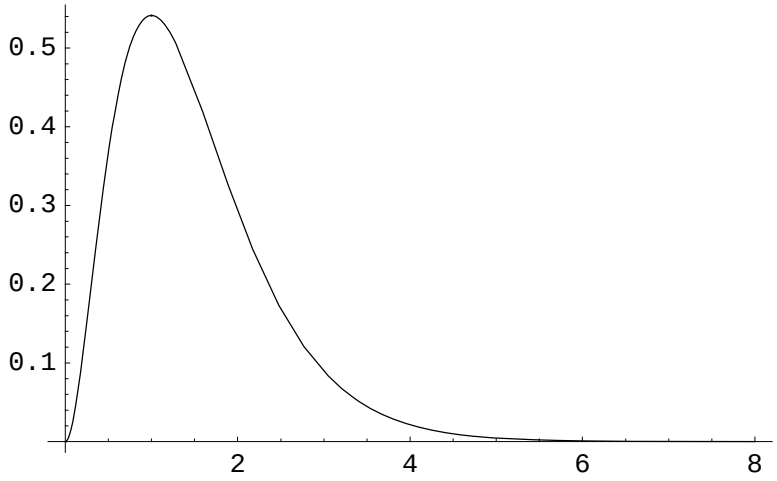
Out[39]= - Graphics -

In[40]:=
```
kuvaloppu = Plot[(c2 loppu[r])^2 / norm, {r, rvali, rloppu}, PlotRange -> All]
```



Out[40]= - Graphics -

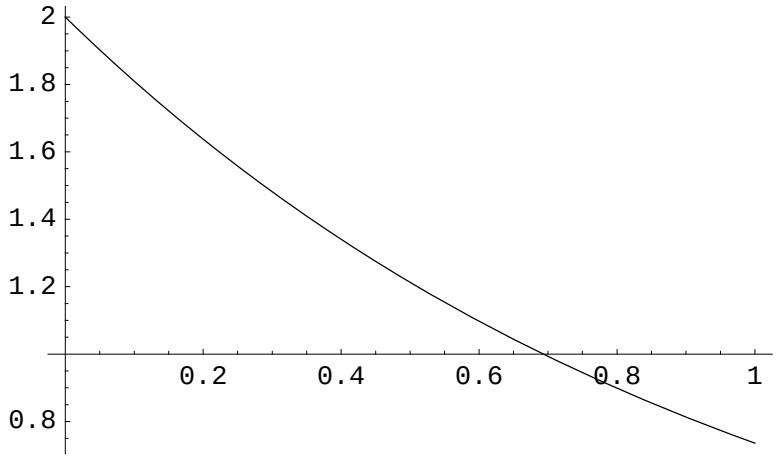
In[41]:=
```
Show[kuvaalku, kuvaloppu, PlotRange -> All]
```

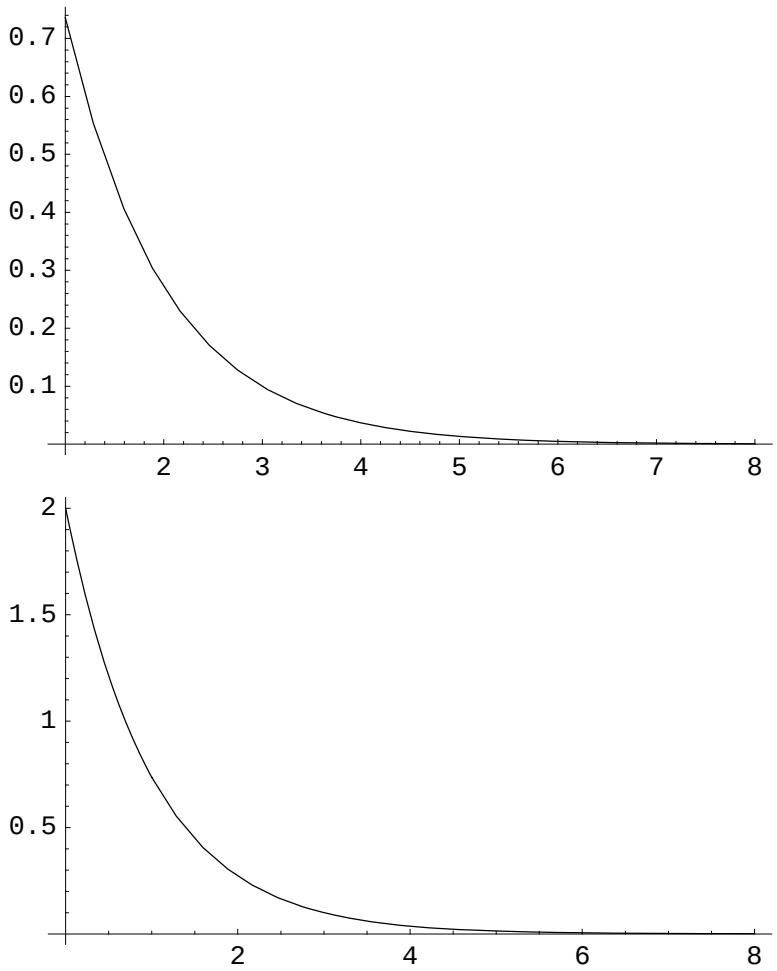


Out[41]=
```
- Graphics -
```

Lasketaan varsinaiset aaltofunktiot, $y(r)/r$.

In[42]:=
```
kuva3 = Plot[(alku[r] / r) / Sqrt[norm], {r, ralku, rvali}, PlotRange -> All];
kuva4 = Plot[(c2 loppu[r] / r) / Sqrt[norm], {r, rvali, rloppu}, PlotRange → All];
Show[kuva3, kuva4, PlotRange -> All]
```





Out[44]=

```
- Graphics -
```

■ Tarkka ratkaisu

Lasketaan myös Schrödingerin yhtälön tarkka ratkaisu Coulombin potentiaalille $(-2/r)$, kun tiedetään, että ominaisarvot ovat muotoa $E = 1/(k+L+1)^2$, missä ns. pääkvanttiluku k saa arvot 0, 1, 2 .... Tuloksena saadaan hypergeometrisia funktioita, jotka ovat Laguerren polynomeja.

In[45]:=

```
Clear[y, e, n, tarkka, L, k]
e = -1 / (1 + k + L) ^2;
rat = DSolve[yhtalo, y[r], r] // Simplify;
tarkka[k_, L_] = PowerExpand[rat] // Simplify
```

Out[48]=

$$\left\{\left\{y[r] \to e^{-\frac{r}{1+k+L}}\, r^{1+L} \left(C[1]\ \text{HypergeometricU}\left[-k,\ 2+2L,\ \frac{2r}{1+k+L}\right] + C[2]\ \text{LaguerreL}\left[k,\ 1+2L,\ \frac{2r}{1+k+L}\right]\right)\right\}\right\}$$

Valitaan kvanttiluvut k ja L, normitetaan ratkaisu ja piirretään kuvaaja.

In[49]:=
```
k = 0;
L = 0;
fun = y[r] /. (tarkka[k, L] /. C[2] → 0);
fu = fun[[1]]
```

Out[52]=
$$e^{-r}\, r\, C[1]$$

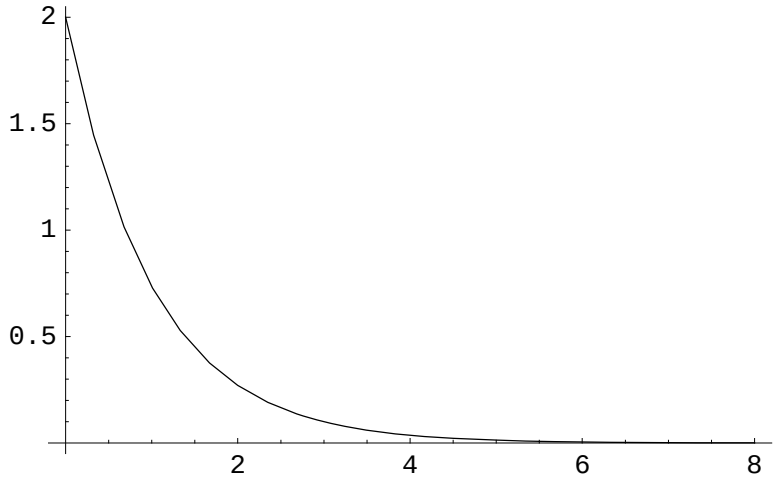
In[53]:=
```
normtark = Integrate[fu^2, {r, 0, Infinity}]
```

Out[53]=
$$\frac{C[1]^2}{4}$$

In[54]:=
```
aalto[r_] = fu / (Sqrt[normtark] * r) // PowerExpand
```

Out[54]=
$$2\, e^{-r}$$

In[55]:=
```
Plot[aalto[r], {r, 0, 8}, PlotRange → All]
```



Out[55]=
```
- Graphics -
```

■ Tehtäviä

72. Yritä löytää vetyatomin 1. viritystilan energia ja määrittää sitä vastaava aaltofunktio.

Vektoreita ja matriiseja

Peruskomentoja

Mathematicassa vektorit ovat listoja. Esimerkiksi vektori $a = (x, y, z)$ esitetään listana **a={{x,y,z}}**. Matriisit vuorostaan ovat listoja, joiden alkioina on listoja. Matriisi

$$M = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

annetaan Mathematicalle muodossa **m={{a,b},{c,d}}**.

Tärkeimmät vektoreiden ja matriisien laskutoimitukset:

Vektorin p kertominen skalaarilla a: **p a**

Skalaarivakion a lisääminen kaikkiin vektorin p komponentteihin: **a+p**

Piste- eli sisätulo: **a.b**

Matriisin **m** ja vektorin **a** tulo: **m.a**

Kahden matriisin tulo: **m1.m2**

Transponoitu matriisi: **Transpose[m]**

Käänteismatriisi: **Inverse[m]**

Determinantti: **Det[m]**

Ominaisarvot: **Eigenvalues[m]**

Ominaisvektorit: **Eigenvectors[m]**

Matriisiyhtälön $M\ x = a$, *Mathematica*ssa **m.x==a**, ratkaiseminen x:n suhteen: **LinearSolve[m,a]**

Mathematica osaa näyttää matriisit selkeämmässä muodossa **MatrixForm**-komennon avulla. Tämän komennon voi kirjoittaa myös varsinaisen laskukomennon perään. Esimerkin matriisi **m** on määritelty kuten tehtävissä.

```
In[1]:= m = {{1, 2}, {3, 4}}
```

```
Out[1]= {{1, 2}, {3, 4}}
```

```
In[2]:= Inverse[m] // MatrixForm
```

```
Out[2]//MatrixForm=
```

$$\begin{pmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{pmatrix}$$

■ **Tehtäviä**

74. Määrittele matriisi

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

ja laske sen käänteismatriisi, ominaisarvot, ominaisvektorit, transponoitu matriisi ja determinantti.

75. Esitä edellisen tehtävän tulokset matriisimuodossa.

76. Määrittele vektori $a = \begin{pmatrix} 4 \\ 5 \end{pmatrix}$ ja ratkaise x yhtälöstä $M\ x = a$.

77. Kokeile auttaako **MatrixForm** esittämään vektoreita selkeämmässä muodossa.

78. Määrittele jokin 3x3-matriisi ja tutki, onko sillä käänteismatriisia. Jos on,niin laske se ja totea tuloksen oikeellisuus kertomalla matriisi käänteismatriisillaan.

Schrödingerin yhtälön ratkaiseminen matriisimuodosta

Esimerkkinä ratkaistaan Schrodingerin yhtälö

$$y''(r) - V(r)\, y(r) = e\ y(r)$$

matriisimuodossa. Sitä varten kirjoitetaan toinen derivaatta ns. kolmen pisteen kaavan avulla,

$$y''(r) \to \frac{y(r-h) - 2\, y(r) + y(r+h)}{h^2}$$

Piste r on matriisin diagonalielementti ja pisteet $r - h$ ja $r + h$ vasemman ja oikeanpuoleisia sivudiagonaaleja.

Alkuarvoiksi tarvitaan

- pisteiden vali, **h**

- alkupiste differentiaaliyhtälön ratkaisemiselle, **alku**

- loppupiste differentiaaliyhtälön ratkaisemiselle, **loppu**

- pisteiden lukumäärä, **nn**

In[3]:=
```
h = 0.05;
alku = 0;
loppu = 8;
nn := (loppu - alku) / h + 1;
```

Potentialiksi valitaan tällä kertaa harmoonisen värähtelijän potentiaali, $V(r) = r^2$.

In[7]:=
```
Clear[x, r, v, i, j, d]
v = Table[r^2, {r, alku, loppu, h}];
```

Muodostetaan kerroinmatriisi toisen derivaatan ja potentiaalitermin laskemiseksi.

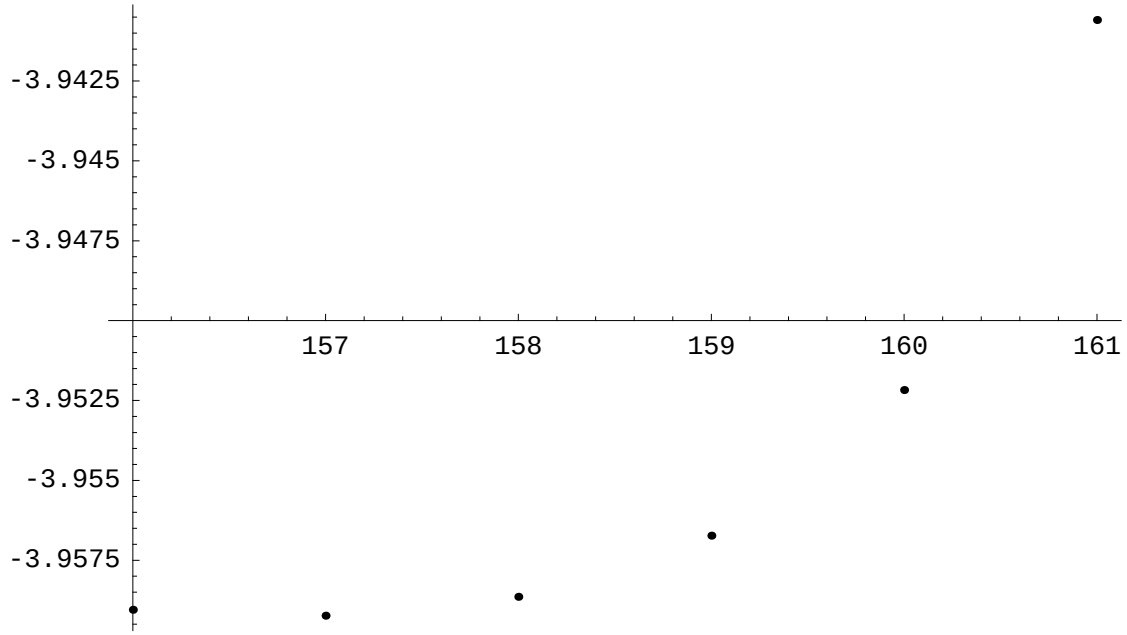
In[9]:=
```
d := -1 / h^2
mat = Table[Switch[i - j, -1, d, 0, v[[i]] - 2 d, 1, d, _, 0], {i, nn}, {j, nn}];
```

Lasketaan ominaisarvot ja ominaisvektorit. Ratkaisuna saadut ominaisarvot on järjestetty siten, että alin ominaisarvo on viimeinen ja ylin ensimmäinen. Vastaavat ominaisvektorit ovat samassa järjestyksessä.

In[11]:=
```
eigen = Eigenvalues[mat];
eigvec = Eigenvectors[mat];
```

Lasketaan peräkkäisten ominaisarvojen erotus. Jos tulokset ovat tarkkoja, niin erotuksien pitäisi olla yhtäsuuria.

In[13]:=
```
erotus[i_] := eigen[[i]] - eigen[[i - 1]]
ListPlot[Table[{i, erotus[i]}, {i, nn, nn - 5, -1}], PlotJoined → False]
```



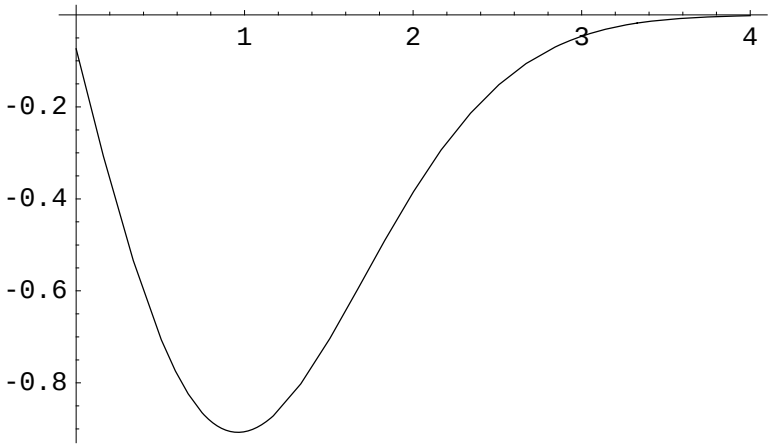
Out[14]=
```
- Graphics -
```

Tulostetaan muutama viimeinen ominaisvektori siten, että interpoloidaan polynomi laskettujen pisteiden kautta.

In[15]:=
```
normi =
 NIntegrate[
  Evaluate[Table[Interpolation[-eigvec[[nn - i]]][(x - alku) / h + 1], {i, 0, 0}]^
    2], {x, alku, loppu}]
```

Out[15]=
```
{0.0499954}
```

In[16]:=
```
kuva1 =
 Plot[
  Evaluate[Table[Interpolation[-eigvec[[nn - i]]][(x - alku) / h + 1 ] /
     (Sqrt[normi]), {i, 0, 0}]], {x, alku, 4}, PlotRange → All]
```



Out[16]=
```
- Graphics -
```

■ Tarkka ratkaisu

Schrödingerin yhtälön tarkka ratkaisu harmooniselle oskillaattorille, $V(r) = r^2$.

In[17]:=
```
Clear[n, L, harmosk, k]
e = 2 (2 * k + L) + 3
harmosk[k_, L_] = u''[r] + (e - r^2 - L * (L + 1) / r^2) u[r] == 0
rathar[k_, L_] = DSolve[harmosk[k, L], u[r], r]
```

Out[18]=
```
3 + 2 (2 k + L)
```

Out[19]=
```
(3 + 2 (2 k + L) - L (1 + L) / r^2 - r^2) u[r] + u''[r] == 0
```

Out[20]=
```
DSolve[(3 + 2 (2 k + L) - L (1 + L) / r^2 - r^2) u[r] + u''[r] == 0, u[r], r]
```

In[21]:=
```
Clear[aahar]
k = 0
L = 0
rat = rathar[k, L] /. C[2] → 0 // Simplify
aah = u[r] /. rat[[1]]
no = Integrate[aah^2, {r, 0, Infinity}]
aalhar[r_] = aah / (Sqrt[no]) // PowerExpand
kuva2 = Plot[aalhar[r], {r, 0, 4}, PlotRange → All]
```

Out[22]=
$$0$$

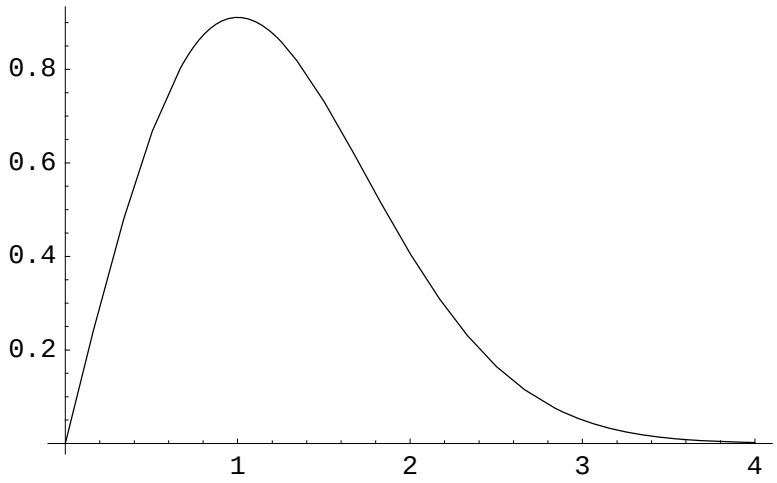
Out[23]=
$$0$$

Out[24]=
$$\left\{\left\{u[r] \to 2\, e^{-\frac{r^2}{2}}\, r\, C[1]\right\}\right\}$$

Out[25]=
$$2\, e^{-\frac{r^2}{2}}\, r\, C[1]$$

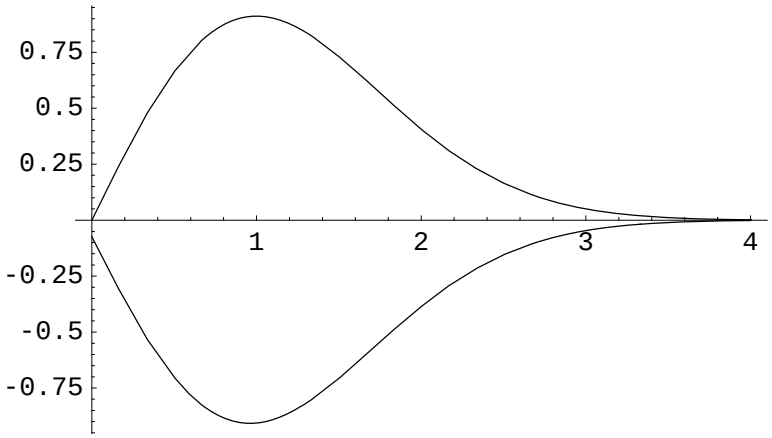
Out[26]=
$$\sqrt{\pi}\, C[1]^2$$

Out[27]=
$$\frac{2\, e^{-\frac{r^2}{2}}\, r}{\pi^{1/4}}$$



Out[28]=
```
- Graphics -
```

In[29]:=
```
Show[kuva1, kuva2]
```



Out[29]=
```
- Graphics -
```