Aalto-yliopisto
Perustieteiden korkeakoulu
Teknillisen fysiikan ja matematiikan tutkinto-ohjelma

Kaksintaistelun approksimatiivinen mallintaminen

kandidaatintyö
16.9.2013

Juho Roponen

Aalto-yliopisto

AALTO-YLIOPISTO PERUSTIETEIDEN KORKEAKOULU PL 11000, 00076 Aalto http://www.aalto.fi	KANDIDAATINTYÖN TIIVISTELMÄ

Tekijä: Juho Roponen

Työn nimi: Kaksintaistelun approksimatiivinen mallintaminen

Tutkinto-ohjelma: Teknillisen fysiikan ja matematiikan tutkinto-ohjelma

Pääaine: Systeemitieteet	Pääaineen koodi: F3010

Vastuuopettaja(t): Prof. Ahti Salo

Ohjaaja(t): ST Esa Lappi

Tiivistelmä:

Taistelumallinnus on tutkimusala, jolla tehdään jatkuvasti kehitystyötä. Suurien taisteluiden tarkka mallintaminen on nykyään erittäin työlästä ja aikaavievää puuhaa ja isonkin taistelun kulku voi muuttua merkittävästi kahden pienemmän joukon kohtaamisen tulosten seurauksena. Tästä syystä tarvitaan tehokkaita ja nopeita algoritmejä yksittäisten joukkojen kohtaamisten lopputulosten arviointiin.

Tässä työssä tarkastellaan kahta approksimatiivista stokastista mallia, joilla voidaan ratkoa pienten joukkojen välisiä kaksintaisteluita. Menetelmät perustuvat kahteen erilaiseen riippumattomuusolettamaan joukkojen vahvuusjakaumista. Menetelmiä verrataan referenssimalliin, joka ei tee minkäänlaisia riippumattomuusolettamia.

Menetelmien nopeuden lisäksi tarkastellaan niiden kykyä ennustaa kaksintaistelun voittava osapuoli sekä joukkojen vahvuudet kaksintaistelun jälkeen. Vertailun tulosten perusteella esitetään arvio mallien soveltuvuudesta erilaisiin käyttötarkoituksiin.

Päivämäärä: 16.9.2013	Kieli: suomi	Sivumäärä: 17

Avainsanat: stokastinen, mallintaminen, kaksintaistelu

Sisältö

1 Johdanto

Tässä työssä tutkitaan kahden hieman toisistaan eroavan laskennallisen menetelmän kykyä ennustaa kahden jalkaväkijoukon välisen taistelun lopputuloksia. Tarkoille ja nopeille menetelmille tällaisten ongelmien ratkaisemiseen on tarvetta, koska esimerkiksi prikaatitason taistelua mallinnettaessa tulee usein vastaan tilanteita, että pienempienkin joukkojen väliset kohtaamiset voivat vaikuttaa siihen, miten koko mallinnettava skenaario jatkuu siitä eteenpäin. ([Lappi and Åkesson, 2011], [Sormunen et al., 2013]) Tällaisissa tilanteissa ollaan ennen kaikkea kiinnostuneita siitä, mitkä ovat joukon mahdollisuudet voittaa taistelu ja missä kunnossa joukko on taistelun jälkeen, jos se kykenee sen voittamaan.
Kahden joukon välisen taistelun lopputuloksen arviointiin on olemassa useita erilaisia työkaluja. Näistä yleisesti ovat käytössä Lanchesterin yhtälöihin perustuvat mallit sekä joukkojen voimasuhteiden arviointiin perustuvat mallit kuten QJM-analyysi. ([Hyytiäinen, 2003]) Näiden lisäksi käytössä on pienempään mittakaavaan soveltuvia niin sanottuja lavettitason malleja, jossa pienin toimija on yksittäinen sotilas tai laite ja itse simulointi tapahtuu Monte Carlo -menetelmällä. Näissä menetelmissä on kuitenkin tiettyjä ongelmia; Lanchesterin yhtälöt tekevät hyvin paljon yksinkertaistavia oletuksia, joista irrottautuminen tekee yhtälöistä vaikeita tai mahdottomia ratkaista. ([Kangas, 2005]) QJM-analyysi soveltuu huonosti käytettäväksi Suomessa, koska se olettaa joukon kooksi vähintään prikaatin. ([Jaakola, 2004]) Vastaavasti myös lavettitason mallit ovat rajoittuneita melko pieniin joukkoihin.
Joukkue- ja prikaatitason välille soveltuvia mallinnustyökaluja ei ole olemassa ainakaan julkisesti saatavilla olevan tiedon mukaan juuri lainkaan ja juuri tämän kokoluokan taistelut ovat kiinnostavia suomalaisten kannalta. ([Lappi et al., 2012b]) Tästä syystä tässä työssä tukeudutaan voimakkaasti operaatioanalyysityökalu Sandista ([Lappi, 2008]) varten kehitettyihin stokastisiin laskentamalleihin, koska niistä on mahdollista löytää tietoa riittävän kattavasti.

2 Teoreettinen tausta

Tutkitaan kahden joukon välistä suoratuliaseilla (esim. kiväärit tai konekiväärit) käytävää taistelua. Merkitään B_t:llä ja R_t:llä joukkojen Sininen ja Punainen vahvuuksia ajanhetkellä $t = 0, 1, \ldots, T$, missä $T \in \aleph$ on simuloinnin aika-askeleiden kokonaislukumäärä. Joukkojen aloitusvahvuudet ovat ennalta määrättyjä vakioita, kun taas kaikki muut vahvuudet ovat satunnaismuuttu-

jia. Kumpikaan joukko ei saa vahvistuksia taistelun aikana, joten $B_t \leq B_{t-1}$ ja $R_t \leq R_{t-1}$. Molemmilla yksiköillä on minimivahvuus $\underline{b} \in \{1, \ldots, B_0\}$ ja $\underline{r} \in \{1, \ldots, R_0\}$, jolla se pystyy toimimaan. Jos yksikön vahvuus putoaa tämän minimivahvuuden alle, ei yksikkö pysty enää jatkamaan taistelua ja se lasketaan lyödyksi. Tässä tapauksessa yksiköiden vahvuudet eivät enää muutu, koska taistelu lasketaan päättyneeksi. (Tässä mallinnusongelmassa ei oteta kantaa siihen, mitä yksiköille käy tämän jälkeen.) Formaalisti siis, jos $B_t < \underline{b}$ tai $R_t < \underline{r}$ jollekin t niin $B_s = B_t$ ja $R_s = R_t$ kaikille $s \geq t$. Jokaiselle ajanhetkelle on siis neljä erilaista mahdollisuutta taistelun tilaksi.

- taistelu jatkuu, $C_t = \{B_t \geq \underline{b}, R_t \geq \underline{r}\}$
- Sininen joukko on voittanut, $W_{B,t} = \{B_t \geq \underline{b}, R_t < \underline{r}\}$
- Punainen joukko on voittanut, $W_{R,t} = \{B_t < \underline{b}, R_t \geq \underline{r}\}$
- molemmat joukot on lyöty, $D_t = \{B_t < \underline{b}, R_t < \underline{r}\}$

Koska joukkojen vahvuudet B_t ja R_t laskevat taistelun edetessä, C_t:n todennäiköisyys laskee ja lopputilojen, $W_{b,t}$, $W_{r,t}$ ja D_t todennäköisyydet kasvavat.

2.1 Referenssimenetelmä

Kuvailen ensimmäisenä tässä työssä käytössä olevan referenssimenetelmän, johon muiden approksimatiivisten menetelmien tarkkuutta ja nopeutta verrataan. Menetelmä on sama kuin [Lappi et al., 2012a] konferenssipaperissa esitelty. Tämä helpottaa tulosten vertailua kyseiseen paperiin. Referenssimenetelmä itsessään on yleistys Sandiksessa ([Lappi and Pottonen, 2006]) nykyisin käytössä olevasta *pistetuli*-mallista ja se muistuttaa osittain stokastista Lanchester-mallia.
Merkitään tiloihin liittyviä todennäiköisyyksiä P:llä. Refenssimenetelmässä simuloinnin t:s aika-askel otetaan seuraavasti.

1. Jokaisen aika-askeleen alussa joukkojen vahvuudet ovat B_{t-1} ja R_{t-1}, vastaavasti.
2. Jokainen Sininen (Punainen) sotilas tulittaa riippumattomasti keskimäärin $\lambda_B > 0$ ($\lambda_R > 0$) ammusta tähdättynä sattumanvaraisesti ja tasaisesti R_{t-1} (B_{t-1}) kohteeseen, jotka ovat vihollisen sotilaita.
3. Jokainen Sinisen (Punaisen) sotilaan ampuma ammus tuottaa osuman, joka lamauttaa vastustajan sotilaan, todennäköisyydellä $p_B \in (0, 1)$ ($p_R \in (0, 1)$)

4. Jokaisen aika-askeleen lopussa lamauttavan osuman saaneet sotilaat poistetaan taistelukentältä ja uudet vahvuudet B_t ja R_t lasketaan

Tällä tavalla toimien vahvuudet (B_t, R_t) seuraavata Markovin ketjua binomisiirtymätodennäköisyyksin

$$\begin{aligned} P[B_t = b, R_t = r | B_{t-1} = b_{-1}, R_{t-1} = r_{-1}] = \\ \binom{b_{-1}}{b_{-1} - b} \pi_R(b_{-1}, r_{-1})^{b_{-1}-b} (1 - \pi_R(b_{-1}, r_{-1}))^b \\ \times \binom{r_{-1}}{r_{-1} - r} \pi_B(b_{-1}, r_{-1})^{r_{-1}-r} (1 - \pi_B(b_{-1}, r_{-1}))^r \end{aligned} \quad (1)$$

jos $b_{-1} \geq \underline{b}$, $b_{-1} \geq b$, $r_{-1} \geq \underline{r}$ ja $r_{-1} \geq r$, missä

$$\pi_B(b_{-1}, r_{-1}) = 1 - (1 - p_B)^{\frac{\lambda_B b_{-1}}{r_{-1}}} \text{ ja } \pi_R(b_{-1}, r_{-1}) = 1 - (1 - p_R)^{\frac{\lambda_R r_{-1}}{b_{-1}}}$$

.

Muutoin asetetaan

$$P[B_t = b, R_t = r | B_{t-1} = b_{-1}, R_{t-1} = r_{-1}] = \begin{cases} 1, & \text{jos } b = b_{-1} \text{ ja } r = r_{-1} \\ 0, & \text{jos } b \neq b_{-1} \text{ tai } r \neq r_{-1} \end{cases} \quad (2)$$

Yhtälöistä (1) ja (2) saadaan alkiot Markovin ketjun (B_t, R_t) tilasiirtomatriisiin. Näin ollen yhteisjakauman (B_t, R_t) arviointi päätyy alkutilavektorin (B_0, R_0) kertomiseen tilasiirtomatriisilla t kertaa. Todennäköisyydet C_t, $W_{b,t}$, $W_{r,t}$ ja D_t saadaan sitten laskettua yhteisjakaumasta. Tilasiirtomatriisin dimensio on $(B_0+1)(R_0+1) \times (B_0+1)(R_0+1)$, mutta se on suhteellisen harva, joten pienillä aloitusvahvuuksilla B_0 ja R_0 matriisin muodostaminen ja kertolaskut ovat laskennallisesti vielä hallittavissa. Aloitusvahvuuksien kasvaessa vaatimukset laskentateholle kasvavat nopeasti, mikä kannustaa etsimään vaihtoehtoisia malleja.

2.2 Ensimmäinen approksimatiivinen menetelmä

Esitän tässä ensimmäisen vertailussa olevan approksimatiivisen menetelmän, joka pyrkii olemaan tuloksiltaan lähellä referenssimallia ainakin taistelun lopputilojen todennäköisyyksien osalta. Menetelmä on myös laskennallisesti selkeästi referenssimenetelmää nopeampi ([Lappi et al., 2012a]).

Merkitään approksimatiiviseen menetelmään liittyviä todennäköisyyksiä $\bar{P}$:llä. Menetelmä perustuu oletukseen, että jokaisella aika-askeleella $t = 1, 2, \ldots, T$ vahvuudet B_t ja R_t ovat riippumattomia ehdolla, että *taistelu on jatkunut ajanhetkellä* $t-1$, siis

$$\bar{P}[B_t = b, R_t = r|C_{t-1}] = \bar{P}[B_t = b|C_{t-1}]\bar{P}[R_t = r|C_{t-1}]. \tag{3}$$

Otetaan t:s aika-askel samalla tavoin kuin referenssimenetelmässä paitsi, että Punaisen ja Sinisen edelliset vahvuudet otetaan toisistaan riippumatta ehdollisista reunajakaumista (marginal conditional distributions) $\bar{P}[B_{t-i} = b_{-1}]$ ja $\bar{P}[R_{t-i} = r_{-1}]$ vastaavasti. Asetetaan siis

$$\begin{aligned}\bar{P}[B_t = b|C_{t-i}] = &\sum_{b_{-1}=\underline{b}}^{B_0} \sum_{r_{-1}=\underline{r}}^{R_0} \bar{P}[B_{t-1} = b_{-1}|C_{t-1}]\bar{P}[R_{t-1} = r_{-1}|C_{t-1}] \\ &\times \binom{b_{-1}}{b_{-1}-b} \pi_R(b_{-1}, r_{-1})^{b_{-1}-b}(1 - \pi_R(b_{-1}, r_{-1}))^b, \end{aligned} \tag{4}$$

missä $\pi_R(b_{-1}, r_{-1})$ ja siihen liittyvät parametrit ovat samoja kuin referenssimenetelmässäkin. Todennäköisyydelle $\bar{P} = [R_t = r|C_{t-i}]$ muotoillaan myös täysin analoginen kaava. Koska taistelu jatkuu ainoastaan, jos Sinisen vahvuus on vähintään $\underline{b}$, saadaan

$$\bar{P}[B_t = b|C_t] = \begin{cases} \frac{\bar{P}=[B_t=b|C_{t-i}]}{\bar{P}=[B_t \geq b|C_{t-i}]}, & \text{jos } b \geq \underline{b} \\ 0, & \text{jos } b < \underline{b} \end{cases} \tag{5}$$

ja Punaisen vahvuudelle $\bar{P} = [R_t = r|C_{t-i}]$ saadaan vastaava kaava.
Tarkastellaan sitten approksimatiivisen menetelmän tärkeimpiä ominaisuuksia. Koska $C_t \subset C_{t-1}$, ehdollisen riippumattomuuden nojalla (3), saadaan taistelun jatkumiselle laskettua todennäköisyys

$$\bar{P}[C_t] = \bar{P}[C_t \cap C_{t-1}] = \bar{P}[B_t \geq \underline{b}|C_{t-1}]\bar{P}[R_t \geq \underline{r}|C_{t-1}]\bar{P}[C_{t-1}]. \tag{6}$$

Lisäksi saadaan

$$\bar{P}[B_t = b, R_t = r] = \bar{P}[B_t = b|C_{t-1}]\bar{P}[R_t = r|C_{t-1}]\bar{P}[C_{t-1}] + \bar{P}[B_{t-1} = b, R_{t-1} = r] \tag{7}$$

ehdolla että $b < \underline{b}$ tai $r < \underline{r}$. Tämä identiteetti mahdollistaa ratkaisevien todennäiköisyyksien $\bar{P}[B_t = b, R_t = r]$, missä $b < \underline{b}$ tai $r < \underline{r}$, arvioimisen rekursiivisesti ilman, että pidetään kirjaa koko yhteisjakaumasta (B_t, R_t). Tällainen menettely vaatii ainoastaan, että pidetään kirjaa ehdollisista reunajakaumista $\bar{P}[B_t = b|C_{t-1}]$ ja $\bar{P}[R_t = r|C_{t-1}]$ aika-askeleeseen t asti. Yhtälöstä (7) saadaan suoraan myös rekursiiviset kaavat taistelun lopputilojen

laskentaan:

$$\bar{P}[W_{B,t}] = \bar{P}[B_t \geq \underline{b}|C_{t-1}]\bar{P}[R_t < \underline{r}|C_{t-1}]\bar{P}[C_{t-1}] + \bar{P}[W_{B,t-1}], \quad (8)$$

$$\bar{P}[W_{R,t}] = \bar{P}[B_t < \underline{b}|C_{t-1}]\bar{P}[R_t \geq \underline{r}|C_{t-1}]\bar{P}[C_{t-1}] + \bar{P}[W_{R,t-1}], \quad (9)$$

ja

$$\bar{P}[D_t] = \bar{P}[B_t < \underline{b}|C_{t-1}]\bar{P}[R_t < \underline{r}|C_{t-1}]\bar{P}[C_{t-1}] + \bar{P}[D_{t-1}]. \quad (10)$$

Muistetaan, että aluksi $\bar{P}[W_{B,t}] = \bar{P}[W_{R,t}] = \bar{P}[D_t] = 0$

On myös muistettava, että lopputilojen todennäköisyyksien lisäksi ollaan myös kiinnostuneita yksiköiden vahvuuksien ehdollisista jakaumista tilanteessa, jossa yksikkö on joko voittanut tai lyöty. Referenssimenetelmässä nämä lasketaan suoraan yhteisjakaumasta (B_t, R_t), mutta tämän approksimatiivisen menetelmän kanssa täytyy menetellä hieman eri tavoin. Ehdollisille jakaumille saadaankin johdettua rekursiiviset kaavat yhtälöiden (8), (9) ja (10) tapaan. Sinisen joukon vahvuusjakaumalle pätee ehdolla, että se on voittanut

$$\bar{P}[B_t = b|W_{B,t}] = \frac{\bar{P}[B_t = b|C_{t-1}]\bar{P}[R_t < \underline{r}|C_{t-1}]\bar{P}[C_{t-1}] + \bar{P}[B_{t-1} = b|W_{B,t-1}]\bar{P}[W_{B,t-1}]}{\bar{P}[W_{B,t}]} \quad (11)$$

kaikilla $b \geq \underline{b}$, kun taas jakaumalle ehdolla, että se on lyöty, pätee

$$\bar{P}[B_t = b|D_t \cup W_{R,t}] = \frac{\bar{P}[B_t = b|C_{t-1}]\bar{P}[C_{t-1}] + \bar{P}[B_{t-1} = b|D_{t-1} \cup W_{R,t-1}](\bar{P}[D_{t-1}] + \bar{P}[W_{R,t-1}])}{\bar{P}[D_t]\bar{P}[W_{R,t}]} \quad (12)$$

kaikilla $b < \underline{b}$. Aloitusjakaumia $\bar{P}[B_0, = b|W_{B,0}]$ ja $\bar{P}[R_0, = r|W_{R,0}]$ ei voida määrittää, koska $\bar{P}[W_{B,0}] = \bar{P}[W_{D,0}] = \bar{P}[D_0] = 0$, mutta se ei ole ongelma, koska niiden vaikutus yhtälöissä (11) ja (12) kerrotaan joka tapauksessa nollalla. Jälleen Punaisen vahvuusjakaumille saadaan kirjoitettua aivan analogiset yhtälöt.

2.3 Toinen approksimatiivinen menetelmä

Toinen esittämäni menetelmä on itse ensimmäisestä menetelmästä muokkaamani. Se pyrkii olemaan laskennallisesti ensimmäistä nopeampi häviämättä

merkittävästi laskennan tarkkuudessa. Toinen menetelmä tekee voimakkaamman riippumattomuusolettaman kuin ensimmäinen. Sen sijaan, että oletetaan, että vahvuudet B_t ja R_t ovat riippumattomia ehdolla, että taistelu jatkuu, oletetaan, että ne ovat riippumattomia aina, eli

$$\bar{P}[B_t = b, R_t = r] = \bar{P}[B_t = b]\bar{P}[R_t = r]. \tag{13}$$

Myös laskenta hoidetaan kuitenkin hiukan eri tavoin. Sen sijaan, että laskennassa käytettäisiin ehdollisia reunajakaumia, käytetään koko reunajakaumaa. Reunajakaumille saadaan siis kirjoitettua yhtälöä (4) vastaavasti

$$\begin{aligned}\bar{P}[B_t = b] = &\sum_{b_{-1}=\underline{b}}^{B_0} \sum_{r_{-1}=\underline{r}}^{R_0} \bar{P}[B_{t-1} = b_{-1}]\bar{P}[R_{t-1} = r_{-1}|] \\ &\times \binom{b_{-1}}{b_{-1}-b} \pi_R(b_{-1}, r_{-1})^{b_{-1}-b}(1 - \pi_R(b_{-1}, r_{-1}))^b \\ &+ \sum_{b_{-1}=0}^{\underline{b}-1} \sum_{r_{-1}=0}^{R_0} \bar{P}[B_{t-1} = b_{-1}]\bar{P}[R_{t-1} = r_{-1}|] \times I_b(b, b_{-1}) \\ &+ \sum_{b_{-1}=\underline{b}}^{B_0} \sum_{r_{-1}=0}^{\underline{r}-1} \bar{P}[B_{t-1} = b_{-1}]\bar{P}[R_{t-1} = r_{-1}|] \times I_b(b, b_{-1}),\end{aligned} \tag{14}$$

missä

$$I_b(b, b_{-1}) = \begin{cases} 1, & \text{jos } b = b_{-1} \\ 0, & \text{muutoin.} \end{cases}$$

Punaisen vahvuudet päivitetään aivan vastaavalla tavalla.
Taistelun jatkumisesta täytyy pitää kirjaa aivan yhtälailla kuin ensimmäisessäkin approksimatiivisessa menetelmässä, jotta saadaan selvitettyä kummankin joukon voittotodennäköisyydet ja vahvuusjakaumat siinä tapauksessa, että kyseinen joukko on voittanut. Yhtälöä (6) vastaavasti saadaan

$$\bar{P}[C_t] = \bar{P}[C_t \cap C_{t-1}] = \frac{\bar{P}[B_t \geq \underline{b}]\bar{P}[R_t \geq \underline{r}]\bar{P}[C_{t-1}]}{P[B_{t-1} \geq \underline{b}]\bar{P}[R_{t-1} \geq \underline{r}]}. \tag{15}$$

Myös taistelun lopputiloista on pidettävä kirjaa samankaltaisesti kuin ensimmäisessä approksimatiivisessa menetelmässä. Yhtälöiden (8), (9) ja (10) tapaan voidaan kirjoittaa

$$\bar{P}[W_{B,t}] = \bar{P}[B_t \geq \underline{b}|B_{t-1} \geq \underline{b}]\bar{P}[R_t < \underline{r}|R_{t-1} \geq \underline{r}]\bar{P}[C_{t-1}] + \bar{P}[W_{B,t-1}], \tag{16}$$

$$\bar{P}[W_{R,t}] = \bar{P}[B_t < \underline{b}|B_{t-1} \geq \underline{b}]\bar{P}[R_t \geq \underline{r}|R_{t-1} \geq \underline{r}]\bar{P}[C_{t-1}] + \bar{P}[W_{R,t-1}], \tag{17}$$

ja

$$\bar{P}[D_t] = \bar{P}[B_t < \underline{b}|B_{t-1} \geq \underline{b}]\bar{P}[R_t < \underline{r}|R_{t-1} \geq \underline{r}]\bar{P}[C_{t-1}] + \bar{P}[D_{t-1}]. \tag{18}$$

Myös yksiköiden vahvuusjakaumat siinä tapauksessa, että ne ovat voittaneet täytyy laskea rekursiivisella kaavalla yhtälön (11) tapaan. Sinisen joukon vahvuusjakaumalle pätee ehdolla, että se on voittanut

$$\bar{P}[B_t = b|W_{B,t}] = \frac{\bar{P}[B_t = b|B_{t-1} \geq \underline{b}]\bar{P}[R_t < \underline{r}|R_{t-1} \geq \underline{r}]\bar{P}[C_{t-1}] + \bar{P}[B_{t-1} = b|W_{B,t-1}]\bar{P}[W_{B,t-1}]}{\bar{P}[W_{B,t}]} \tag{19}$$

kaikilla $b \geq \underline{b}$. Jakauma ehdolla, että yksikkö on lyöty on myös laskettava samankaltaisella rekursiivisella kaavalla yhtälön (12) tapaan

$$\bar{P}[B_t = b|D_t \cup W_{R,t}] = \frac{\bar{P}[B_t = b|B_{t-1} \geq \underline{b}]\bar{P}[C_{t-1}] + \bar{P}[B_{t-1} = b|D_{t-1} \cup W_{R,t-1}](\bar{P}[D_{t-1}] + \bar{P}[W_{R,t-1}])}{\bar{P}[D_t]\bar{P}[W_{R,t}]} \tag{20}$$

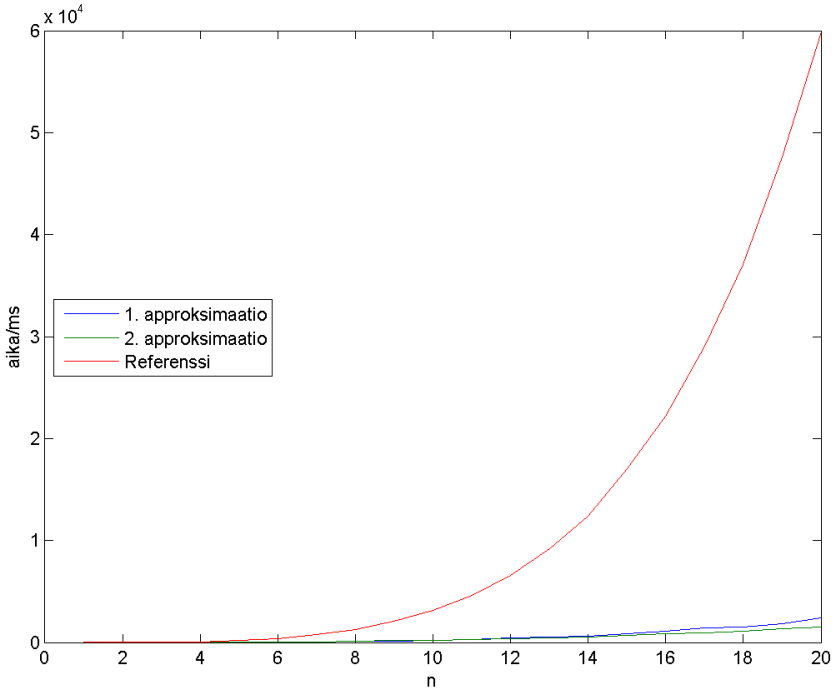
kaikilla $b < \underline{b}$. Punaisen vahvuusjakaumalle saadaan jälleen kirjoitettua täysin analoginen kaava.

3 Tulokset

Arvioin eri menetelmien tehokkuutta toteuttamalla ne Javalla ([Lindholm and Yellin, 1999]) ja tarkastelemalla sitten, miten eri menetelmät selviytyivät erilaisista testiskenaarioista. Tarkkailin, kuinka eri menetelmät suoriutuivat laskenta-aikansa ja tarkkuutensa osalta. Tarkkuutta vertailin tutkimalla kuinka hyvin approksimativiivisten menetelmien tulokset vastasivat referenssimenetelmän tuloksia taistelun lopputilatodennäköisyyksien ja lopputiloja vastaavien vahvuusjakaumien osalta.

3.1 Laskenta-aikojen vertailu

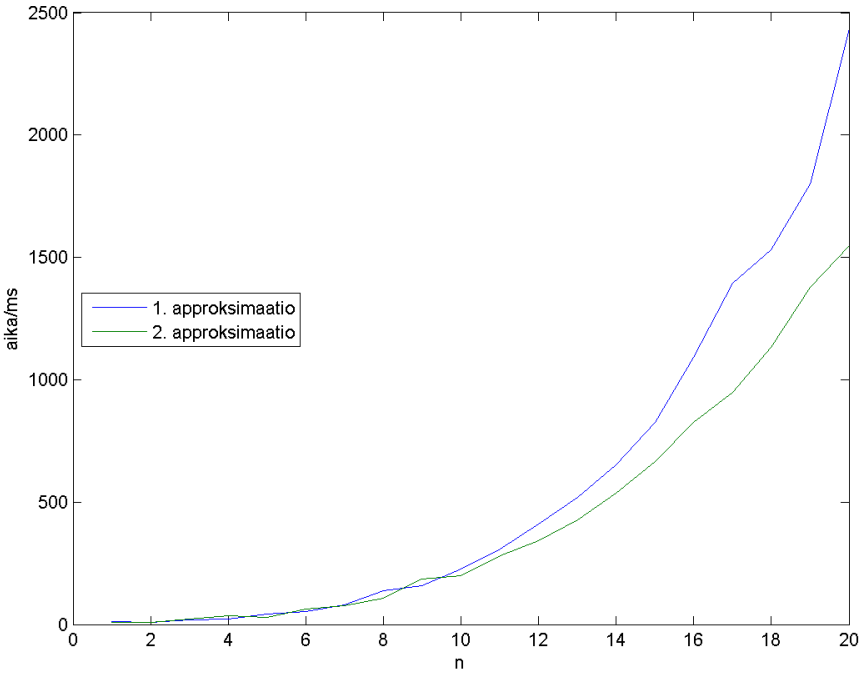
Vertailin eri menetelmiin kuluvaa laskenta-aikaa laskemalla kaikilla niistä joukon testiskenaarioita, joissa $p_B = p_r = 0.02$ ja $\lambda_B = \lambda_R = 2$, mikä vastaa kenttäkokeissa saavutettuja parametrejä jalkaväen sotilaille [Lappi and Pottonen, 2006] ja [Keinonen, 1954]. Joukkojen aloitusvahvuudet muuttivat ajokertojen välillä siten, että $B_0 = n * 5$ ja $R_0 = n * 4$, kun $n = 1, 2, \ldots, 20$, eli B_0 vaihteli viidestä sataan ja R_0 neljästä kahdeksaankymmeneen. Joukkojen taistelua ei ole järkevää mallintaa kaksintaisteluna, jossa jokainen sotilas



Kuva 1: Eri menetelmien laskenta-ajat

voi ampua jokaista vastapuolen sotilasta merkittävästi tätä isommilla joukoilla. Joukkojen pienimmät toimintavahvuudet olivat kaikissa skenaarioissa puolet aloitusvahvuudesta. Kaikilla menetelmillä käytettiin laskennassa 200 aika-askelta.

Kuvassa 1 on näkyvissä kaikkien eri menetelmien käyttämät laskenta-ajat. Kumpikin approksimatiivinen malli on referenssimenetelmää merkittävästi nopeampi. Kaikkien ajankäyttö kasvaa huomattavasti, kun joukkojen koko kasvaa, ja toisella approksimatiivisella menetelmällä aletaan saavuttaa hyötyjä suhteessa ensimmäiseen, kun joukkojen koko kasvaa yli 50:n, kuten kuvasta 2 voidaan helpommin nähdä.



Kuva 2: Approksimatiivisten menetelmien laskenta-ajat

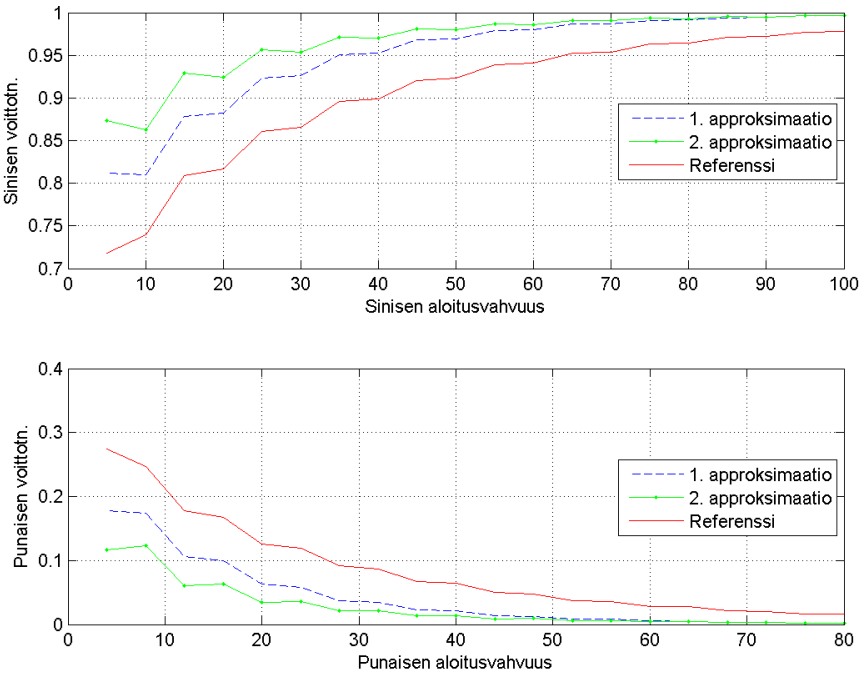
3.2 Voittotodennäköisyyksien virheet

Vertailin approksimatiivisten menetelmien virheitä lopputilatodennäköisyyksissä vertaamalla niiden antamia voittotodennäköisyyksiä referenssimenetelmän antamiin todennäköisyyksiin. Toteutin tämän kahdella testiskenaariosarjalla, joista toinen oli sama kuin laskenta-aikavertailussa, ja toisessa varioin ainoastaan Punaisen aloitusvahvuutta saadakseni paremmin esiin voittotodennäköisyyden muutoksen virheeseen. Käytin samoja parametreja ampumiseen kuin laskenta-aikojen vertailussakin, eli $p_B = p_r = 0.02$ ja $\lambda_B = \lambda_R = 2$. Joukkojen aloitusvahvuudet muuttivat ajokertojen välillä ensimmäisessä testisarjassa siten, että $B_0 = n*5$ ja $R_0 = n*4$, kun $n = 1, 2, \ldots, 20$, eli B_0 vaihteli viidestä sataan ja R_0 neljästä kahdeksaankymmeneen. Joukkojen pienimmät toimintavahvuudet olivat taas puolet aloitusvahvuudesta. Toisessa sarjassa ainoastaan Punaisen aloitusvahvuus vaihtui ajokertojen välillä siten, että $R_0 = n*2$, kun $n = 1, 2, \ldots, 20$ eli aloitusvahvuus vaihteli kahden ja neljänkymmenen välillä. Pienimmät toimintavahvuudet olivat jälleen puolet aloitusvahvuudesta. Kaikilla menetelmillä käytettiin kaikissa laskennoissa 200 aika-askelta.

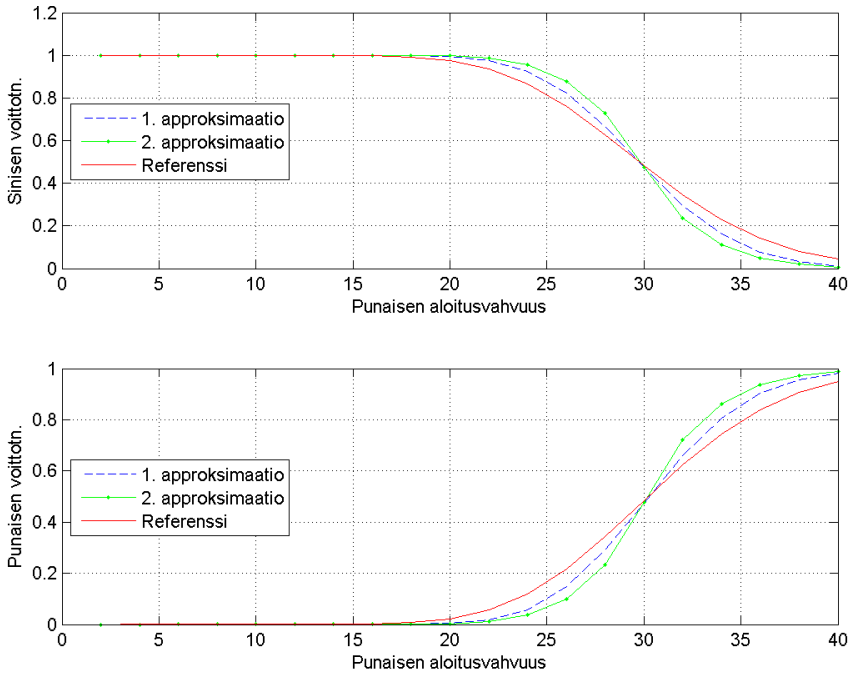
Kuvassa 3 näkyy ensimmäisen testisarjan tulokset. Molemmat approksimatiiviset menetelmät suosivat todennäköisempää voittajaa, mutta ensimmäisen menetelmän virhe on pienillä joukoilla vain noin kaksi kolmasosaa toisen menetelmän virheestä. Isoilla aloitusvahvuuksilla erot tasoittuvat huomattavasti ja molemmat menetelmät tuottavat käytännössä saman ennusteen. Kuvasta 4 nähdään paremmin, kuinka virheiden suuruudet kehittyvät voittotodennäköisyyden mukana. Virheet ovat pienimmillään, kun referenssimenetelmän mukainen voittotodennäköisyys kertoo, että toinen joukko voittaa käytännössä varmasti, tai molempien joukkojen ollessa hyvin tasaväkisiä. Kummankin approksimatiivisen menetelmän virheet ovat aina samaan suuntaan eli voittajasuosikin hyväksi, mutta ensimmäisen menetelmän virhe on käytännössä aina pienempi.

3.3 Lopputilan vahvuusjakaumien virheet

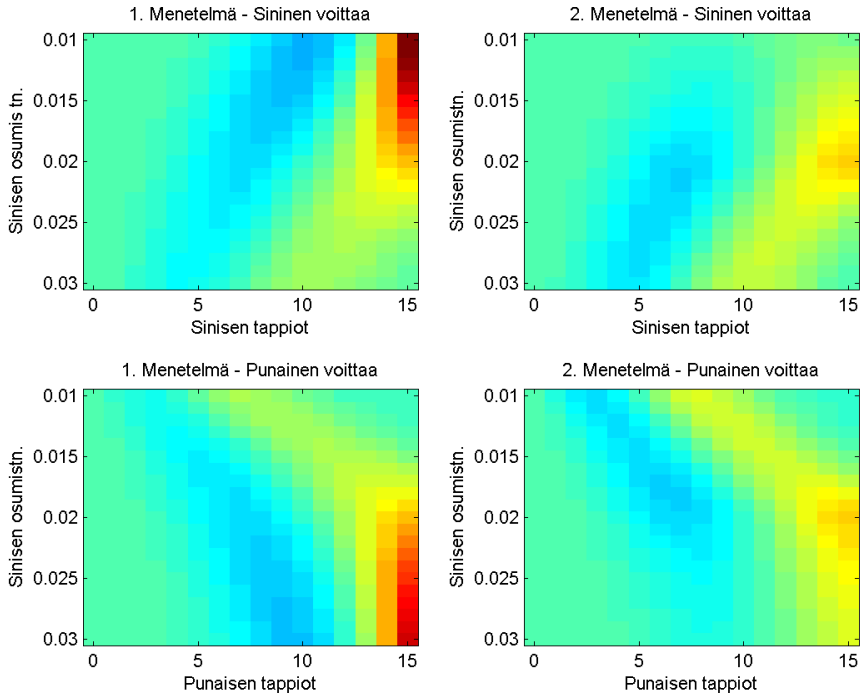
Vahvuusjakaumien virheen vertailun tein hieman eri tavoin kuin aiemmat vertailut tulosten visualisoinnin helpottamiseksi. Toteutin taas kaksikymmentä eri testiskenaariota, joissa tällä kertaa muuttui aloitusvahvuuksien sijaan Sinisen osumatodennäköisyys p_B 0.01:stä 0.03:een. Punaisen osuma-



Kuva 3: Erot voittotodennäköisyyksissä, kun molempien yksiköiden aloitusvahvuudet muuttuvat samassa suhteessa



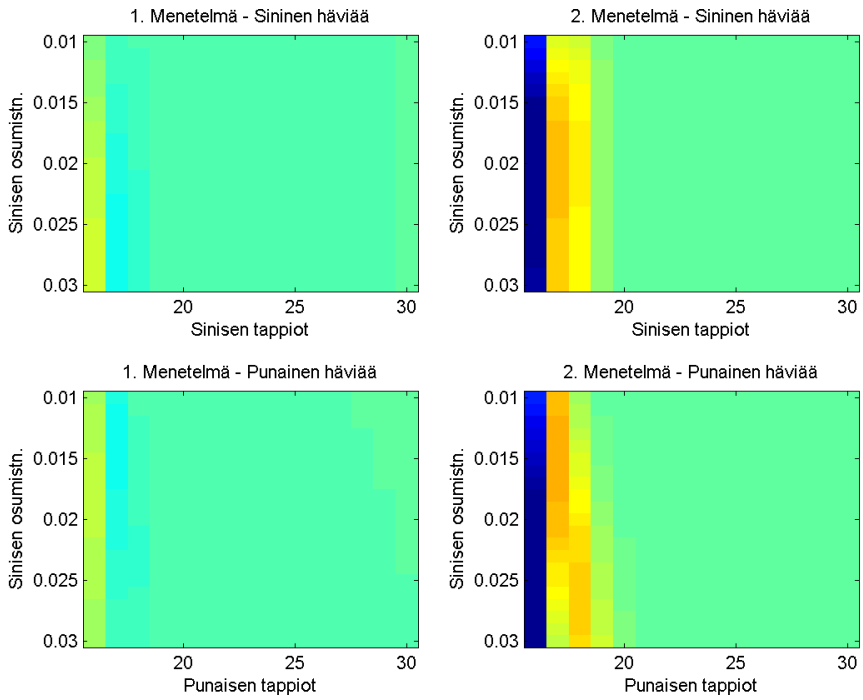
Kuva 4: Erot voittotodennäköisyyksissä, kun vain punaisen joukon aloitusvahvuus muuttuu



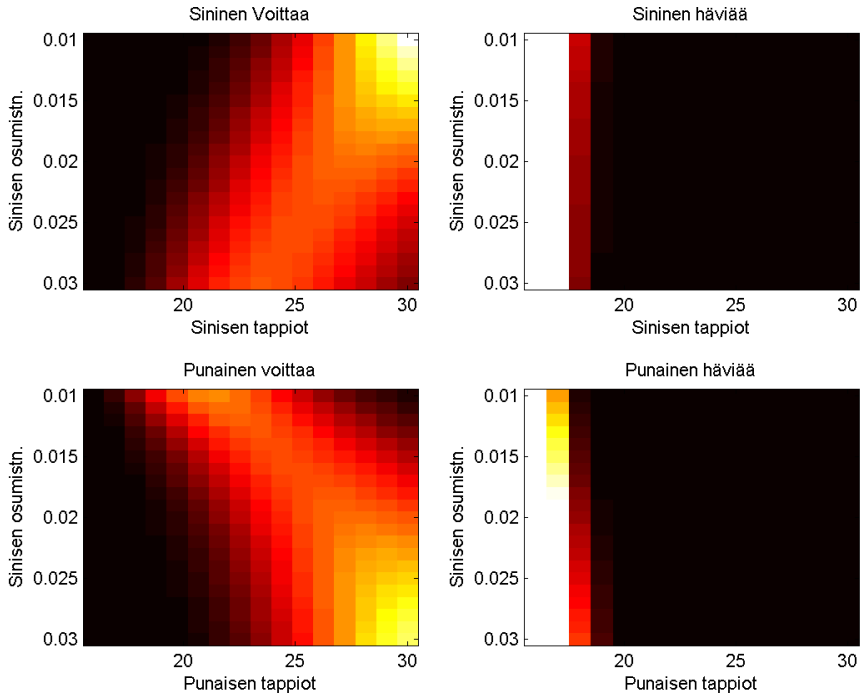
Kuva 5: Vahvuusjakaumien absoluuttiset virheet prosenttiyksiköittäin - Tummin väri vastaa 10 prosenttiyksikköä, siniset sävyt ovat negatiivisia arvoja ja punaiset positiivisia

todennäiköisyys oli vakio $p_R = 0.2$. Molempien joukkojen aloitusvahvuudet pysyivät vakioina $B_0 = R_0 = 30$ ja molempien pienimmät toimintavahvuudet olivat edelleen puolet vahvuudesta. Tämä mahdollisti sen, että molempien yksiköiden kaikissa lopputilajakaumissa oli yhtä monta alkiota, mutta voittotodennäköisyyttä saatiin silti varioitua.

Kuvassa 5 on esitetty yksiköiden tappiojakaumien virheet tilanteessa, jossa ne ovat voittaneet taistelun. Ylhäällä on esitetty Sinisen joukon tappioiden virheet ja alhaalla Punaisen. Vasemmalla olevat jakaumat on saatu laskettu ensimmäisellä menetelmällä ja oikealla olevat toisella menetelmällä. Tulosten perusteella toinen menetelmä selviytyy voittavan yksikön jakauman ennustamisessa paremmin kuin ensimmäinen, eli tilanne on päinvastainen, kuin kaksintaistelun voittajaa ennustettaessa. Häviötilanteessa ensimmäisen jakauman virheet ovat kuitenkin pienempiä kuten voidaan nähdä kuvasta 6. Vertailun vuoksi referenssimenetelmän tuottamat jakaumat on esitetty kuvassa



Kuva 6: Vahvuusjakaumien absoluuttiset virheet prosenttiyksiköittäin - Tummin väri vastaa 10 prosenttiyksikköä, siniset sävyt ovat negatiivisia arvoja ja punaiset positiivisia



Kuva 7: Joukkojen tappiojakaumat eri tilanteissa - Valkea väri tarkoittaa yli 20 prosenttiyksikköä ja musta nollaa

7. Kuva auttaa ymmärtämään paremmin virhejakaumien muotoja. Tappiojakaumat häviötilanteessa eivät ole erityisen mielenkiintoisia, koska harvoin yksikkö kuitenkaan menettää näillä aloitusvahvuuksilla paljoa yli yhtä miestä laskentakierroksessa. Seurauksena on häviötilanteen vahvuusjakauma on keskittynyt erittäin voimakkaasti juuri pienimmän toimintavahvuuden alapuolelle.

4 Yhteenveto

Kumpikaan tutkituista approksimatiivivisista menetelmistä ei osoittautunut yksiselitteisesti toista paremmaksi. Ensimmäinen menetelmä oli selkeästi tarkempi ennustamaan kaksintaistelun voittajan, mutta toinen menetelmä osoittautui hieman nopeammaksi ja onnistui laskemaan voittavan joukon vahvuusjakauman tarkemmin. Kummankaan menetelmän virheet eivät kuitenkaan olleet valtaosassa tapauksista niin suuria, etteikö tuloksia voisi käyttää

hyväkseen joukkojen välisten vahvuuksien vertailuissa.
Jos mietitään menetelmien sovellettavuutta osana laajempaa kokonaisuutta kuin ainoastaan kahden jalkaväkijoukon välisen taistelun tulosten ennustamisessa, niin toiselle menetelmälle on selkeästi eduksi, että se käyttää laskennassa joukon koko reunajakaumaa pelkän ehdollisen jakauman sijaan. Tämä on edullista siitä syystä, että todellisella taistelukentällä yksikköön voisi tulla asevaikutusta myös muualta kuin sitä vastustavasta yksiköstä. Toinen syy, miksi koko reunajakauman käyttäminen voi olla edullista menetelmää sovellettaessa muualla, on se, että esimerkiksi Sandis operaatioanalyysityökalu ([Lappi, 2008]) käyttää vastaavanlaisia jakaumia joukkojen vahvuuksien tarkasteluun.
Vaikka tässä työssä käytettiin menetelmien menetelmien vertailuun kahta jalkaväkijoukkoa, joilla oli käytössään ainoastaan kiväärin kaltaisia aseita, eivät menetelmien oletukset varsinaisesti vaadi, että joukkojen aseistukset eivät voisi olla monimutkaisempiakin. Andreas Åkesson onkin diplomityössään ([Åkesson, 2012]) toteuttanut tällaisen laajennuksen Lapin, Pakkasen ja Åkessonin menetelmään ([Lappi et al., 2012a]), joka siis on tämän työn vertailtavista approksimatiivisista menetelmistä ensimmäinen. Vastaavan laajennuksen tekeminen toiselle vertailussa olleelle menetelmälle ei ole yhtään sen vaikeampaa. Ei myöskään olemassa mitään syytä, miksi kumpi vain tarkasteltu menetelmä ei voisi saada joukoille koituvia tappioita jostain ulkopuolelta, joten menetelmiä voisi käyttää vain kaksintaistelun voittajan ja vahvuuksien laskentaan ja antaa esimerkiksi Sandiksen laskea joukoille koituvat tappiot.

Viitteet

Mika Hyytiäinen. *Paikkatietoylivoima digitaalisella taistelukentällä - Sotilaallisten maastoanalyysien metamalli*. PhD thesis, Maanpuolustuskorkeakoulu, 2003.

Keijo Jaakola. Asevaikutus ja suomalainen maasto komppanian taistelun mallintamisessa. Master's thesis, Teknillinen korkeakoulu, 2004.

Lauri Kangas. Taistelun stokastinen mallinnus. Master's thesis, Teknillinen korkeakoulu, 2005.

Yrjö Keinonen. *Jalkaväen tulen vaikutuksesta*. Puolustusvoimat, 1954.

Andreas Åkesson. Automatic calculation of win probabilities and conditional strength distribution of units in stochastic simulation model. Master's thesis, Åbo Akademi University, 2012.

Esa Lappi. Sandis military operation analysis tool. In *2nd Nordic Military Analysis Symposium*, pages 17–18, 2008.

Esa Lappi and Bernt Åkesson. Combat simulation as tool for evaluation of future weapon systems and some risks in sce. *SECURITY IN FUTURES–SECURITY IN CHANGE*, pages 217–224, 2011.

Esa Lappi and Olli Pottonen. Combat parameter estimation in sandis OA software. In *Lanchester and Beyond—A Workshop on Operational Analysis Methodology, edited by JS Hämäläinen*, volume 11, pages 31–38, 2006.

Esa Lappi, Mikko Pakkanen, and Bernt Åkesson. An approximative method of simulating a duel. In *Proceedings of the Winter Simulation Conference*, WSC '12, pages 208:1–208:10, Berlin, Germany, 2012a. Winter Simulation Conference. URL http://dl.acm.org/citation.cfm?id=2429759.2430038.

Esa Lappi et al. *Computational methods for tactical simulations*. PhD thesis, Maanpuolustuskorkeakoulu, 2012b.

Tim Lindholm and Frank Yellin. *Java virtual machine specification*. Addison-Wesley Longman Publishing Co., Inc., 1999.

Jari Sormunen, Lt Col, and Harri Eskelinen. Utilizing sensitivity analysis and data filtering processes to support overall optimizing and decision making models in military tactics. In *Society 40 th Anniversary Workshop–FORS40*, pages 86–93, 2013.

A Referenssimenetelmän javatoteutus

```
1  import java.io.PrintWriter;
2
3
4  public class ExactCalculator {
5    double tulinopeusB=1, tulinopeusR=1;
6    double phB=0.02, phR=0.02;
7    int nB=50,  nR=40;
8    int steps = 60;
9    int lB=25, lR=20;
10   int step = 0;
11
12   //public double[][][] R, B, IB, IR;
13   public double[] Pr;
14   public double[][][] Prm;
15
16   public double [] jatkuu, voittoR, voittoB, moltuhottu;
17   public double [] stR, stRl, stB, stBl;
18
19   double[][][] siirto;
20   double eapu1=1, eapu2=1;
21
22   public ExactCalculator (){
23     Pr=VNull((nB+1)*(nR+1));
24     Pr[0]=1;
25
26     jatkuu= VNull(steps+1);
27     voittoR= VNull(steps+1);
28     voittoB= VNull(steps+1);
29     moltuhottu= VNull(steps+1);
30     jatkuu [0]=1;
31
32     //luodaan ammunnan tilasiirtotensori
33     double[][][][][] siirto=MNull(nB+1,nR+1,nB+1,nR+1);
34     for(int iB=0;iB<=nB;iB++)
35       for(int iR=0;iR<=nR;iR++)
36         for(int iBo=0;iBo<=nB;iBo++)
37           for(int iRo=0;iRo<=nR;iRo++)
38             if (iRo>lR && iBo>lB) siirto[nB-iB][nR-iR][nB-iBo][nR
                 -iRo]=DBinom(iBo-iB,iBo,1-Math.pow(1-phR,
                  tulinopeusR*iRo/iBo))*DBinom(iRo-iR,iRo,1-Math.
                  pow(1-phB,tulinopeusB*iBo/iRo));
39             else
40               if(iBo==iB&&iRo==iR) siirto[nB-iB][nR-iR][nB-iBo][
                   nR-iRo]=1;
41               else siirto[nB-iB][nR-iR][nB-iBo][nR-iRo]=0;
```

```
42
43      //muunnetaan tilasiirtotensori matriisiksi
44      this.siirto=new double[(nB+1)*(nR+1)][(nB+1)*(nR+1)];
45      for(int iB=0;iB<=nB;iB++)
46        for(int iR=0;iR<=nR;iR++)
47          for(int iBo=0;iBo<=nB;iBo++)
48            for(int iRo=0;iRo<=nR;iRo++)
49              this.siirto[iB+(nB+1)*iR][iBo+(nB+1)*iRo]=siirto[iB][
                    iR][iBo][iRo];
50
51
52      //vektorit ehdollisia jakaumia varten
53      stR=VNull(nR-lR+1);
54      stB=VNull(nB-lB+1);
55      stRl=VNull(lR);
56      stBl=VNull(lB);
57    }
58
59    public ExactCalculator (int s, int nB, int nR, int lB, int lR,
        double phB, double phR, double tulinopeusB, double
        tulinopeusR){
60      this.phB=phB;
61      this.phR=phR;
62      this.tulinopeusB=tulinopeusB;
63      this.tulinopeusR=tulinopeusR;
64      steps=s;
65      this.nB=nB;
66      this.nR=nR;
67      this.lB=lB;
68      this.lR=lR;
69      Pr=VNull((nB+1)*(nR+1));
70      Pr[0]=1;
71
72      jatkuu= VNull(steps+1);
73      voittoR= VNull(steps+1);
74      voittoB= VNull(steps+1);
75      moltuhottu= VNull(steps+1);
76      jatkuu [0]=1;
77
78      //luodaan ammunnan tilasiirtotensori
79      double[][][][] siirto=MNull(nB+1,nR+1,nB+1,nR+1);
80      for(int iB=0;iB<=nB;iB++)
81        for(int iR=0;iR<=nR;iR++)
82          for(int iBo=0;iBo<=nB;iBo++)
83            for(int iRo=0;iRo<=nR;iRo++)
84              if (iRo>=lR && iBo>=lB) siirto[nB-iB][nR-iR][nB-iBo][
                    nR-iRo]=DBinom(iBo,iBo-iB,1-Math.pow(1-phR,
                    tulinopeusR*iRo/iBo))*DBinom(iRo,iRo-iR,1-Math.
```

```
                    pow(1-phB,tulinopeusB*iBo/iRo));
85                else
86                  if((iBo==iB) && (iRo==iR)) siirto[nB-iB][nR-iR][nB-
                        iBo][nR-iRo]=1;
87                  else siirto[nB-iB][nR-iR][nB-iBo][nR-iRo]=0;
88      //System.out.println(siirto[nB][nR][nB][nR]);
89      //muunnetaan tilasiirtotensori matriisiksi
90      this.siirto=new double[(nB+1)*(nR+1)][(nB+1)*(nR+1)];
91      for(int iB=0;iB<=nB;iB++)
92        for(int iR=0;iR<=nR;iR++)
93          for(int iBo=0;iBo<=nB;iBo++)
94            for(int iRo=0;iRo<=nR;iRo++)
95              this.siirto[iB+(nB+1)*iR][iBo+(nB+1)*iRo]=siirto[iB][
                    iR][iBo][iRo];
96
97      //MPrint(this.siirto);
98      //System.out.println();
99
100     //vektorit ehdollisia jakaumia varten
101     stR=VNull(nR-lR+1);
102     stB=VNull(nB-lB+1);
103     stRl=VNull(lR);
104     stBl=VNull(lB);
105   }
106
107   public void NextStep(){
108
109     Pr=MTimes(siirto, Pr);
110
111
112
113
114     //pivitet n voitto- ja muut todenn k isyydet
115     for(int i=0;i<=nB-lB;i++)
116       for(int j=0;j<=nR-lR;j++)
117         jatkuu[step+1]+=Pr[i+(nB+1)*j];
118     for(int i=0;i<=nB-lB;i++)
119       for(int j=nR-lR+1;j<=nR;j++)
120         voittoB[step+1]+=Pr[i+(nB+1)*j];
121     for(int i=nB-lB+1;i<=nB;i++)
122       for(int j=0;j<=nR-lR;j++)
123         voittoR[step+1]+=Pr[i+(nB+1)*j];
124     for(int i=nB-lB+1;i<=nB;i++)
125       for(int j=nR-lR+1;j<=nR;j++)
126         moltuhottu[step+1]+=Pr[i+(nB+1)*j];
127
128
```

```
129       step++;
130     }
131
132     public void ehdolliset(){
133       //tehd   n tulosjakaumasta matriisi
134       Prm=new double[nB+1][nR+1];
135       for(int i=0;i<=nB;i++)
136         for(int j=0;j<=nR;j++)
137           Prm[i][j]=Pr[i+j*(nB+1)];
138
139       double apu=0;
140       for(int i=0;i<=nB-lB;i++){
141         for(int j=nR-lR+1;j<=nR;j++){
142           apu+=Prm[i][j];
143           stB[i]+=Prm[i][j];
144         }
145       }
146       stB=MTimesC(stB, 1/apu);
147
148       apu=0;
149       for(int i=nB-lB+1;i<=nB;i++){
150         for(int j=0;j<=nR-lR;j++){
151           apu+=Prm[i][j];
152           stR[j]+=Prm[i][j];
153         }
154       }
155       stR=MTimesC(stR, 1/apu);
156
157       apu=0;
158       for(int i=nB-lB+1;i<=nB;i++){
159         for(int j=0;j<=nR;j++){
160           apu+=Prm[i][j];
161           stBl[i-nB+lB-1]+=Prm[i][j];
162         }
163       }
164       stBl=MTimesC(stBl, 1/apu);
165
166       apu=0;
167       for(int i=0;i<=nB;i++){
168         for(int j=nR-lR+1;j<=nR;j++){
169           apu+=Prm[i][j];
170           stRl[j-nR+lR-1]+=Prm[i][j];
171         }
172       }
173       stRl=MTimesC(stRl, 1/apu);
174
175
176
```

```
177   }
178
179
180
181   //luo ammuntamatriisin
182   public double[][] ammuntamatriisi(int n, double ph, double
          tulinopeus, int luovutus){
183     double[][] p;
184     if(tulinopeus==0||n<luovutus)return MDiag(n+1,1);
185     else{
186       p=MNull(n+1,n+1);
187       for(int maaleja=n;maaleja>=0;maaleja--){
188         double pk;
189         if(maaleja>0)pk=1-Math.pow((1-ph),tulinopeus/maaleja);
190         else {pk=0;}
191         for(int i=0;i<=maaleja;i++)
192           if(maaleja>=luovutus)p[n-maaleja+i][n-maaleja]=DBinom(
                maaleja,i,pk);
193           else p[n-maaleja+i][n-maaleja]=DBinom(maaleja,i,0);
194       }
195       return p;
196     }
197
198
199   }
200
201   //palauttaa binomijakauman tiheysfunktion arvon arvoille n, k,
        p
202   public double DBinom(int n, int k, double p){
203     if(k>n)return 0;
204     double taikakerroin=1;
205     for(int i=k+1;i<=n;i++)taikakerroin*=i;
206     for(int i=1;i<=n-k;i++)taikakerroin/=i;
207     if(p==1||p==0){
208       if(k==0||k==n)return 1;
209       else return 0;
210     }
211     return taikakerroin*Math.pow(p, k)*Math.pow((1-p),(n-k));
212   }
213
214
215   //Palauttaa taulukon joka on t ynn  nollia
216   public static double[][] MNull(int dim1,int dim2){
217     double[][] M=new double[dim1][dim2];
218     for(int i=0;i<dim1;i++)for(int j=0;j<dim2;j++)M[i][j]=0;
219     return M;
220   }
221
```

```
222   //Palauttaa taulukon joka on t ynn  nollia
223   public double[] VNull(int dim1){
224     double[] M=new double[dim1];
225     for(int i=0;i<dim1;i++)M[i]=0;
226     return M;
227   }
228
229   //Palauttaa taulukon joka on t ynn  nollia
230   public double[][][][] MNull(int dim1,int dim2, int dim3){
231     double[][][][] M=new double[dim1][dim2][dim3];
232     for(int i=0;i<dim1;i++)for(int j=0;j<dim2;j++)for(int k=0;k<
          dim3;k++)M[i][j][k]=0;
233     return M;
234   }
235
236   //Palauttaa taulukon joka on t ynn  nollia
237   public double[][][][][][] MNull(int dim1,int dim2, int dim3, int
        dim4){
238     double[][][][][][] M=new double[dim1][dim2][dim3][dim4];
239     for(int i=0;i<dim1;i++)for(int j=0;j<dim2;j++)for(int k=0;k<
          dim3;k++)for(int l=0;l<dim4;l++)M[i][j][k][l]=0;
240     return M;
241   }
242
243   //Palauttaa dim x dim taulukon jonka diagonaalilla on vakiota C
244   public double[][] MDiag(int dim,double C){
245     double[][] M=MNull(dim,dim);
246     for(int i=0;i<dim;i++){
247       M[i][i]=C;
248     }
249     return M;
250   }
251
252   //kertoo kesken  n matriisit A ja B
253   public double[][] MTimes(double[][] A,double[][] B){
254     if(A[0].length!=B.length){System.err.println("paskan␣
          matriisin␣annoit");return null;}
255     double[][] C=new double[A.length][B[0].length];
256     for(int i=0;i<C.length;i++)for(int j=0;j<C[0].length;j++){
257       C[i][j]=0;
258       for(int k=0;k<B.length;k++)C[i][j]+=A[i][k]*B[k][j];
259     }
260     return C;
261   }
262
263   //kertoo kesken  n vaakavektorin A ja matriisin B
264   public double[] MTimes(double[] A,double[][] B){
265     if(A.length!=B.length){System.err.println("paskan␣matriisin␣
          annoit");return null;}
```

```
266     double[] C=new double[B[0].length];
267     for(int i=0;i<C.length;i++){
268       C[i]=0;
269       for(int k=0;k<B.length;k++)C[i]+=A[k]*B[k][i];
270     }
271     return C;
272   }
273
274   //kertoo kesken  n matriisin A ja psytyvektorin B
275   public double[] MTimes(double[][] A,double[] B){
276     if(A[0].length!=B.length){System.err.println("paskan␣
          matriisin␣annoit");return null;}
277     double[] C=new double[A.length];
278     for(int i=0;i<C.length;i++){
279       C[i]=0;
280       for(int k=0;k<B.length;k++)C[i]+=A[i][k]*B[k];
281     }
282     return C;
283   }
284
285
286
287   //palauttaa matriisin A transpoosin
288   public double[][] MTranspose(double[][] A){
289     double [][] C = new double[A[0].length][A.length];
290     for(int i=0;i<A.length;i++)for(int j=0;j<A[0].length;j++)C[j
          ][i]=A[i][j];
291     return C;
292   }
293
294   //kertoo matriisin A vakiolla b
295   public double[][] MTimesC(double[][] A,double b){
296     double[][] C=new double[A.length][A[0].length];
297     for(int i=0;i<C.length;i++)for(int j=0;j<C[0].length;j++){
298       C[i][j]=A[i][j]*b;
299     }
300     return C;
301   }
302
303   //kertoo vektorin A vakiolla b
304   public double[] MTimesC(double[] A,double b){
305     double[] C=new double[A.length];
306     for(int i=0;i<C.length;i++){
307       C[i]=A[i]*b;
308     }
309     return C;
310   }
311
312   //laskee yhteen matriisit A ja B
```

```
313   public double[][] MPlus(double[][] A,double[][] B){
314     if(A.length!=B.length||A[0].length!=B[0].length)return null;
315     double[][] C=new double[A.length][B[0].length];
316     for(int i=0;i<C.length;i++)for(int j=0;j<C[0].length;j++){
317       C[i][j]=A[i][j]+B[i][j];
318     }
319     return C;
320   }
321
322   //tulostaa matriisin debuggaustarkoituksiin
323   public void MPrint(double[][] M){
324     for(int i=0;i<M.length;i++){
325       for(int j=0;j<M[0].length;j++)System.out.print(M[i][j]+"␣")
              ;
326       System.out.println();
327     }
328
329   }
330
331   //tulostaa matriisin debuggaustarkoituksiin
332   public void MPrint(double[] M){
333     for(int i=0;i<M.length;i++){
334       System.out.print(M[i]+"␣");
335
336     }
337     System.out.println();
338   }
339
340   //tulostaa matriisin debuggaustarkoituksiin
341   public void MPrint(double[] M, PrintWriter p){
342     for(int i=0;i<M.length;i++){
343       p.print(M[i]+"␣");
344
345     }
346   }
347 }
```

B Lapin, Pakkasen ja Åkessonin menetelemän javatoteutus

```
1  import java.io.PrintWriter;
2
3
4  public class DuelCalculator {
5    double tulinopeusB=1, tulinopeusR=1;
6    double phB=0.02, phR=0.02;
7    int nB=50,  nR=40;
8    int steps = 60;
9    int lB=25, lR=20;
10   int step = 0;
11
12   public double[][] B, R;
13
14   public double [] jatkuu, voittoR, voittoB, moltuhottu;
15   public double [] stR, stRl, stB, stBl;
16
17   double[][][][] AB = new double[nR+1][][][], AR=new double[nB
         +1][][][];
18
19   public DuelCalculator (){
20     B=MNull(steps+1,nB+1); R=MNull( steps+1,nR+1);
21     B[0][0]=R[0][0]=1;
22
23     jatkuu= VNull(steps+1);
24     voittoR= VNull(steps+1);
25     voittoB= VNull(steps+1);
26     moltuhottu= VNull(steps+1);
27     jatkuu [0]=1;
28
29     //lasketaan ammuntamatriisit valmiiksi
30     for(int i=nR-1;i>=0;i--)
31       AB[i]=ammuntamatriisi(nB,phR,tulinopeusR*i);
32     for(int i=nB-1;i>=0;i--)
33       AR[i]=ammuntamatriisi(nR,phB,tulinopeusB*i);
34
35     //vektorit ehdollisia jakaumia varten
36     stR=VNull(nR-lR+1);
37     stB=VNull(nB-lB+1);
38     stRl=VNull(lR);
39     stBl=VNull(lB);
40   }
41
```

```
42  public DuelCalculator (int s, int nB, int nR, int lB, int lR,
        double phB, double phR, double tulinopeusB, double
        tulinopeusR){
43    this.phB=phB;
44    this.phR=phR;
45    this.tulinopeusB=tulinopeusB;
46    this.tulinopeusR=tulinopeusR;
47    steps=s;
48    this.nB=nB;
49    this.nR=nR;
50    this.lB=lB;
51    this.lR=lR;
52    B=MNull(steps+1,nB+1); R=MNull( steps+1,nR+1);
53    B[0][0]=R[0][0]=1;
54
55    jatkuu= VNull(steps+1);
56    voittoR= VNull(steps+1);
57    voittoB= VNull(steps+1);
58    moltuhottu= VNull(steps+1);
59    jatkuu [0]=1;
60
61    //lasketaan ammuntamatriisit valmiiksi
62    AB=new double[nR+1][][];
63    AR=new double[nB+1][][];
64    for(int i=nR;i>=0;i--)
65      AB[i]=ammuntamatriisi(nB,phR,tulinopeusR*i);
66    for(int i=nB;i>=0;i--)
67      AR[i]=ammuntamatriisi(nR,phB,tulinopeusB*i);
68
69    //vektorit ehdollisia jakaumia varten
70    stR=VNull(nR-lR+1);
71    stB=VNull(nB-lB+1);
72    stRl=VNull(lR);
73    stBl=VNull(lB);
74  }
75
76  public void NextStep(){
77    //lasketaan sinisen siirtym matriisi
78    double[][] PB=MNull(nB+1,nB+1);
79    for(int i=0;i<nR;i++)PB=MPlus(MTimesC(AB[i],R[step][nR-i-1])
          , PB);
80
81    //lasketaan sinisen uusi jakauma
82    B[step+1]=MTimes(B[step],MTranspose(PB));
83
84    //lasketaan punaisen siirtym matriisi
85    double[][] PR=MNull(nR+1,nR+1);
```

```
86      for(int i=0;i<nB;i++)PR=MPlus(MTimesC(AR[i],B[step][nB-i-1]),PR);
87
88      //lasketaan punaisen uusi jakauma
89      //System.out.println(PR.length+" "+PR[0].length+" "+R[step].length);
90      R[step+1]=MTimes(R[step],MTranspose(PR));
91      //MPrint((PR));
92
93      //p ivitet n tn, ett taistelu jatkuu
94      double apu1=0, apu2=0;
95      for(int i=nR-lR+1;i<=nR;i++)apu1+=R[step+1][i];
96      for(int i=nB-lB+1;i<=nB;i++)apu2+=B[step+1][i];
97      jatkuu[step+1]=(1-apu1)*(1-apu2)*jatkuu[step];
98
99      //p ivitet n tn, ett punainen on voittanut
100     voittoR[step+1]=(1-apu1)*apu2*jatkuu[step]+voittoR[step];
101
102     //p ivitet n tn, ett sininen on voittanut
103     voittoB[step+1]=apu1*(1-apu2)*jatkuu[step]+voittoB[step];
104
105     //p ivitet n tn, ett molemmat ovat ly tyj
106     moltuhottu[step+1]=apu1*apu2*jatkuu[step]+moltuhottu[step];
107
108     //p ivitet n ehdolliset jakaumat
109     for(int i=0;i<=nR-lR;i++)stR[i]=(1/voittoR[step+1])*(R[step
            +1][i]*apu2*jatkuu[step]+stR[i]*voittoR[step]);
110     for(int i=0;i<=nB-lB;i++)stB[i]=(1/voittoB[step+1])*(B[step
            +1][i]*apu1*jatkuu[step]+stB[i]*voittoB[step]);
111     for(int i=0;i<lR;i++)stRl[i]=(1/(voittoB[step+1]+moltuhottu[
            step+1]))*(R[step+1][nR-lR+1+i]*jatkuu[step]+stRl[i]*(
            voittoB[step]+moltuhottu[step]));
112     for(int i=0;i<lB;i++)stBl[i]=(1/(voittoR[step+1]+moltuhottu[
            step+1]))*(B[step+1][nB-lB+1+i]*jatkuu[step]+stBl[i]*(
            voittoR[step]+moltuhottu[step]));
113
114
115     //asetetaan jakaumissa ly ty vastaavien tilojen tn:t
            nolliksi
116     for(int i=nB-lB+1;i<=nB;i++)B[step+1][i]=0;
117     for(int i=nR-lR+1;i<=nR;i++)R[step+1][i]=0;
118
119     //Renormalisoidaan saadut vektorit jakaumiksi
120     double apu=0;
121     for(int i=0;i<=nB;i++)apu+=B[step+1][i];
122     B[step+1]=MTimesC(B[step+1],1/apu);
123     apu=0;
```

```
124       for(int i=0;i<=nR;i++)apu+=R[step+1][i];
125       R[step+1]=MTimesC(R[step+1],1/apu);
126       //MPrint(R[step+1]);
127
128       step++;
129     }
130
131     //luo ammuntamatriisin
132     public double[][] ammuntamatriisi(int n, double ph, double
          tulinopeus){
133       double[][] p;
134       if(tulinopeus==0)return MDiag(n+1,1);
135       else{
136         p=MNull(n+1,n+1);
137         for(int maaleja=n;maaleja>=0;maaleja--){
138           double pk=1-Math.pow((1-ph),tulinopeus/maaleja);
139           for(int i=0;i<=maaleja;i++)p[n-maaleja+i][n-maaleja]=
                DBinom(maaleja,i,pk);
140         }
141         return p;
142       }
143
144     }
145
146     //palauttaa binomijakauman tiheysfunktion arvon arvoille n, k,
          p
147     public double DBinom(int n, int k, double p){
148       if(k>n)return 0;
149       double taikakerroin=1;
150       for(int i=k+1;i<=n;i++)taikakerroin*=i;
151       for(int i=1;i<=n-k;i++)taikakerroin/=i;
152       if(p==1||p==0){
153         if(k==n)return 1;
154         else return 0;
155       }
156       return taikakerroin*Math.pow(p, k)*Math.pow((1-p),(n-k));
157     }
158
159
160     //Palauttaa taulukon joka on t ynn  nollia
161     public static double[][] MNull(int dim1,int dim2){
162       double[][] M=new double[dim1][dim2];
163       for(int i=0;i<dim1;i++)for(int j=0;j<dim2;j++)M[i][j]=0;
164       return M;
165     }
166
167     //Palauttaa taulukon joka on t ynn  nollia
168     public double[] VNull(int dim1){
```

```
169     double[] M=new double[dim1];
170     for(int i=0;i<dim1;i++)M[i]=0;
171     return M;
172   }
173
174   //Palauttaa taulukon joka on t ynn  nollia
175   public double[][][][] MNull(int dim1,int dim2, int dim3){
176     double[][][][] M=new double[dim1][dim2][dim3];
177     for(int i=0;i<dim1;i++)for(int j=0;j<dim2;j++)for(int k=0;k<
            dim3;k++)M[i][j][k]=0;
178     return M;
179   }
180
181   //Palauttaa dim x dim taulukon jonka diagonaalilla on vakiota C
182   public double[][] MDiag(int dim,double C){
183     double[][] M=MNull(dim,dim);
184     for(int i=0;i<dim;i++){
185       M[i][i]=C;
186     }
187     return M;
188   }
189
190   //kertoo kesken  n matriisit A ja B
191   public double[][] MTimes(double[][] A,double[][] B){
192     if(A[0].length!=B.length){System.err.println("paskan␣
            matriisin␣annoit");return null;}
193     double[][] C=new double[A.length][B[0].length];
194     for(int i=0;i<C.length;i++)for(int j=0;j<C[0].length;j++){
195       C[i][j]=0;
196       for(int k=0;k<B.length;k++)C[i][j]+=A[i][k]*B[k][j];
197     }
198     return C;
199   }
200
201   //kertoo kesken  n vaakavektorin A ja matriisin B
202   public double[] MTimes(double[] A,double[][] B){
203     if(A.length!=B.length){System.err.println("paskan␣matriisin␣
            annoit");return null;}
204     double[] C=new double[B[0].length];
205     for(int i=0;i<C.length;i++){
206       C[i]=0;
207       for(int k=0;k<B.length;k++)C[i]+=A[k]*B[k][i];
208     }
209     return C;
210   }
211
212   //kertoo kesken  n matriisin A ja psytyvektorin B
213   public double[] MTimes(double[][] A,double[] B){
```

```
214     if(A[0].length!=B.length){System.err.println("paskan␣
            matriisin␣annoit");return null;}
215     double[] C=new double[A.length];
216     for(int i=0;i<C.length;i++){
217       C[i]=0;
218       for(int k=0;k<B.length;k++)C[i]+=A[i][k]*B[k];
219     }
220     return C;
221   }
222
223
224
225   //palauttaa matriisin A transpoosin
226   public double[][] MTranspose(double[][] A){
227     double [][] C = new double[A[0].length][A.length];
228     for(int i=0;i<A.length;i++)for(int j=0;j<A[0].length;j++)C[j
            ][i]=A[i][j];
229     return C;
230   }
231
232   //kertoo matriisin A vakiolla b
233   public double[][] MTimesC(double[][] A,double b){
234     double[][] C=new double[A.length][A[0].length];
235     for(int i=0;i<C.length;i++)for(int j=0;j<C[0].length;j++){
236       C[i][j]=A[i][j]*b;
237     }
238     return C;
239   }
240
241   //kertoo vektorin A vakiolla b
242   public double[] MTimesC(double[] A,double b){
243     double[] C=new double[A.length];
244     for(int i=0;i<C.length;i++){
245       C[i]=A[i]*b;
246     }
247     return C;
248   }
249
250   //laskee yhteen matriisit A ja B
251   public double[][] MPlus(double[][] A,double[][] B){
252     if(A.length!=B.length||A[0].length!=B[0].length)return null;
253     double[][] C=new double[A.length][B[0].length];
254     for(int i=0;i<C.length;i++)for(int j=0;j<C[0].length;j++){
255       C[i][j]=A[i][j]+B[i][j];
256     }
257     return C;
258   }
259
260   //tulostaa matriisin debuggaustarkoituksiin
```

```
261   public void MPrint(double[][] M){
262     for(int i=0;i<M.length;i++){
263       for(int j=0;j<M[0].length;j++)System.out.print(M[i][j]+" ")
              ;
264       System.out.println();
265     }
266
267   }
268
269   //tulostaa matriisin debuggaustarkoituksiin
270   public void MPrint(double[] M){
271     for(int i=0;i<M.length;i++){
272       System.out.print(M[i]+" ");
273
274     }
275     System.out.println();
276   }
277
278   //tulostaa matriisin debuggaustarkoituksiin
279   public void MPrint(double[] M, PrintWriter p){
280     for(int i=0;i<M.length;i++){
281       p.print(M[i]+" ");
282
283     }
284   }
285 }
```

C Oman menetelmäni javatoteutus

```
1  import java.io.PrintWriter;
2
3
4  public class RefCalculator {
5    double tulinopeusB=1, tulinopeusR=1;
6    double phB=0.02, phR=0.02;
7    int nB=50,  nR=40;
8    int steps = 60;
9    int lB=25, lR=20;
10   int step = 0;
11
12   public double[][] B, R, IB, IR;
13
14   public double [] jatkuu, voittoR, voittoB, moltuhottu;
15   public double [] stR, stRl, stB, stBl;
16
17   double[][][][] AB = new double[nR+1][][][], AR=new double[nB
         +1][][][];
18   double eapu1=1, eapu2=1;
19
20   public RefCalculator (){
21     B=MNull(steps+1,nB+1); R=MNull( steps+1,nR+1);
22     B[0][0]=R[0][0]=1;
23
24     jatkuu= VNull(steps+1);
25     voittoR= VNull(steps+1);
26     voittoB= VNull(steps+1);
27     moltuhottu= VNull(steps+1);
28     jatkuu [0]=1;
29
30     //lasketaan ammuntamatriisit valmiiksi
31     for(int i=nR-1;i>=0;i--)
32       AB[i]=ammuntamatriisi(nB,phR,tulinopeusR*i,lB);
33     for(int i=nB-1;i>=0;i--)
34       AR[i]=ammuntamatriisi(nR,phB,tulinopeusB*i,lR);
35     IB=MDiag(nB+1, 1);
36     IR=MDiag(nR+1, 1);
37
38
39     //vektorit ehdollisia jakaumia varten
40     stR=VNull(nR-lR+1);
41     stB=VNull(nB-lB+1);
42     stRl=VNull(lR);
43     stBl=VNull(lB);
44   }
```

```
45
46    public RefCalculator (int s, int nB, int nR, int lB, int lR,
          double phB, double phR, double tulinopeusB, double
          tulinopeusR){
47      this.phB=phB;
48      this.phR=phR;
49      this.tulinopeusB=tulinopeusB;
50      this.tulinopeusR=tulinopeusR;
51      steps=s;
52      this.nB=nB;
53      this.nR=nR;
54      this.lB=lB;
55      this.lR=lR;
56      B=MNull(steps+1,nB+1); R=MNull( steps+1,nR+1);
57      B[0][0]=1;
58      R[0][0]=1;
59
60      jatkuu= VNull(steps+1);
61      voittoR= VNull(steps+1);
62      voittoB= VNull(steps+1);
63      moltuhottu= VNull(steps+1);
64      jatkuu [0]=1;
65
66      //lasketaan ammuntamatriisit valmiiksi
67      AB=new double[nR+1][][];
68      AR=new double[nB+1][][];
69      for(int i=nR;i>=0;i--)
70        AB[i]=ammuntamatriisi(nB,phR,tulinopeusR*i,lB);
71      for(int i=nB;i>=0;i--)
72        AR[i]=ammuntamatriisi(nR,phB,tulinopeusB*i,lR);
73      //MPrint(AR[30]);
74      IB=MDiag(nB+1, 1);
75      IR=MDiag(nR+1, 1);
76
77      //vektorit ehdollisia jakaumia varten
78      stR=VNull(nR-lR+1);
79      stB=VNull(nB-lB+1);
80      stRl=VNull(lR);
81      stBl=VNull(lB);
82    }
83
84    public void NextStep(){
85      //lasketaan sinisen siirtym matriisi
86      double[][] PB=MNull(nB+1,nB+1);
87      for(int i=lR;i<=nR;i++)PB=MPlus(MTimesC(AB[nR],R[step][nR-i]
            ), PB);
88      for(int i=0;i<lR;i++)PB=MPlus(MTimesC(IB,R[step][nR-i] ), PB)
            ;
```

```
89
90
91      //lasketaan sinisen uusi jakauma
92      B[step+1]=MTimes(B[step],MTranspose(PB));
93
94      //lasketaan punaisen siirtym matriisi
95      double[][] PR=MNull(nR+1,nR+1);
96      for(int i=lB;i<=nB;i++)PR=MPlus(MTimesC(AR[nB],B[step][nB-i]
            ), PR);
97      for(int i=0;i<lB;i++)PR=MPlus(MTimesC(IR,B[step][nB-i] ), PR)
            ;
98      //MPrint(PR);
99      if(eapu1<=1e-5)
100       PR=IR;
101     if(eapu2<=1e-5)
102       PB=IB;
103
104     //lasketaan punaisen uusi jakauma
105     //System.out.println(PR.length+" "+PR[0].length+" "+R[step].
            length);
106     R[step+1]=MTimes(R[step],MTranspose(PR));
107
108     //p ivitet n tn, ett taistelu jatkuu
109     double apu1 =0, apu2=0;
110     for(int i=nR-lR+1;i<=nR;i++)apu1+=(R[step+1][i]-R[step][i]);
111     for(int i=nB-lB+1;i<=nB;i++)apu2+=(B[step+1][i]-B[step][i]);
112     if(apu1<0){
113       if(apu1>-1e-6)apu1=0;
114       else System.err.println("maailma r j ht apu1");
115     }
116     if(apu2<0){
117       if(apu2>-1e-6)apu1=0;
118       else System.err.println("maailma r j ht apu2");
119     }
120     //System.out.println(apu2);
121     double krapu1=eapu1-apu1;
122     double krapu2=eapu2-apu2;
123     apu1/=eapu1;
124     apu2/=eapu2;
125
126     if(apu1>1){
127       if(apu1<1+1e-6)apu1=1;
128       else System.err.println("maailma r j ht apu1");
129     }
130     if(apu2>1){
131       if(apu2>1+1e-6)apu1=1;
132       else System.err.println("maailma r j ht apu2");
```

```
133         }
134         //System.out.println(apu1);
135         //System.out.println(apu2);
136
137         jatkuu[step+1]=(1-apu1)*(1-apu2)*jatkuu[step];
138
139         //p  ivitet   n  voittotodenn k isyydet
140
141         moltuhottu[step+1]=apu1*apu2*jatkuu[step]+moltuhottu[step];
142         voittoR[step+1]=(1-apu1)*apu2*jatkuu[step]+voittoR[step];
143         voittoB[step+1]=(1-apu2)*apu1*jatkuu[step]+voittoB[step];
144
145         //p  ivitet   n  ehdolliset jakaumat
146         if(eapu1>0)for(int i=0;i<=nR-lR;i++)stR[i]=(1/voittoR[step
                +1])*((R[step+1][i])*apu2/eapu1*jatkuu[step]+stR[i]*
                voittoR[step]);
147         if(eapu2>0)for(int i=0;i<=nB-lB;i++)stB[i]=(1/voittoB[step
                +1])*((B[step+1][i])*apu1/eapu2*jatkuu[step]+stB[i]*
                voittoB[step]);
148         if(eapu1>0&&eapu2>0)for(int i=0;i<lR;i++)stRl[i]=(1/(voittoB[
                step+1]+moltuhottu[step+1]))*((R[step+1][nR-lR+1+i]-R[
                step][nR-lR+1+i])/eapu1*jatkuu[step]+stRl[i]*(voittoB[
                step]+moltuhottu[step]));
149         if(eapu2>0&&eapu2>0)for(int i=0;i<lB;i++)stBl[i]=(1/(voittoR[
                step+1]+moltuhottu[step+1]))*((B[step+1][nB-lB+1+i]-B[
                step][nB-lB+1+i])/eapu2*jatkuu[step]+stBl[i]*(voittoR[
                step]+moltuhottu[step]));
150         //for(int i=0;i<lR;i++)stRl[i]=R[step+1][nR-lR+1+i]/(1-krapu1
                );
151         //for(int i=0;i<lB;i++)stBl[i]=B[step+1][nB-lB+1+i]/(1-krapu2
                );
152
153         //Renormalisoidaan saadut vektorit jakaumiksi
154         double apu=0;
155         for(int i=0;i<=nB;i++)apu+=B[step+1][i];
156         assert(apu<1.001&&apu>0.999);
157         B[step+1]=MTimesC(B[step+1],1/apu);
158         apu=0;
159         for(int i=0;i<=nR;i++)apu+=R[step+1][i];
160         assert(apu<1.001&&apu>0.999);
161         R[step+1]=MTimesC(R[step+1],1/apu);
162         //MPrint(R[step+1]);
163
164         eapu1=krapu1;
165         eapu2=krapu2;
166
167         step++;
```

```
168   }
169
170
171
172   //luo ammuntamatriisin
173   public double[][] ammuntamatriisi(int n, double ph, double
          tulinopeus, int luovutus){
174     double[][] p;
175     if(tulinopeus==0||n<luovutus)return MDiag(n+1,1);
176     else{
177       p=MNull(n+1,n+1);
178       for(int maaleja=n;maaleja>=0;maaleja--){
179         double pk;
180         if(maaleja>0)pk=1-Math.pow((1-ph),tulinopeus/maaleja);
181         else {pk=0;}
182         for(int i=0;i<=maaleja;i++)
183           if(maaleja>=luovutus)p[n-maaleja+i][n-maaleja]=DBinom(
                maaleja,i,pk);
184           else p[n-maaleja+i][n-maaleja]=DBinom(maaleja,i,0);
185       }
186       return p;
187     }
188
189
190   }
191
192   //palauttaa binomijakauman tiheysfunktion arvon arvoille n, k,
        p
193   public double DBinom(int n, int k, double p){
194     if(k>n)return 0;
195     double taikakerroin=1;
196     for(int i=k+1;i<=n;i++)taikakerroin*=i;
197     for(int i=1;i<=n-k;i++)taikakerroin/=i;
198     if(p==1||p==0){
199       if(k==0)return 1;
200       else return 0;
201     }
202     return taikakerroin*Math.pow(p, k)*Math.pow((1-p),(n-k));
203   }
204
205
206   //Palauttaa taulukon joka on t ynn  nollia
207   public static double[][] MNull(int dim1,int dim2){
208     double[][] M=new double[dim1][dim2];
209     for(int i=0;i<dim1;i++)for(int j=0;j<dim2;j++)M[i][j]=0;
210     return M;
211   }
212
```

```
213   //Palauttaa taulukon joka on t ynn  nollia
214   public double[] VNull(int dim1){
215     double[] M=new double[dim1];
216     for(int i=0;i<dim1;i++)M[i]=0;
217     return M;
218   }
219
220   //Palauttaa taulukon joka on t ynn  nollia
221   public double[][][][] MNull(int dim1,int dim2, int dim3){
222     double[][][][] M=new double[dim1][dim2][dim3];
223     for(int i=0;i<dim1;i++)for(int j=0;j<dim2;j++)for(int k=0;k<
          dim3;k++)M[i][j][k]=0;
224     return M;
225   }
226
227   //Palauttaa dim x dim taulukon jonka diagonaalilla on vakiota C
228   public double[][] MDiag(int dim,double C){
229     double[][] M=MNull(dim,dim);
230     for(int i=0;i<dim;i++){
231       M[i][i]=C;
232     }
233     return M;
234   }
235
236   //kertoo kesken  n matriisit A ja B
237   public double[][] MTimes(double[][] A,double[][] B){
238     if(A[0].length!=B.length){System.err.println("paskan␣
          matriisin␣annoit");return null;}
239     double[][] C=new double[A.length][B[0].length];
240     for(int i=0;i<C.length;i++)for(int j=0;j<C[0].length;j++){
241       C[i][j]=0;
242       for(int k=0;k<B.length;k++)C[i][j]+=A[i][k]*B[k][j];
243     }
244     return C;
245   }
246
247   //kertoo kesken  n vaakavektorin A ja matriisin B
248   public double[] MTimes(double[] A,double[][] B){
249     if(A.length!=B.length){System.err.println("paskan␣matriisin␣
          annoit");return null;}
250     double[] C=new double[B[0].length];
251     for(int i=0;i<C.length;i++){
252       C[i]=0;
253       for(int k=0;k<B.length;k++)C[i]+=A[k]*B[k][i];
254     }
255     return C;
256   }
257
258   //kertoo kesken  n matriisin A ja psytyvektorin B
```

```
259   public double[] MTimes(double[][] A,double[] B){
260     if(A[0].length!=B.length){System.err.println("paskan␣
            matriisin␣annoit");return null;}
261     double[] C=new double[A.length];
262     for(int i=0;i<C.length;i++){
263       C[i]=0;
264       for(int k=0;k<B.length;k++)C[i]+=A[i][k]*B[k];
265     }
266     return C;
267   }
268
269
270
271   //palauttaa matriisin A transpoosin
272   public double[][] MTranspose(double[][] A){
273     double [][] C = new double[A[0].length][A.length];
274     for(int i=0;i<A.length;i++)for(int j=0;j<A[0].length;j++)C[j
            ][i]=A[i][j];
275     return C;
276   }
277
278   //kertoo matriisin A vakiolla b
279   public double[][] MTimesC(double[][] A,double b){
280     double[][] C=new double[A.length][A[0].length];
281     for(int i=0;i<C.length;i++)for(int j=0;j<C[0].length;j++){
282       C[i][j]=A[i][j]*b;
283     }
284     return C;
285   }
286
287   //kertoo vektorin A vakiolla b
288   public double[] MTimesC(double[] A,double b){
289     double[] C=new double[A.length];
290     for(int i=0;i<C.length;i++){
291       C[i]=A[i]*b;
292     }
293     return C;
294   }
295
296   //laskee yhteen matriisit A ja B
297   public double[][] MPlus(double[][] A,double[][] B){
298     if(A.length!=B.length||A[0].length!=B[0].length)return null;
299     double[][] C=new double[A.length][B[0].length];
300     for(int i=0;i<C.length;i++)for(int j=0;j<C[0].length;j++){
301       C[i][j]=A[i][j]+B[i][j];
302     }
303     return C;
304   }
305
```

```
306     //tulostaa matriisin debuggaustarkoituksiin
307     public void MPrint(double[][] M){
308       for(int i=0;i<M.length;i++){
309         for(int j=0;j<M[0].length;j++)System.out.print(M[i][j]+" ")
                ;
310         System.out.println();
311       }
312
313     }
314
315     //tulostaa matriisin debuggaustarkoituksiin
316     public void MPrint(double[] M){
317       for(int i=0;i<M.length;i++){
318         System.out.print(M[i]+" ");
319
320       }
321       System.out.println();
322     }
323
324     //tulostaa matriisin debuggaustarkoituksiin
325     public void MPrint(double[] M, PrintWriter p){
326       for(int i=0;i<M.length;i++){
327         p.print(M[i]+" ");
328
329       }
330     }
331 }
```