Kaksiulotteisesta fraktionaalisesta Brownin liikkeestä ja kuvankoodauksesta

Aleksi Penttinen

13.8.1999

1 Johdanto

Tässä työraportissa käsitellään kaksiulotteisen fraktionaalisen Brownin liikkeen soveltamista kuvatiedon mallintamiseen ja toisaalta esitellään tapoja simuloida itse prosessia. Käsittely on varsin karkea ja tulee vain kokoamaan tehtyjä havaintoja ja selittämään käytettyjä menetelmiä. Varsinainen työ on suoritettu Mathematica-ohjelmistolla ja keskeinen osa tätä työtä ovatkin tehdyt ohjelmat.

2 Kuvankoodaus FBM-mallilla

2.1 Yleistä

Kuvainformaatiolle on tyypillistä, että kuvassa toisiaan lähellä olevat pisteet ovat usein tummuusasteeltaan samankaltaisia. Tutkitaan kuinka tätä dataa voidaan mallintaa stokastisella prosessilla, 2-ulotteisella fraktionaalisella Brownin liikkeellä (FBM).

Mallin tarkoitus on mahdollistaa, esim. tiedonsiirron yhteydessä, kuvadatan kohtuullisen tarkka ennustaminen aiemmin välitetyn informaation perusteella, jolloin siirrettävän tiedon määrä vähenee. Toimintaperiaatteena on se, että lasketaan mallilla ennuste, sekä ennenkaikkea ennustevirheen jakauma, jonka avulla voidaan suorittaa esim. Huffman-koodaus (ks. [1]) ennustevirheelle. Pakattu tieto voidaan sitten saman jakauman avulla purkaa vastaanottopäässä. Mallin ansiosta ylimääräistä informaatiota joudutaan siirtämään vain kolmen prosessiparametrin verran.

Tässä yhteydessä 2dFBM tulkitaan siten, että se liittää pinta-alaan pinnan värimäärän ja nyt kuvan ajatellan olevan jonkin tällaisen prosessin $Z_{\mathbf{x}}$ realisaatio. $Z_{\mathbf{x}}$ on gaussinen ja sillä on stationaariset lisäykset. Lisäksi olkoon $\mathbf{x}$ vastaavan vektorin määrittämä suorakaidealue, jolloin

$$Z_{\mathbf{x}}(\alpha\mathbf{x}) \sim \alpha^{2H} Z_{\mathbf{x}}(\mathbf{x}) \tag{1}$$

Tästä seuraa esimerkiksi se, kun pinta-ala kasvaa nelinkertaiseksi, niin värimäärän varianssi kasvaa vastaavasti 4^{2H}-kertaiseksi.

Prosessi määritteleekin yksikäsitteisesti mm. kuvan eri alueiden väliset korrelaatiot. Itse prosessi puolestaan voidaan määritellä kolmella parametrillä; m, a ja H, missä m on odotusarvo, a varianssiamplitudi ja H ns. Hurstin parametri,

mikä kuvaa prosessin lisäysten välisiä korrelaatioita. Kun H=0.5 $Z_{\mathbf{x}}$ on tavallinen 2D Brownin liike (riippumattomat lisäykset, kuva kohinaa) ja kun H=1 lisäykset ovat vakioita (täysin pos. korreloituneita, kuva tasavärinen). Keskeinen huomio on se, että kahden pisteen välinen kovarianssi riippuu FBM-mallissa vain H:sta ja pisteiden välisestä etäisyydestä (ks. luku 2.3).

2.2 Hierarkinen malli

Tutkitaan kokoa $2^n \times 2^n$ olevia 8-bit harmaasävykuvia. Mallissa käsiteltävä informaatio on "mustan värin" määrä tietyssä (neliö)alueessa. Edetään hierarkisesti ylhäältä alaspäin siten, että alue jaetaan neljään lohkoon, puolittamalla molemmat sivut ja nämä osat jälleen jaetaan jne. Värimäärä emolohkossa on siis aina neljän tytärlohkon värimäärien summa. Aloitetaan värimäärästä koko kuvassa ja jatketaan kunnes kukin alue on mustan värin määrä yhdessä pikselissä, jolloin kuva on valmis. Rajoitetaan käytettyjen lukuarvojen tarkkuus 8 bittiin, jolloin suuret värimäärät pyöristetään muotoon $4^n \cdot$8-bittinen luku.

Ennuste kullekin lohkolle ja ennustevirheen varianssi lasketaan ehdollisesta jakaumasta, joka riippuu jo koodattujen (hierarkiassa saman ja yhtä ylemmän tason), lähellä sijaitsevien lohkojen arvoista. Riippuvuuden määrittelee taustalle oletettu FBM-malli. Koodaaminen joudutaan suorittamaan jossakin järjestyksessä ja siitä riippuu ehdollistamiseen käytettävissä olevat saman tason elementit, eli ne jotka on jo koodattu. Tässä yhteydessä järjestykseksi on valittu vasemmalta oikealle ja ylhäältä alas.

Joka tapauksessa, kun lohko jaetaan neljään eri osaan, tulee myös ehdollisten jakaumien laskemiseen 4 perustapausta, jotka sitten saattavat reuna-alueilla muuttua. Kolme tapausta lasketaan normaalisti mallin mukaan (vaikkakin eri lohkoja käyttäen). Neljännessä tapauksessa tunnetaan ylemmän tason (emo)lohkon värimäärä ja kaikkien muiden sisarlohkojen värimäärät, joten teoriassa lohkon värimäärä voidaan näistä laskea. Nyt kuitenkin johtuen 8 bitin tarkkuudesta 4. lohkon värimäärälle jää 2 bitin epätarkkuus eli 4 eri vaihtoehtoa. Tämä voidaan koodata suoraan 2 bitillä, sillä on epätodennäköistä, että yksi vierekkäisistä värimääristä olisi kaksi kertaa muita todennäköisempi, jolloin vasta saavutettaisiin etua todellisen jakauman avulla koodatessa.

Reuna-alueilla otetaan ehdollistavista lohkoista käyttöön ne mitkä pystytään. Jos nämä lohkot ovat samalla tai ylemmällä hierarkiatasolla koodattavan lohkon naapureita, tulee kaiken kaikkiaan 7 erikoistapausta perustapauksen lisäksi; 3 kulmaa ja 4 reunaa. Esimerkiksi käytetyllä koodausjärjestyksellä vasemman yläkulman koodaamisessa on käytettävissä tieto ainoastaan emolohkosta.

Käytännössä siis kuvan kompressiovaiheessa identifioidaan kuvasta m, a ja H ja tämän jälkeen lasketaan värimääriä hierarkisesti lohkoissa ja pakataan ne ehdollisten jakaumien avulla. Purettaessa samat jakaumat konstruoidaan parametrien ja jo puretun informaation avulla.

Nyt olennaisia kysymyksiä ovat mm.: Alueiden välisten kovarianssien ja ehdollisten jakaumien laskeminen FBM-mallin mukaan? Mitä lohkoja tulisi huomioida jakaumien muodostamisessa? Mikä on H:n arvo annetussa kuvassa?

2.3 Kovarianssien laskeminen

Ehdollisten jakaumien laskeminen vaatii siis sen, että tunnetaan eri lohkojen väliset kovarianssit. FBM-mallissa kahden pisteen kovarianssi $Cov[Z_{\mathbf{x}}, Z_{\mathbf{y}}] \sim \frac{1}{r^a}$, missä r on pisteiden välinen etäisyys ja $a = 4-4H$. Kovarianssi prosessin arvojen välillä kahdessa mielivaltaisessa alueessa X ja Y voidaan laskea intergraalina alueiden yli;

$$Cov[Z_X, Z_Y] = \int_X \int_Y \frac{1}{(\mathbf{x}-\mathbf{y})^{a/2}} d\mathbf{x}\, d\mathbf{y} \qquad \mathbf{x} \in X, \mathbf{y} \in Y$$

Tämä kuitenkin 2-ulotteisessa tapauksessa vaatii nelinkertaisen integroinnin (numeerisesti) ja ei näin ole kovin tehokas. Vaihtoehtoisesti voidaan laskea varianssi mielivaltaisessa alueessa ja sitten eri alueiden variansseja ja laskukaavoja käyttäen laskea haluttu kovarianssi. Lasketaan varianssi suorakaidealueessa (sivut x ja y) (summa kaikkien differentiaalilohkojen välisistä kovariansseista) napakoordinaatistossa, jolloin;

$$Var[X] = 4 \int_0^{\pi/2} d\varphi \int_0^{\infty} r \frac{1}{r^a} A(r, \varphi) dr$$

Tässä $A(r, \varphi) = (y - r\sin\varphi)^+ (x - \cos\varphi)^+$, eli sen leikkauksen pinta-ala, joka syntyy alkuperäisen (x,y)-suorakaiteen ja suuntaan φ matkan r siirretyn (x,y)-suorakaiteen välille. Nyt integraali voidaan laskea analyyttisesti r:n suhteen ja jäljelle jää yksi numeerisesti laskettava:

$$\tan\varphi_0 = \beta = \frac{y}{x} \qquad R(\varphi) = \begin{cases} \frac{x}{\cos\varphi} & \varphi \le \varphi_0 \\ \frac{y}{\sin\varphi} & \varphi \ge \varphi_0 \end{cases}$$

$$\begin{aligned} Var[X] = \quad & x^{4-a} 4 \textstyle\int_0^{\pi/2} \{ \frac{\beta}{2-a} R(\varphi)^{2-a} - (\sin\varphi + \beta\cos\varphi) \frac{1}{3-a} R(\varphi)^{3-a} + \\ & \sin\varphi\cos\varphi \frac{1}{4-a} R(\varphi)^{4-a} \} d\varphi \end{aligned}$$

Nyt yleinen kahden korttelietäisyydellä (x,y) olevan neliölohkon (toisen lohkon vasen alakulma origossa, toinen lohko 1. neljänneksessä) välinen kovarianssi voidaan laskea rekursiivisesti varianssien avulla lähtien kaavan 2 avulla lasketun (x+a,y+a)-kokoisen (a ylemmän neliölohkon sivun pituus) alueen varianssista ja käyttäen hyväksi laskukaavaa $Var[\sum_{i=1}^n X_i] = \sum_{i=1}^n \sum_{j=1}^n Cov[X_i, X_j]$. Tällä tavoin laskettuna pienillä etäisyyksillä laskentatehon tarve on huomattavasti numeerista integrointia pienempi. Suuremmilla etäisyyksillä/lohkoilla tilanne on päinvastoin (ellei kovariansseja lasketa analyyttisesti etukäteen) ja numeerinen integrointi käytännöllisempää.

2.4 Ehdolliset jakaumat

Käytetyllä menetelmällä ehdollinen jakauma lasketaan siis jokaiselle koodattavalle lohkolle erikseen. Jakauma kertoo koodattavan lohkon kunkin väriarvon todennäköisyyden ja sen varianssi pyritään saamaan pieneksi, jotta ennustevirhe voidaan pakata tehokkaasti. Luonnollisesti mitä useampia ympäröiviä lohkoja huomioidaan jakaumaa laskettaessa, sitä pienempi varianssiestimaatti on ja sitä parempi kompressio saavutetaan, mikäli malli on voimassa. On kuitenkin

makuasia kuinka monta lohkoa ehdollistamisessa huomioidaan sillä parannus ei ole enää kovinkaan merkittävä verrattuja lisääntyneeseen monimutkaisuuteen kun otetaan useampia kuin 3 lohkoa (emo + sivunaapurit) mukaan.

FBM-prosessille ehdollinen jakauma on gaussinen ja riippuu lohkojen arvoista sekä niiden välisistä kovariansseista seuraavasti. Symmetrinen $\mathbf{\Gamma}$-matriisi sisältää lohkojen väliset kovarianssit, ensimmäisellä rivillä/sarakkeella koodattavana olevan z (tässä siis skalaari) kovarianssit muiden lohkojen kanssa. $\mathbf{z_2}$ sisältää ehdollistamisessa käytettyjen lohkojen arvot (tässä: värimäärä). Merkitään vielä $\mathbf{A} = \mathbf{\Gamma^{-1}}$ ja ositetaan se seuraavasti;

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix}$$

Missä $\mathbf{A}_{11}$ on yleisesti neliömatriisi kokoa $dim z$, ja siis nyt skalaari. Nyt FBM-mallin mukaisesti ehdollinen jakauma [2] on gaussinen odotusarvolla ja kovarianssilla:

$$\begin{aligned} \mathrm{E}[z|\mathbf{z_2}] &= -{\mathbf{A}_{11}}^{-1}\mathbf{A}_{12}\mathbf{z_2} \\ \mathrm{E}[z^2|\mathbf{z_2}] &= {\mathbf{A}_{11}}^{-1} \end{aligned} \tag{2}$$

2.5 H:n identifiointi

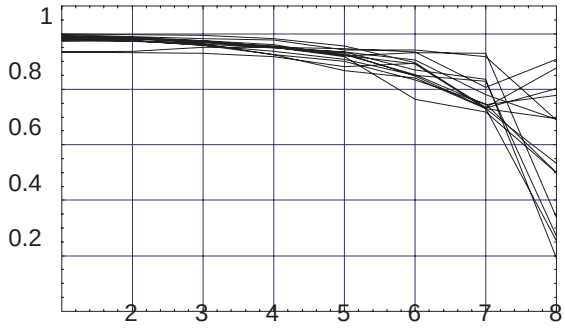
Nyt siis tunnetaan kuinka malli toimii, mutta vielä olisi pystyttävä määräämään ne kolme prosessiparametriä, mitkä kuvaavat koodattavaa kuvaa parhaiten. FBM-prosessin parametrit on laskettava kullekin kuvalle erikseen. Keskimääräinen värimäärä m voidaan helposti laskea kuvadatasta, mutta hieman ongelmalliseksi muodostuu H:n etsiminen ja sitä kautta myös varianssin estimointi suurilla H:n arvoilla.

Yksinkertainen heuristinen tapa H:n arvioimiseksi on ns. *variance plot* [4]. Nyt kaksiulotteisessa tapauksessa estimoidaan värimäärän varianssi eri hierarkian tasoilla. Kun otetaan näistä variansseista 4-kantaiset logaritmit ja piirretään kuvaaja, niin varianssin tulisi muuttua kulmakertoimella 2H. Kulmakerroin voidaan laskea esim. sovittamalla suora PNS-menetelmällä. Ongelmana tässä kuitenkin on varianssin estimoiminen. Otetaan muutama esimerkkikuva kokoa 512x512 (kuvankäsittelyyn tarkoitettuja kuvia löytyy mm. [5]) ja estimoidaan varianssi aluksi yksinkertaisesti otosvarianssin kaavalla [4] (kuva 2.5).

Pikselitasolla estimaatit ovat väliltä $(0.93, 0.999)$, kun taas ylimmillä tasoilla lähes satunnaisia. Otosvarianssi ei kuitenkaan ole välttämättä paras tapa estimoida H. Korreloituneelle prosessille harhaton Var-estimaatti [6] olisi:

$$s^2 = \frac{1}{n - n^{1-2(1-H)}} \sum_i (x_i - \bar{x})^2 \tag{3}$$

Tässä ongelmana tietysti on H:n mukanaolo, joten ratkaisu joudutaan suorittamaan iteratiivisesti. Nyt kuitenkin käy niin että H kasvaa ilman rajoja ja menee aina yli 1. Toisaalta jos yritetään estimoida H lokaalisti erikseen käytetyille hierarkiatasopareille, havaitaan että iteraatio konvergoi jos aluksi $H < 0.9$. Kuten edellisestä kuvastakin havaitaan H-estimaatti on aina alimmilla tasoilla tämän rajan yläpuolella. Jostakin syystä siis suurilla H:n arvoilla estimaatti kasvaa



Kuva 1: H-estimaatit, otosvarianssimenetelmä

jatkuvasti iteraation edetessä. Mallin käypyyttä voidaan tällä perusteella siis epäillä.

Ongelmaa voi sitten yrittää lähestyä myös toista kautta, sillä sovelluksen kannalta keskeistä on loppujen lopuksi kuitenkin vain datan kompressio. Voidaankin yrittää optimoida millä a:n ja H:n arvoilla saavutetaan paras pakkaussuhde (pakattu bittimäärä/alkuperäinen). Kokeiluilla pystytään löytämään lähes optimaaliset (a,H), mutta osoittautui myös, että näitä arvoja ei kuvadatasta pysty estimoimaan. Periaatteessa olisi käytettävä jotakin 2-ulotteista optimointialgoritmiä, kuten Hooke-Jeeves, mutta ongelmaksi muodostuu kompressiosuhteen laskeminen annetuilla (a,H), sillä esim. käytetyllä kuvakoolla kys. funktion evaluointi vie helposti kymmeniä minuutteja ja algoritmissä tarvitaan helposti kymmeniä-satoja evaluointeja.

Siis FBM-mallilla saavutetaan jonkinasteinen kompressio, mutta taustalla olevat optimaaliset prosessiparametrit a ja H eivät ole kuvasta estimoitavissa mallille tyypillisin menetelmin. Tutkitaan kuitenkin kuinka hyvä näin kokeilemalla saavutettu pakkaussuhde on.

2.6 Yleinen lineaarinen sovitus

Jotta saataisiin kuva FBM-mallin toimivuudesta täytyy vertailupohjana käyttää kuvalle yleisempää mallia, jossa kukin lohko y riippuu lineaarisesti sopivasti valituista toisista lohkoista x_i tietyillä kertoimilla a_i, ts.

$$y = a_0 \cdot 1 + a_1 x_1 + a_2 x_2 + \dots$$

Käytännössä kyseessä on sama malli, jota käytettiin edellä FBM-koodauksen yhteydessä, mutta nyt tarkoitus on laskea kertoimet a_n kullekin kuvalle erikseen siten, että jokaiselle lohkolle yhtälö toteutuu mahdollisimman hyvin pienimmän neliösumman mielessä. Edellähän kyseiset kertoimet laskettiin taustalle oletetun FBM-mallin avulla, jolloin kertoimet voitiin "pakata" kolmeen prosessiparametriin. Merkinnät; $\mathbf{y}$ on vektori, jossa pisteiden arvot $(y_1, y_2, y_3, \dots, y_n)^T$, $\mathbf{X}$ on matriisi, jossa vaakarivillä $i \in (1, \dots, n)$ on lohkoa y_i vastaavat $(1, x_{i1}, x_{i2}, \dots)$ ja $\mathbf{a} = (a_0, a_1, \dots)^T$ Lasketaan optimaaliset kertoimet $\mathbf{a}$, eli minimoidaan neliösumma laskemalla derivaatan nollakohta;

$$min \qquad (\mathbf{y} - \mathbf{Xa})^{\mathbf{T}}(\mathbf{y} - \mathbf{Xa})$$

$$\frac{d}{d\mathbf{a}}(\mathbf{y}-\mathbf{X}\mathbf{a})^{\mathbf{T}}(\mathbf{y}-\mathbf{X}\mathbf{a}) = -2\mathbf{X}^{\mathbf{T}}\mathbf{y} + 2\mathbf{X}^{\mathbf{T}}\mathbf{X}\mathbf{a} = 0$$
$$\Rightarrow \mathbf{a} = (\mathbf{X}^{\mathbf{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}\mathbf{y} \quad (4)$$

Tiedonsiirron tms. yhteydessä poimitaan kuvasta $\mathbf{y}$, $\mathbf{X}$ ja lasketaan (kaavalla 4) $\mathbf{a}$, joka siirretään ennustevirheen jakaumaparametrien (tai koodisanakirjan) kanssa vastaanottajalle. Vastaanottopäässä ennustetaan mallilla tulevat pisteet jo olemassa olevien avulla ja näin tarvitsee välittää ainoastaan ennustevirhe, joka kapeasti jakautuneena voidaan pakata tehokkaasti.

Ongelmana on siis valita lohkot x_i ja niiden määrä sopivasti, jotta ennustevirheen varianssi olisi pieni ja toisaalta välitettävien kertoimien määrä kohtuullinen. Tämän jälkeen olisi vielä löydettävä sopiva tapa kuvata ennustevirheen jakauma mahdollisimman vähin parametrein.

2.6.1 Tasomalli

Yksinkertaisin lähestymistapa lineaariseen malliin on ilmoittaa kukin pikseli jo koodattujen naapuripikseleiden avulla, jolloin siis kaikki lohkot ovat kuvapikseleitä. Tyypillisesti, jos koodaus suoritetaan vasemalta oikealle ja ylhäältä alas, otetaan mukaan naapurit vasemmalta, ylhäältä ja vasemmasta yläkulmasta. Useampien pikseleiden huomioiminen hidastaa turhaan algoritmiä (erikoisjärjestelyjä reuna-alueilla), lisää muistintarvetta ja parannus kompressiosuhteessa on kuitenkin korkeintaan luokkaa $\frac{1}{100}$ (256x256 kuvalla). Reunoissa käytetään "ennusteena" edellistä arvoa.

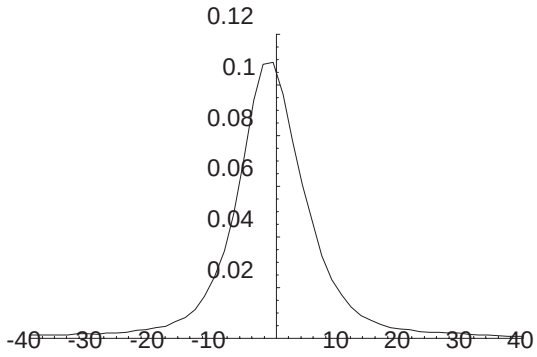
Saavutettu kompressiosuhde riippuu luonnollisesti kuvakoosta, mutta esimerkiksi 256×256 kuva joka onnistuttiin pakkaamaan 0.75:een (hierarkisella) FBM-mallilla saatiin 0.65:een tällä menetelmällä. Yleisesti kyseisellä kuvakoolla pakkaussuhde vaihteli 0.59-0.65 välillä eri testikuvilla. Lasketut kompressiosuhteet ovat teoreettisia (alarajoja) entropiakoodauksella. Lisäksi on huomattava, että lineaariselle mallille käytettiin tarkkaa ennustevirheen jakaumaa vaikkakaan sen välittäminen ei olisi käytetyllä tarkkuudella tulosta muuttanut.

Ennustevirheen jakauman (kuva 2) keskihajonnaksi tulee esimerkkikuvalla kolmea lohkoa käytettäessä 11.9. Jakauma on tyypillisesti normaalijakaumaa terävämpi ja paksuhäntäisempi jopa siinä määrin, että approksimoiminen normaalijakaumalla ei ole välttämättä hyvä ratkaisu. Mitä enemmän otetaan mukaan pisteitä x_i sitä enemmän alkaa ennustevirheen jakauma muistuttaa normaalijakaumaa. Pahimmillaan joudutaan kuitenkin koko koodisanakirja toimittamaan vastaanottajalle. Tällaisella lisäinformaatiolla ei kuitenkaan ole merkittävää vaikutusta sillä käytetty kovakoko 256x256 sisältää 65536 koodattavaa elementtiä joten muutama satakaan 8-bittisiä lukuja ei huononna koodausta huomattavasti.

2.6.2 Hierarkinen malli

Tasomallin yhteydessä havaittiin, että ylimääräisen informaation määrä ei vaikuta paljoa kompressiosuhteeseen. Niinpä FBM-mallia vastaava hierarkinen malli saattaisi myös toimia. Nyt joudutaan siis laskemaan kolmet eri kertoimet ja ennustevirheen jakaumat. Kuten FBM-mallissakin aina ns. 4. lohko (2x2 lohkon oikea alakulma) koodataan 2-bittisenä.

Tässä yhteydessä kovin perusteellisia tarkasteluja ei ollut mahdollista käytettävissä olleen ajan puitteissa tehdä, joten tulokset ovat vain suuntaa antavia. Kokeiluissa havaittiin, että merkittävää hyötyä hierarkisen mallin käytöstä



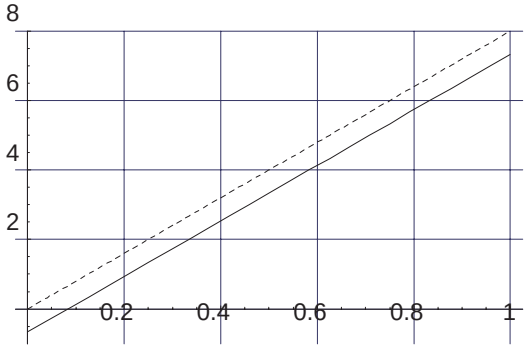
Kuva 2: Ennustevirheen jakauma

ei ole. Optimoimalla ehdollistamisessa käytettävät lohkot saataneen kuitenkin hieman tasomallia parempi pakkaussuhde. Esimerkkikuvalla saatiin pakkaussuhdearvioksi pahimmillaan noin 0.64. Ongelmia hierarkisen mallin kanssa on se, että lohkojen varianssi kasvaa ylemmille tasoille mentäessä ja näin ei pystytä naapureiden avulla rajaamaan ennustevirheen jakaumaa kovinkaan tehokkaasti ja tämä kompensoi saavutetun edun jakauman keskihajonnassa pikselitasolla ja 4. lohkon tehokkaan pakkaamisen.

2.7 Vertailua

Tasomallia ja hierarkista mallia voidaan parhaiten verrata saavutetulla pakkaussuhteella. Kuvassa 3 on esitetty hierarkisen mallin teoreettinen pakkaussuhteen ja keskimääräisen bittimäärän/8-bit väriarvo välinen riippuvuus (tasomalli katkoviivalla). Periaatteessa sekä taso- että hierarkisessa mallissa koodataan sama määrä arvoja, mutta ero tulee neljännen lohkon epätarkkuudesta. Toisaalta hierarkisessa mallissa päästään kapeampiin ennustevirheen jakaumiin, mikä parantaa kompressiota tasomalliin nähden.

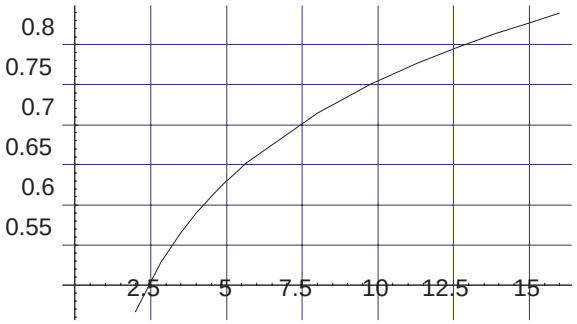
Miksi lineaarinen malli toimii sitten paremmin kuin FBM-malli? Keskeisiä ha-



Kuva 3: Hierarkinen vs tasomalli

vaintoja on se, että FBM-mallilla sivunaapurit ovat samanarvoisia. Optimoidulla mallilla näin ei ole, vaan ero voi olla jopa 100% luokkaa. Tämä johtuu kuvan ominaisuuksista; valon tulosuunnasta, geometrisistä rakenteista ym. Toi-

saalta FBM-malli antaa myös optimoituun malliin nähden aivan liian suuren painoarvon emolohkolle (etenkin, jos H on pieni). Suurimpia syitä lienee kuitenkin se, että ennustevirheen jakaumaa kuvataan normaalijakaumalla. Kuvassa 4 on esitetty teoreettinen pakkaussuhde normaalijakauman keskihajonnan funktiona ja huomataan, että jakaumaa tulee olla hyvin kapea jotta vertailukelpoinen kompressio saavutetaan. Tämä ei kuitenkaan ole realistinen vaatimus etenkään ylemmillä hierarkiatasoilla. Malli pystyy pakkaamaan tehokkaasti pikselitason, mutta tehottomuus vaivaa ylemmillä tasoilla. Kuvadata on hyvin



Kuva 4: Pakkaussuhde N-jakauman keskihajonnan funktiona

FBM-realisaation kaltaista paikallisesti, mutta laajemmassa mittakaavassa kuvadata on ennemminkin eräänlainen moduloitu prosessi, jolla on tasaisia tummia ja vaaleita alueita, kun FBM-realisaatio puolestaan tasoittuu tasaisesti koko alueessa H:n kasvaessa. Lopuksi voidaankin yhteenvetona todeta, että edeltäkäsin FBM-mallin suurin vahvuus kuvankoodauksessa - vähäinen lisäinformaation tarve - ei pysty ollenkaan kompensoimaan mallin epätarkkuuden aiheuttamaa tehottomuutta.

3 2-dimensioisen FBM-realisaation generointi

3.1 Yleistä

FBM-prosessin Z_t realisaatioita voidaan generoida määritelmän mukaan seuraavasti [2]:

$$\mathbf{z} = \mathbf{\Gamma^{1/2}w} \tag{5}$$

Tässä $\mathbf{z}$ on prosessin Z_t arvot $k \times 1$-vektorissa, $\mathbf{\Gamma} = \mathrm{E}[\mathbf{z}\mathbf{z}^T]$ ja $\mathbf{w} = k \times 1$-vektori riippumattomia $N(0,1)$-jakautuneita satunnaismuuttujia.

2-ulotteisessa tapauksessa asia on siis yksinkertainen; generoidaan $n^2 \times 1$ vektori (jossa on peräkkäin $n \times n$ realisaation vaakarivit ylhäältä alas) muodostamalla vastaava kovarianssimatriisi ja arpomalla $n^2 \times 1$ kokoinen vektori $N(0,1)$-jakautuneita lukuja. Kovarianssit voidaan helposti lasketa luvussa 2.3 esitetyllä tavalla, mutta erityisesti kannattaa huomata, että matriisi sisältää vain $\frac{n}{2}(n+1)$ eri arvoa ja näin matriisin muodostaminen voidaan suorittaa varsin tehokkaasti.

Käytännössä $n \times n$ kokoisella realisaatiolla kaavassa oleva kovarianssimatriisi on kuitenkin jo kokoa $n^2 \times n^2$, jolloin neliöjuuren laskeminen on tyypillisesti liian iso (muistia kuluttava) operaatio tavalliselle tietokoneelle. Esim. 433MHz Alpha selviää vielä kokoluokkaa 20x20 olevasta kuvasta, muttei enää esim. 32x32. Jos huomattavasti suurempaa muistikapasiteettia ja laskentatehoa ei ole käytettävissä on ongelma kierrettävä jollakin tavalla.

Ongelman ratkaisumahdollisuuksia on useita, mutta tässä yhteydessä on tarkoitus käsitellä lähinnä Random Midpoint Displacement(RMD)-menetelmän yleistys kaksiulotteiseen tapaukseen ja sen paranneltu versio. Lisäksi luodaan lyhyt katsaus muihin potentiaalisiin ratkaisumahdollisuuksiin.

3.2 RMD

RMD-menetelmä perustuu kumulatiiviseen hierarkiseen malliin (sama mitä edellä käytettiin koodaukseen), jossa emoyksikön arvo on sen tytäryksiköiden arvojen summa. Tytäryksiköiden arvot voidaan generoida tämän summaominaisuuden sekä niiden emon arvolla ehdollistetun yhteisjakauman avulla. Ehdollisen yhteisjakauman (gaussinen) määrää yksikäsitteisesti FBM-malli.

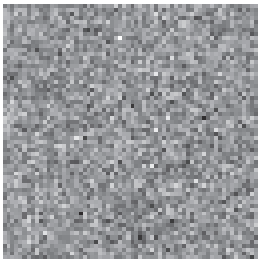
Kaksiulotteisessa tapauksessa yksinkertaisimmillaan emolohko muodostuu 4 tytärlohkosta (2x2), joista kolme (vektori $\mathbf{z_1}$) voidaan generoida nyt kaavalla:

$$\mathbf{z_1} = \mathrm{E}[\mathbf{z_1}|\mathbf{z_2}] + \mathrm{E}[\mathbf{z_1}\mathbf{z_1}^{\mathbf{T}}|\mathbf{z_2}]^{1/2}\mathbf{w} \tag{6}$$

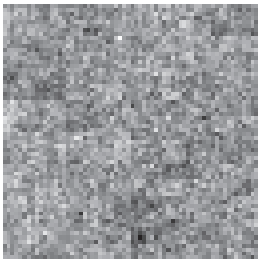
Tässä $\mathbf{w}$ on siis 3-vektori satunnaislukuja N(0,1)-jakaumasta ja z_2 emolohkon arvo. Kyseiset odotusarvot lasketaan kuten (2) käyttämällä kovarianssimatriisina kolmen tytärlohkon ja emolohkon välisistä kovariansseista muodostettua matriisia. Kun kolme tytärlohkon arvoa on selvillä saadaan neljäs mallin kumulatiivisuuden perusteella vähentämällä emolohkon arvosta jo muodostetut (tytärlohkojen)arvot. Perusmallissa algoritmi aloitetaan arpomalla "ylin" emolohko normaalijakaumasta (käyttäen ominaisuutta 1) Mallilla realisaatiosta (kuvat 5(a)-5(f)) tulee hieman lohkomainen ja hyvin epätasainen, koska eri neljänneksissä olevilla lohkoilla on tietoa toisistaan vain alkukuvan hierarkiatason kautta. Menetelmää voidaan sitten kehittää edelleen periaatteessa kahdella tavalla. Sen sijaan, että lähdetään generoimaan realisaatiota yhdestä alkuarvosta, voidaan muodostaa (mahdollisimman suuri) alkukuva suoraan määritelmän 5 mukaan ja vasta sen jälkeen aloittaa lohkoittainen operointi. Toisaalta menetelmää voidaan parantaa ottamalla $\mathbf{z_2}$-vektoriin useampia jo generoituja lohkojen arvoja ja käyttää vastaavaa kovarianssimatriisia. Tarkoitus on valita nämä lohkot sopivasti generoitavan lohkon ympäriltä siten, että vierekkäiset lohkot ovat tietoisia myös naapureistaan.

Esimerkiksi tällainen toteutus on tehty liitteenä olevassa Matematica-ajossa, missä generointi etenee vasemmalta oikealle ja ylhäältä alas lähtien 16x16 alkukuvasta. $\mathbf{z_2}$:een on valittu 6 tytärlohkotason lohkoa sekä 4 emotason lohkoa (kuva 6, generoitavana lohkot 1-3 ja x). Lohkot 1-3 generoidaan siis kaavan 6 mukaan siis yhteisjakaumasta, joka on ehdollistettu lohkojen 4-13 arvoilla ja ns. 4. lohko (kuvassa x) saadaa vähentämällä emolohkon (10) värimäärästä lohkojen 1-3 värimäärien summa. Reuna-osissa jakauman määrityksessä käytössä ovat ne lohkot mitkä sijaitsevat kokonaan kuva-alueella.

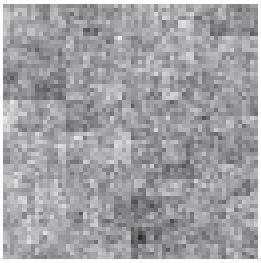
Lohkomaisuutta ei nyt enää esiinny (kuvat 5(g)-5(l), sekä liitteet). Ehdollisen yhteisjakauman laskemiseen käytettyjen lohkojen valinta voidaan suorittaa monella muullakin tavalla ilman että tulos merkittävästi kärsii. Keskeistä



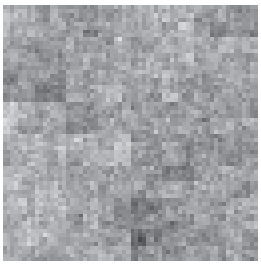
(a) RMD $H = 0.6$



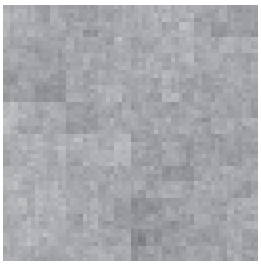
(b) RMD $H = 0.8$



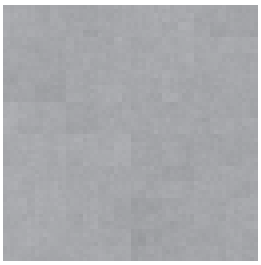
(c) RMD $H = 0.9$



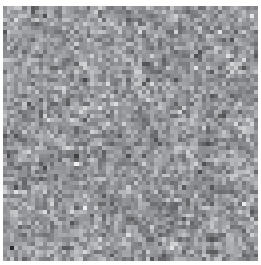
(d) RMD $H = 0.95$



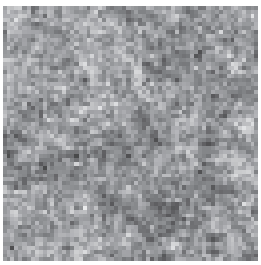
(e) RMD $H = 0.99$



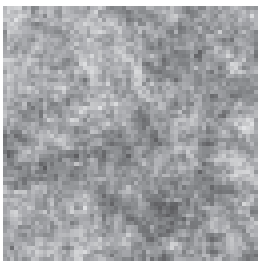
(f) RMD $H = 0.999$



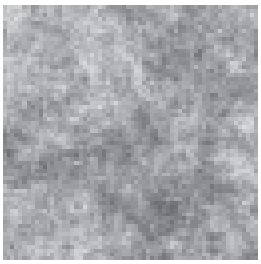
(g) $H = 0.6$



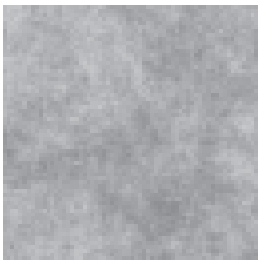
(h) $H = 0.8$



(i) $H = 0.9$



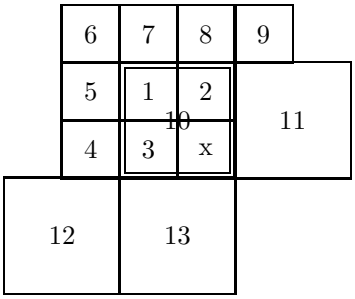
(j) $H = 0.95$



(k) $H = 0.99$

(l) $H = 0.999$

Kuva 5: Realisaatioita



Kuva 6: Parannettu RMD, eräs lohkojako

on ainostaan, että informaatiota on olemassa joka puolelta generoitavan lohkon ympäristöstä.

3.3 Muita menetelmiä

Kaavassa 5 olevan neliöjuuren laskeminen ei välttämättä tarvitse suurta kapasiteettia jos matriisin ominaisuuksia pystytään käyttämään hyväksi. Kovarianssimatriisi (kokoa $n^2 \times n^2$) on neliöalueessa symmetrinen molempien diagonaalien suhteen ja positiividefiniitti kun $H \in (0.5, 1)$, joten sillä on olemassa yksikäsitteinen neliöjuurimatriisi. Tämän lisäksi kovarianssimatriisi muodostuu n^2 kappaleesta $n \times n$ Toeplitz-lohkosta, jotka nekin sijaitsevat symmetrisesti molempien diagonaalien suhteen (lohkot muodostavat myös tavallaan Toeplitz-matriisin). Symmetriasta johtuen erilaisia lohkoja on siis $\frac{n}{4}(n+2)$ kpl. Koska kovarianssi riippuu ainoastaan etäisyydestä, kaiken kaikkiaan kovarianssimatriisissa on vain $\frac{n}{2}(n+1)$ erilaista alkoita;

$$\underset{\mathrm{n^2 \times n^2}}{\mathbf{C}} = \begin{pmatrix} A & B & C & D & \dots \\ B & A & B & C & \dots \\ C & B & A & B & \dots \\ D & C & B & A & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \underset{\mathrm{n \times n}}{\mathbf{A}} = \begin{pmatrix} a_1 & a_2 & a_3 & a_4 & \dots \\ a_2 & a_1 & a_2 & a_3 & \dots \\ a_3 & a_2 & a_1 & a_2 & \dots \\ a_4 & a_3 & a_2 & a_1 & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix}$$

Neliöjuurimatriisi on myös (per)symmetrinen ja positiividefiniitti sekä lohkorakenteinen, sillä erolla kovarianssimatriisista, että lohkot ovat sisäisesti ja ulkoisesti ainoastaan persymmetrisiä, mutta eivät Toeplitz-tyyppiä. $\surd$-matriisille täytyy siis laskea ainoastaan yläsektorin lohkot, ja niissäkin vain yläsektorin alkiot, joten muistiin tarvitaan $(\frac{n}{4}(n+2))^2$ eri alkiota n^4 sijaan. Matriisin "yläsektorilla" tarkoitetaan tässä siis niitä alkioita, jotka peilamaamalla molempien diagonaalien suhteen muodostavat koko matriisin:

$$\begin{matrix} x & x & x & x & x & x \\ & x & x & x & x & \\ & & x & x & & \end{matrix}$$

Toeplitz-tyypin matriiseille on olemassa kirjallisuudessa (mm. [3]) paljon eri-

laisia hyvin tehokkaita algoritmeja ja kenties niitä modifioimalla saadaan tehokas algoritmi myös lohkomaisen tapauksen laskemiseen.

Kokonaan toinen lähestymistapa olisi mahdollinen jos käytetään kaavaa 5 vastaavaa jatkuvaa relaatiota, missä FBM-prosessin arvot lasketaan stokastisten integraalien avulla [2].

3.4 Yhteenveto

Parannetulla RMD-mallilla, jossa ehdollistetaan generoitavat lohkot kaikkiin naapureihin, saavutetaan jo varsin hyviä tuloksia. Tuloksia pitäisi kuitenkin pystyä vertailemaan todellisiin, jotta niiden oikeellisuudesta olisi varmuutta. On selvää että käytetyt yksinkertaistukset vaikuttavat jonkin verran laatuun, mutta kuinka paljon?

Neliöjuurimatriisin tehokas laskeminen tai approksimoiminen olisi yksi keskeisistä probleemoista. Vaikka 256x256 kokoiset realisaatiot jäisivät edelleekin laskentakapasiteetin ulottumattomiin, jo pieni parannus tehokkuuteen mahdollistaisi myös isomman alkukuvan käytön RMD-menetelmässä ja tätä kautta realistisemman simulointituloksen.

3.5 Liitteet

- 2dFBM-realisaatioita molemmilla RMD-menetelmillä.

Viitteet

[1] A. K. Jain: *Fundamentals of Digital Image Processing*, Prentice Hall, 1989.

[2] I. Norros, J. Virtamo: *Handbook of FBM formulae*, COST257TD(96), September 1996.

[3] G.H. Golub, C.F. Van Loan: *Matrix Computations*, 3rd ed. Johns Hopkins U. Press, 1996.

[4] J. Beran: *Statistics for Long-memory Processes* Chapman and Hall, 1994.

[5] `http://sipi.usc.edu/services/database/Database.html`

[6] J.L. Jerkins, A.L. Neidhardt, J.L. Wang, A. Erramilli: *Operations Measurements for Engineering Support of High-Speed Networks with Self-Similar Traffic*, Teletraffic Engineering in a Competitive World, P. Key, D. Smith (Editors), Elsevier, 1999.