Aalto Yliopisto

SCI-C0200 - Fysiikan ja matematiikan menetelmien studio

Tietokoneharjoitus 8: Monte Carlo -simulointi

Elli Kiiski

Sisällys

1 Tehtävä A: Log-normaalijakauma

Tarkastellaan erästä log-normaalijakaumaa satunnaismuuttujalla $Y = e^X$, missä X on normaalijakautunut. Ensin vertaillaan normaalijakauman ja log-normaalijakauman histogrammeja ja sen jälkeen otoskoon vaikutusta log-normaalijakauman perametreihin ja luottamusväliin.

1.1 Jakaumien histogrammit

Vertaillaan ensin normaalijakauman ja log-normaalijakauman histogrammeja. Generoidaan 500 standardinormaalijakautunutta satunnaislukua ja tehdään niille exponenttimuunnos. Plotataan 20 luokan histogrammi kummastakin jakaumasta. Homma hoituu seuraavasti MATLABilla:

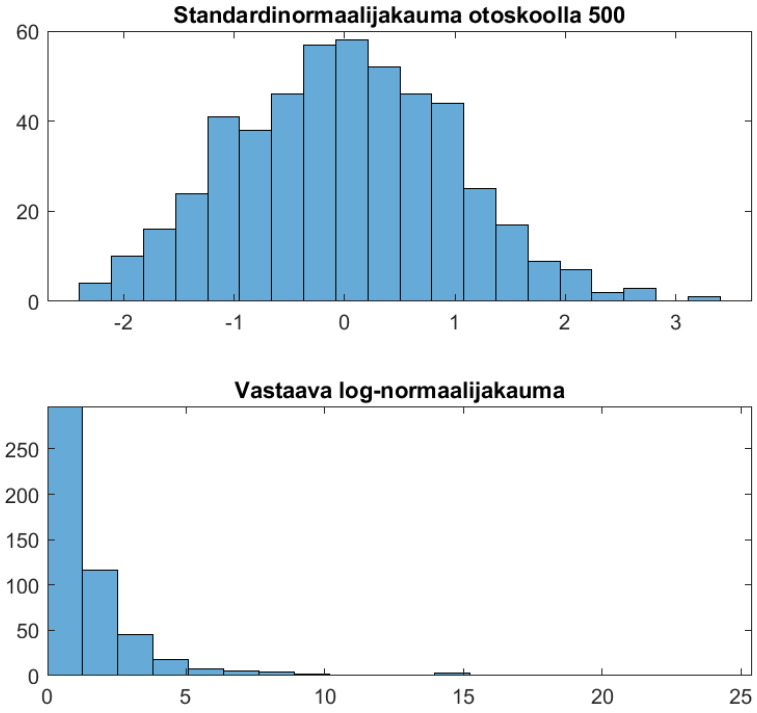
```
% Maaritetaan ns. random seed, jotta saadut tulokset voidaan
% tarvittaessa toistaa
rng(666);
% Generoidaan 500 normaalijakautunutta lukua
% odotusarvolla 0 ja keskihajonnalla 1
r500 = randn(500, 1);
% Tehdaan eksponenttimuunnos
ex500 = exp(r500);
% Plotatataan molemmista jakaumista histogrammit
figure
subplot(2,1,1);
histogram(r500,20);
subplot(2,1,2);
histogram(ex500,20);
...
```

Kuvan 1 ylemmästä histogrammista huomataan jo selkeä normaalijakauman muoto, vaikka satunnaisuuden ja verrattaen pienen otoskoon vuoksi jotkin pylväät ovat ympäristöönsä nähden liian korkeita tai matalia. Korkeimpia ne ovat kuitenkin keskimäärin mitä lähempänä nollaa (standardinormaalijakauman keskiarvoa) ollaan, kuten pitääkin.

Alempi histogrammi eli log-normaalijakauma kokoaa suurimman osan normaalijakauman datapisteistä myös lähelle arvoa yksi, mikä on täysin loogista, onhan $e^0 = 1$. Tietysti myös suurimmat arvot ovat merkittävästi suurempia kuin normaalijakaumassa, koska e^x kasvaa paljon nopeammin kuin x.

1.2 Otoskoon merkitys

Estimoidaan vielä edellisessä osiossa generoidulle log-normaalijakaumalle sekä sitä pienemälle sadan ja suuremmalle kymmenentuhannen luvun jakaumalle parametrit ja 95% luottamusväli seuraavasti.



Kuva 1: Normaali- ja log-normaalijakaumien histogrammit.

```
% Generoidaan jakaumat myos sadalla ja kymmenellatuhannella luvulla
...
% Estimoidaan log-normaalijakaumien parametrit
% ja lasketaan 95% luottamusvalit
[param500, ci500] = lognfit(ex500)
[param100, ci100] = lognfit(ex100)
[param10000, ci10000] = lognfit(ex10000)
```

Tulokset ovat seuraavat:

100 lukua:

- estimoidut parametrit: -0.0820 ja 0.9803
- parametrien luottamusväli: $[-0.2765, 0.1125]$ ja $[0.8607, 1.1388]$

500 lukua:

- estimoidut parametrit: -0.0126 ja 0.9829
- parametrien luottamusväli: $[-0.0989, 0.0738]$ ja $[0.9255, 1.0479]$

10000 lukua:

- estimoidut parametrit: 0.0035 ja 1.0035
- parametrien luottamusväli: $[-0.0162, 0.0232]$ ja $[0.9897, 1.0176]$

Huomataan siis selvästi, että otoskoon kasvattaminen tuottaa parametreille arvoja yhä lähempänä arvoja 0 ja 1 sekä antaa pienemmän luottamusvälin.

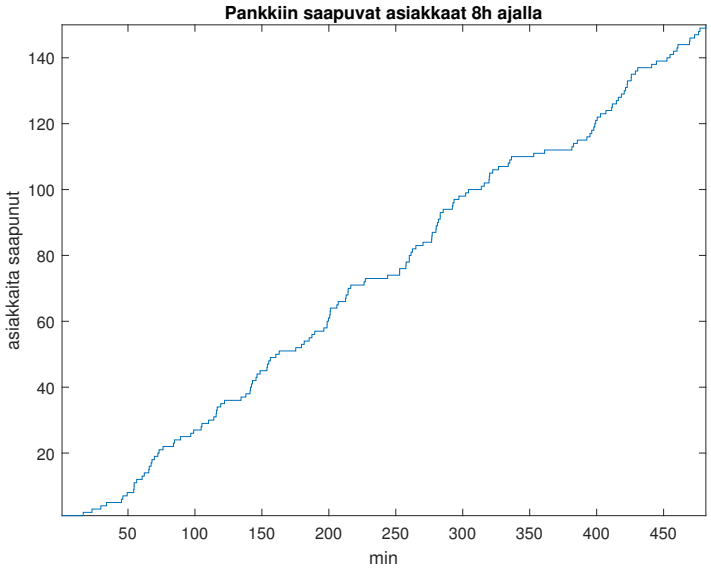
2 Palvelujonon Monte Carlo -simulointi

Simuloidaan tilannetta, jossa pankkiin saapuu asiakkaita satunnaisesti siten, että kahden peräkkäisen asiakkaan väli noudattaa Exp(3)-jakaumaa. Lisäksi kunkin asiakkaan palvelemiseen kuluva aika noudattaa jakaumaa Exp(5). Halutaan selvittää, mikä on optimaalinen määrä kassoja, jotta jonon pituus pysyisi kohtuullisena.

2.1 Asiakkaiden kertyminen

Aloitetaan mallintaminen tutkimalla ensin pelkästään asiakkaiden saapumista pankkiin siten, ettei kukaan poistu pankista. Eli mallinnetaan pelkkää jonon kertymistä.

Kirjoitetaan tarvittava while-luuppi asiakkaiden saapumisen mallintamiseksi ja plotataan saatu kertymä (kuvan 2 kuvaaja) seuraavanlaisella MATLAB-koodilla:



Kuva 2: Asiakkaiden saapuminen pankkiin, kun kahden peräkkäisen asiakkaan saapumisväli on eksponenttijakautunut odotusarvolla 3 minuuttia.

```
% Mallinnetaan asiakkaiden kertymista kahdeksan tunnin ajan
t=0;
asiakkaat = 0;
ajat = [];
while t<8*60
    t=t+exprnd(3);
    asiakkaat=asiakkaat+1;
    ajat = [ajat,t];
end
kertyma = 1:asiakkaat;
% Plotataan asiakkaiden kertyminen
figure
stairs(ajat,kertyma)
...
```

2.2 Asiakkaiden poistumisen huomioiminen

Lisätään simulaatioon asiakkaiden poistuminen pankista. Oletetaan, että asiakkaita poistuu pankista vasta kun he ovat saaneet palvelua, jonka kesto on jakaumaltaan Exp(5).

Eksponenttijakauman unohtamisominaisuuden nojalla se, kuinka kauan asiakas on jo ollut jonossa tai palveltavana ei vaikuta siihen, kuinka kauan hänellä vielä kyseisessä toimituksessa tämän jälkeen kestää. Täten simulointi

voidaan toteuttaa yksinkertaisesti generoimalla lukuja kummastakin jakaumasta (Exp(3) ja Exp(5)) ja käsitellä ainostaan näistä ensin tapahtuva tapahtuma kullakin iteraatiolla.

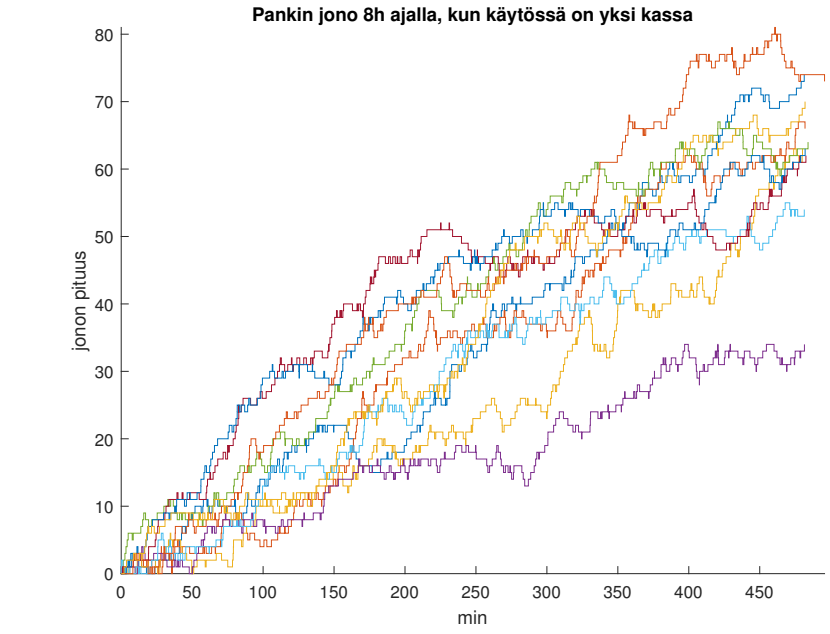
Luodaan seuraavanlainen funktio `jonotus`, joka ottaa parametreinaan kassojen määrän k, asiakkaiden saapumisvälin ja palveluajan odotusarvot a ja p minuutteina sekä simuloitavan ajan h tunteina.

```
function [ajat, jono] = jonotus(k,a,p,h)
% Alustetaan muuttujat
t=0;
jono = [0];
ajat = [0];
% Ajetaan simulaatiota h tuntia
while t<h*60
    % Arvotaan seuraavan asiakkaan saapumisaika
    asiakas_saapuu = t+exprnd(as);
    % Alustetaan palvelun valmistuminen aarettomaksi vertailua varten
    palvelu_valmistuu = Inf;
    % Arvotaan kassojen maaran verran palvelun kestoaikoja ja
    % valitaan niista lyhyin
    for i=1:k
        palvelu_valmistuu = min(palvelu_valmistuu,t+exprnd(pal));
    end
    % Tutukitaan tapahtuuko uuden asiakkaan sisaantulo vai
    % palvelun valmistuminen ensin ja pidennetaan tai lyhennetaan
    % jonoa sen mukaan seka asetetaan tapahtuma-aika muuttujaan t
    if (asiakas_saapuu < palvelu_valmistuu)
        jono = [jono, jono(end)+1];
        t = asiakas_saapuu;
    else
        jono = [jono,max(jono(end)-1,0)];
        t = palvelu_valmistuu;
    end
    % Lisataan tapahtuma-aika aikavektoriin
    ajat = [ajat,t];
end
```

2.3 Yhden kassan simulaatio

Piirretään kymmenen aikasarjaa jonon kehittymisestä päällekkäin (kuva 3) yksinkertaisella for-luupilla:

```
for i=1:10
    [ajat,jono] = jonotus(1,3,5,8);
    stairs(ajat,jono)
end
```



Kuva 3: Kymmenen mahdollista jonon pituuden kehitystä.

Kuvan 3 mukaan jonon pisin pituus voi vaihdella ainakin 30 ja 80 välillä, mikä johtuu juurikin tapahtumien satunnaisuudesta.

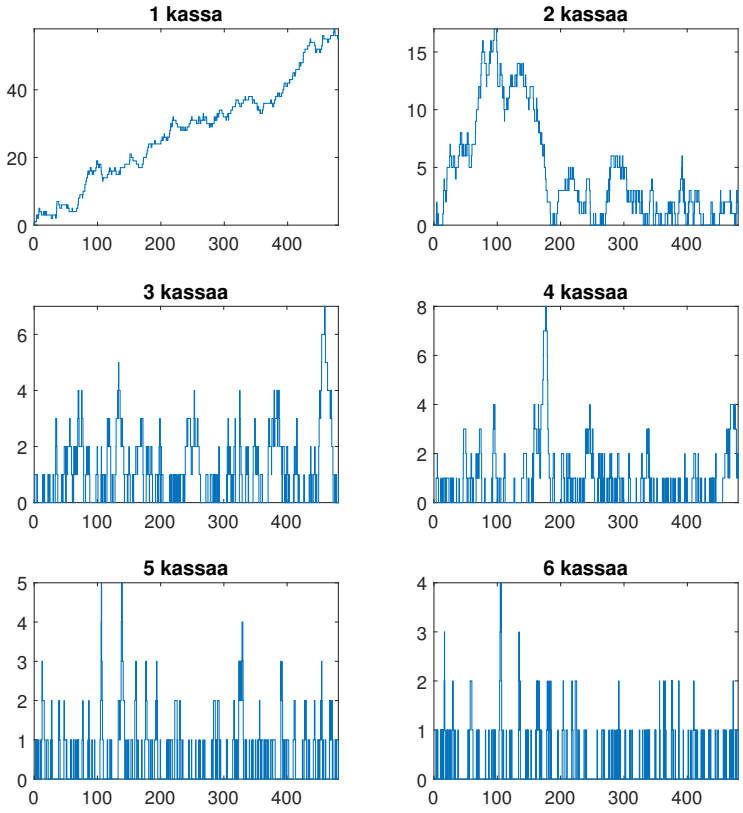
2.4 Simulaatio eri kassojen määrillä

Simuloidaan nyt jonon pituutta eri kassojen määrällä k piirtämällä kuvaajat tapauksista $k \in \{1, ..., 6\}$ toisella for-luupilla.

```
for k=1:6
    subplot(3,2,k)
    [ajat,jono] = jonotus(k,3,5,8);
    stairs(ajat,jono)
    ...
end
```

Kuvasta 4 huomataan, että kolmen palvelevan kassan jälkeen uusien lisääminen ei enää merkittävästi lyhennä keskimääräistä eikä pisintä jonon pituutta.

Pankinjohtajan kannalta siis optimaalinen kassojen määrä lienisi kolme tai neljä, riippuen tietenkin kassahenkilöiden työllistyskustannuksista ja siitä, kuinka tasaisena hän tahtoisi palvelun pitää. Pääluottamusmiehellä olisi tietysti päällimmäisenä mielessään työllistävä vaikutus, jolloin tämän mielestä varmasti olisi parempi pitää auki jopa viittä kassaa.



Kuva 4: Jonon pituuden muutos eri kassojen määrällä.

3 Kotitehtävä: Newsvendor-malli simuloiden

Autetaan jälleen kutosviikolta tuttua Vesa-vekotinmyyjää optimoimaan tuottonsa selvittämällä optimaalinen tilauksen koko. Tällä kertaa uudet vekottimet ovat muotituotteita, joten myymättä jääneet joutavat roskiin myyntijakson jälkeen.

Vekotinten kysynytä D (kpl) on tasajakautunut välillä $[0, 300]$. Tilauskustannus on $c = 30€/\text{kpl}$ ja myyntihinta $p = 120€/\text{kpl}$. Halutaan selvittää kuinka monta vekotinta Vesan kannattaisi tilata.

Mutuillaan ensin hetki ennen tilanteen simuloimista. Vesan pitää myydä neljäsosa tilaamistaan vekottimista, jottei ota takkiin bisneksessään. Ehkä tähän suhteeseen liittyen mieleni tekisi sanoa, että optimaalinen tilausmäärä olisi kolme neljäsosaa kolmestasadasta eli 225 kappaletta. Toisaalta olen ehkä maailman historian surkein arvioimaan asioita, joten eiköhän mallineta tilannetta mieluummin matemaattisesti.

3.1 Myyntivoiton simulointi

Laaditaan funktio `myyntivoitto`, joka ottaa parametreikseen tilausmäärän q ja havaintojen määrän n.

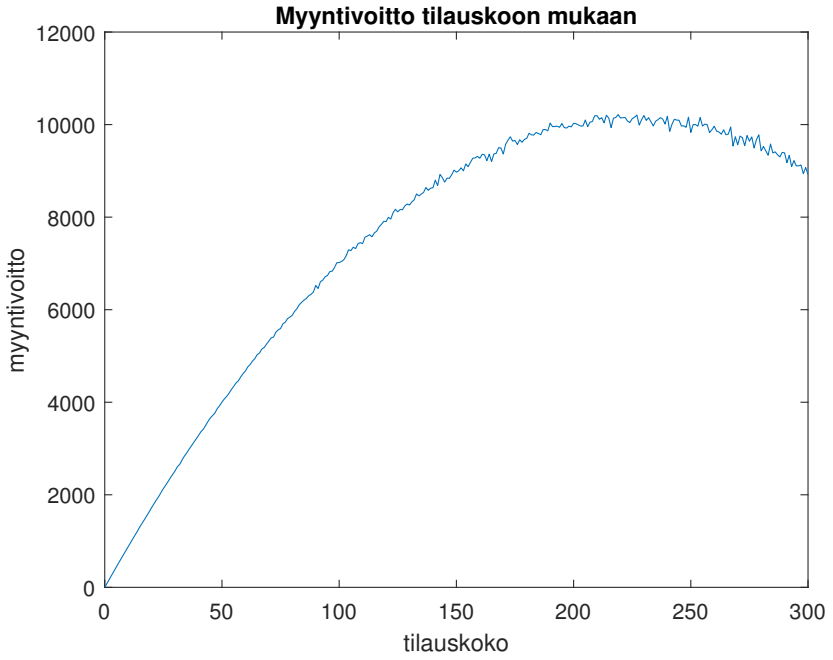
```
function voitot = myyntivoitto(q,n)
% Generoidaan n kappaletta tasajakautuneita satunnaislukuja valilla 0-300
D = 300*rand(n,1);
% Lasketaan myyntivoitto kullakin kysynnan arvolla
voitot = [];
for i=1:n
    voitot(i) = min(D(i),q)*120-q*30;
end
```

Lasketaan funktiota hyödyntäen esimerkkitapauksessa $q = 150$, millä todennäköisyydellä Vesa jää miinukselle. Alta löytyvällä komennolla ajetaan simulaatiota 10000 kertaa ja lasketaan tappiollisten tapausten osuus kaikista tapauksista. Tulokseksi saadaan tappiotodennäköisyys 12.6%.

```
length(find(myyntivoitto(150,10000)<0))/10000
```

3.2 Optimaalinen tilauskoko

Ajetaan seuraavaksi simulaatio useita kertoja tilauskoolla $q \in [0, 300]$ ja lasketaan tuloksista odotusarvo myyntivoitolle kussakin tapauksessa. Kun simulointien määrä on tarpeeksi suuri (alla olevassa skriptissä 1000), tuloksena saatu maksimaalisen tuoton odotusarvo alkaa lähestyä todellista odotusarvoa. Lasketaan odotusarvot ja plotataan kuvaaja (kuva 5) voitoista tilausmäärän funktiona:



Kuva 5: Myyntivoiton estimoitu odotusarvo tilausmäärän funktiona.

```
% Estimoidaan myyntivoiton odotusarvo tilausmaarilla 0-300
E = [];
n = 10000;
for q=0:300
    % Ajetaan simulaatio
    mv = myyntivoitto(q,n);
    % Lasketaan odotusarvot
    E(q+1) = sum(mv)/n;
end
% Maksimivoitto
max(E)
% Maksimivoiton tuottava tilauskoko
find(E==max(E))
% Plotataan myyntivoitto tilauskoon funktiona
q = 0:300;
figure
plot(q,E)
...
```

Kuvan 5 kuvaajasta nähdään, että tilauksen optimaalinen koko on kuin onkin jossakin 225 kappaleen hujakoilla. Suoraan odotusarvovektorista laskettu maksimaalisen voiton tuottava tilauskoko on `find(E==max(E)) = 220`. Se ei kuitenkaan ole vielä järin tarkka, mikä huomataan kuvaajan röpelöisyydestä

juurikin maksimin paikkeilla.

Vesa voi kuitenkin jo näillä mallinnuksilla todeta parhaan tilausmäärän olevan jossakin 220 paikkeilla. Hän voi sitten itse pohtia, kuinka paljon riskiä on valmis ottamaan.

3.3 Epäluotettava tavarantoimittaja

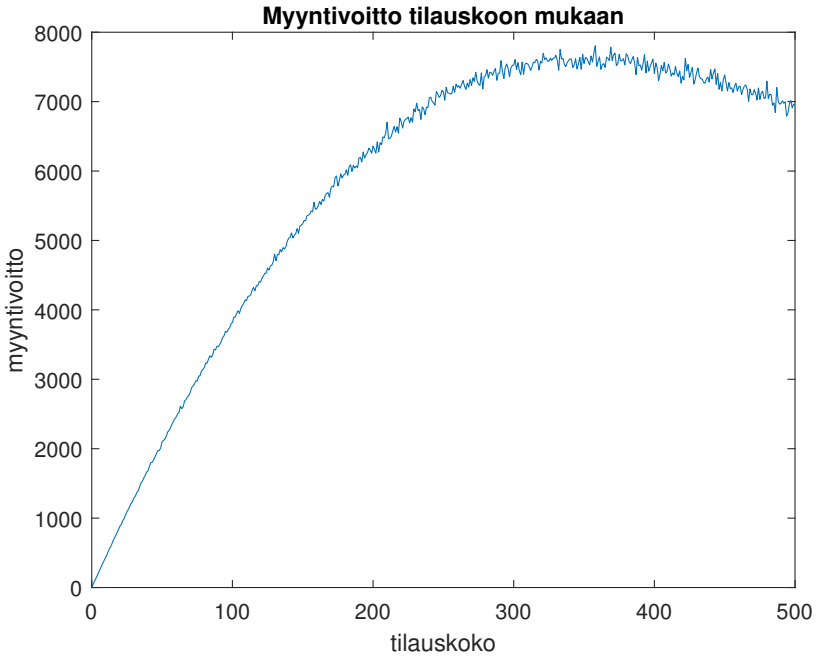
Vekotintehtaan ei voikaan luottaa toimittavan kaikkia tilattuja tuotteita, jolloin Vesan ongelmaan tulee yksi muuttuja lisää. Toimitettujen vekottimien prosenttiosuus Z on tasajakautunut välille $[0, 1]$.

Luodaan uusi funktio `myyntivoitto2`, joka ottaa huomioon toimituksen epävarmuuden:

```
function voitot = myyntivoitto2(q,n)
% Generoidaan n kappaletta tasajakautuneita satunnaislukuja valilla 0-300
D = 300*rand(n,1);
% Generoidaan n kappaletta tasajakautuneita satunnaislukuja valilla 0-1
Z = rand(n,1);
% Lasketaan myyntivoitto kullakin kysynnan arvolla
voitot = [];
for i=1:n
    qq = q*Z(i); % Toimituksen toteutuma
    voitot(i) = min(D(i),qq)*120-qq*30;
end
```

Estimoidaan odotusarvot tilauskoilla 0-500 täysin vastaavanlaisella for-luupilla kuin aiemmin. Tulokseksi saadaan kuva 6, josta huomataan kaksi asiaa: tuoton odotusarvo on kaikilla tilauskoilla selvästi pienempi (kuten sopii odottaa) sekä optimaalinen tilauskoko on selvästi aiempaa suurempi. Estimoitujen odotusarvojen perusteella tässä tapauksessa parhaan voiton tuottava tilauskoko olisi 359 vekotinta.

Aiemmin kuudennella viikolla laskettu analyyttinen ratkaisu optimaaliselle tilauskoolle oli $200\sqrt{3} \approx 346$, joka osuu jo melko lähelle simuloiden saatua arvoa. Suuremmalla otannalla päästäisiin varmasti vielä lähemmäksi kyseistä arvoa.



Kuva 6: Myyntivoiton estimoitu odotusarvo tilausmäärän funktiona, kun tilauksen toteutuma on epävarma.