Volatility estimation and visualization for stock/option traders
Bachelorscriptie leerstoelen SST/SP

Peter Bosschaart Jeroen Spoor Berend Steenhuisen

9 juni 2011

Inhoudsopgave

Hoofdstuk 1

Introductie

Iedere dag wordt er aan miljarden euro's verhandeld op de aandelenmarkt. Aandelenhandelaars kijken naar beeldschermen met vele getallen er op die constant veranderen van waarde en kopen en verkopen op basis hiervan aandelen en opties. De aandeelprijs is uiteraard een van deze getallen, maar ook de volatiliteit van het aandeel moet op het scherm komen te staan. De volatiliteit is een belangrijke waarde voor aandeelhouders bij het vaststellen van de waarde van een optie, maar deze is niet direct observeerbaar. Wij zullen ons in dit onderzoek dan ook richten op het schatten van deze volatiliteit en het zichtbaar maken van deze voor de aandelenhandelaars.

Allereerst zullen we ingaan op wat volatiliteit is en wat voor model wij als basis hebben gebruikt om tot het onze te komen. Vervolgens zullen we ons eigen model introduceren en zullen we een manier behandelen waarmee we aan de hand van ons model voorspellingen kunnen doen over de waarden van de volatiliteit.

Volatiliteit is de mate van beweeglijkheid van de koers van een aandeel. Is de volatiliteit hoog, dan is het aandeel zeer beweeglijk en heeft het een hoog risico, is de volatiliteit laag, dan is het aandeel minder beweeglijk en heeft het een lager risico. De volatiliteit is een waarde die geschat wordt aan de hand van observaties van de aandeelprijs in het verleden en voorgaande schattingen van de volatiliteit. Om de waarde van een optie vast te stellen hebben aandeelhouders op ieder tijdstip de waarde van de volatiliteit van het onderliggende aandeel nodig, de volatiliteit is echter niet te observeren en zal dus op ieder tijdstip geschat moeten worden.

In 1973 introduceerden Black en Scholes [2] hun model, waarin de aandeelprijs gemodelleerd wordt met behulp van Brownse bewegingen. In 1993 publiceert Heston [7] een uitbreiding op dit model; het Heston model. Heston veronderstelt in dit model de volatiliteit van het aandeel niet meer constant, zoals Black en Scholes dit deden, en beschrijft de beweeglijkheid van de volatiliteit ook met behulp van een Brownse beweging. Het Heston model zullen wij gebruiken als basis voor ons model:

$$dS_t = \mu S_t dt + \sqrt{v_t} S_t dB_t, \tag{1.1}$$

$$dv_t = \kappa(\theta - v_t)dt + \xi\sqrt{v_t} dZ_t. \tag{1.2}$$

Vergelijking (1.1) geeft de verandering van de aandeelprijs weer, vergelijking (1.2) de verandering in de volatiliteit. De symbolen in de vergelijkingen staan voor het volgende:

- S_t is de prijs van het aandeel op tijdstip t
- v_t is de volatiliteit van het aandeel op tijdstip t
- μ is de "rate of return" van het aandeel en is constant
- θ is de lange termijnsvariantie van het aandeel en is constant
- κ is de mate waarmee v_t lokaal convergeert naar θ en is constant
- ξ is de volatiliteit van de volatiliteit; de variantie van v_t en constant
- B_t en Z_t zijn Brownse bewegingen met correlatie ρ, met ρ constant

In dit model is de aandeelprijs observeerbaar, maar de volatiliteit niet. We zullen de volatiliteit dus op ieder tijdstip moeten schatten. Dit kan met behulp van 'particle filtering'. Dit is een discrete methode van stochastisch filteren voor niet-lineaire processen, waarmee op basis van waarden uit het verleden een schatting gemaakt kan worden van de waarde op het huidige tijdstip. De output van dit filter is waar we in ons onderzoek naar op zoek zijn.

Het model waar wij mee zullen werken is een aangepaste vorm van het model van Heston, welke we verkrijgen door te werken met de natuurlijke logaritme van de aandeelprijs, de logprijs van het aandeel, in plaats van met de aandeelprijs zelf ($y_t = \ln(S_t)$). De substitutie kan uitgevoerd worden met stochastische calculus (Itô), waar we in dit verslag niet op in zullen gaan. Het model is na substitutie als volgt gegeven:

$$dy_t = (\mu - \frac{1}{2}v_t)dt + \sqrt{v_t}dB_t, \tag{1.3}$$

$$dv_t = \kappa(\theta - v_t)dt + \xi\sqrt{v_t}dZ_t. \tag{1.4}$$

De symbolen in deze vergelijkingen zijn zoals ze bovenaan deze pagina gedefiniëerd zijn. Vergelijking (1.3) beschrijft de verandering in de logprijs van het aandeel en volgt uit vergelijking (1.1) na substitutie van $y_t = \ln(S_t)$, vergelijking (1.4) beschrijft de verandering in de volatiliteit. Ook nu is slechts de logprijs observeerbaar en zullen we de volatiliteit moeten schatten. Het particle filter wat we gaan gebruiken werkt met discrete data, dus is het nodig om ons model te discretiseren, dit doen we in hoofdstuk 2.

In ons onderzoek zullen we eerst zelf het verloop van de volatiliteit genereren en aan de hand van die volatiliteit een bijbehorende aandelenmarkt simuleren, deze prijzen zullen we gebruiken als input voor ons model. Dit doen we in hoofdstuk 2. Hierbij veronderstellen we de parameters μ, θ, κ, ξ en ρ bekend.
Vervolgens gaan we met ons model en een particle filter de volatiliteit die we gegenereerd hebben schatten en kijken we hoe goed dit gebeurt en of er verbeteringen mogelijk zijn. Dit doen we in hoofdstuk 3, waar we tevens zullen uitleggen hoe particle filters werken.

Hierna zullen we ons model aanpassen, opdat er gewerkt kan worden met onbekende parameters μ, θ, κ, ξ en ρ. Dit doen we omdat van een echt aandeel deze parameters niet bekend zijn en er geen analytische formule is om deze mee te berekenen. We zullen een manier zoeken om naast de volatiliteit ook deze parameters te schatten en tot slot zullen we met dit nieuwe model de volatiliteit van echte aandelen gaan schatten. Dit nieuwe model zullen in hoofdstuk 4 afleiden.

Na deze hoofdstukken is alle theorie die we gebruiken bij ons onderzoek behandeld en zullen we in hoofdstuk 5 uitwijden over de algoritmen die we gebruiken bij het simuleren van onze modellen. De resultaten van deze simulaties behandelen we vervolgens in hoofdstuk 6. Aan de hand van deze resultaten zullen we in hoofdstuk 7 conclusies trekken en tot slot aanbevelingen voor vervolgonderzoek doen.

In de appendices staan de MATLAB-codes die we hebben gebruikt en enkele tabellen die volgen uit simulaties die we hebben uitgevoerd.

Hoofdstuk 2

Discretisatie van het model

In ons onderzoek gaan we de volatiliteit van aandelen schatten met behulp van een particle filter. Aangezien particle filters met discrete data werken, is het nodig om ons model te discretiseren. Dat zullen we in dit hoofdstuk doen.

In ons model is Z_t (vergelijkingen (1.3) en (1.4)) gecorreleerd met B_t met parameter ρ. Voor de simulatie willen we dit uit de weg gaan en schrijven ons model op een soortgelijke manier als Broadie en Kaya [3] om naar:

$$dy_t = \left(\mu - \frac{1}{2}v_t\right)dt + \sqrt{v_t}\left[\rho dZ_t + \sqrt{1-\rho^2}d\widetilde{B_t}\right], \tag{2.1}$$

$$dv_t = \kappa(\theta - v_t)dt + \xi\sqrt{v_t}dZ_t, \tag{2.2}$$

waarbij Z_t en $\widetilde{B_t}$ ongecorreleerde Brownse bewegingen zijn.

We discretiseren eerst (2.2). Een gemakkelijke manier om dit te doen is via een Eulerdiscretisatie, waarbij $t_k > t_{k-1}$, $\Delta t = t_k - t_{k-1}$ en $k = 1, 2, \ldots$:

$$\begin{aligned} v_k &= v_{k-1} + \int_{t_{k-1}}^{t_k} \kappa(\theta - v_s)\, ds \; + \int_{t_{k-1}}^{t_k} \xi\sqrt{v_s}\, dZ_s \\ &\approx v_{k-1} + \kappa(\theta - v_{k-1})\Delta t + \xi\sqrt{v_{k-1}}(Z_k - Z_{k-1}), \end{aligned} \tag{2.3}$$

waarbij $(Z_k - Z_{k-1}) \sim \mathcal{N}(0, \Delta t)$ (dit volgt uit de definitie van een Brownse beweging). We gebruiken nu het subscript k als tijdsindex om aan te duiden dat het om een discrete vergelijking gaat, de transitie hiernaar is eenvoudig: waar voorheen v_{t_k} gebruikt zou moeten worden, wordt nu v_k gebruikt en waar $v_{t_{k-1}}$ gebruikt zou moeten worden, gebruiken we v_{k-1}. Ook voor andere variabelen gebruiken we deze transitie in het vervolg van het verslag.

Met de normaalverdeelde benadering (2.3) kan de volatiliteit echter nog negatieve waarden aannemen; vanuit praktisch oogpunt gezien is dit niet mogelijk en willen we dus een betere discretisatie van de volatiliteit gebruiken. Hiertoe biedt de non-centrale χ^2-verdeling een oplossing, Broadie en Kaya [3] gebruiken deze om v_k te genereren uit gegeven v_{k-1}, waarbij geen negatieve waarden meer voor kunnen komen:

$$\begin{aligned} v_k | v_{k-1} \sim & C_1 \cdot \chi^2_d(\lambda), \\ \text{waarbij} \quad C_1 = \frac{\xi^2\left(1 - e^{-\kappa\Delta t}\right)}{4\kappa}, \quad & \lambda = \frac{4\kappa e^{-\kappa\Delta t}}{\xi^2\left(1 - e^{-\kappa\Delta t}\right)} \cdot v_{k-1}, \quad d = \frac{4\theta\kappa}{\xi^2}, \end{aligned} \tag{2.4}$$

waarbij $\chi^2_d(\lambda)$ een stochastische variabele is die non-centraal χ^2-verdeeld is met d vrijheidsgraden en parameter λ.

Nu we een discrete uitdrukking voor de volatiliteit hebben, dienen we vergelijking (2.1) nog te discretiseren om een discrete uitdrukking voor de logprijs te krijgen. Ook hierbij volgen we Broadie en Kaya [3] en veronderstellen we $t_k > t_{k-1}$, $\Delta t = t_k - t_{k-1}$ en $k = 1, 2, \ldots$:

$$y_k = y_{k-1} + \mu\Delta t - \frac{1}{2}\int_{t_{k-1}}^{t_k} v_s\, ds + \rho\int_{t_{k-1}}^{t_k} \sqrt{v_s}\, dZ_s \; + \sqrt{1-\rho^2}\int_{t_{k-1}}^{t_k} \sqrt{v_s}\, d\widetilde{B_s}\ . \tag{2.5}$$

We benaderen de eerste integraal in (2.5) met:

$$\int_{t_{k-1}}^{t_k} v_s \, ds \;\approx \frac{1}{2}(v_k + v_{k-1})\Delta t. \tag{2.6}$$

Om de tweede integraal in vergelijking (2.5) uit te rekenen, gebruiken we wat er volgt uit vergelijking (2.3) en verkrijgen:

$$v_k - v_{k-1} = \kappa\theta\Delta t - \kappa \int_{t_{k-1}}^{t_k} v_s \, ds \; + \xi \int_{t_{k-1}}^{t_k} \sqrt{v_s} \, dZ_s \; . \tag{2.7}$$

We gebruiken vergelijkingen (2.6) en (2.7) en krijgen:

$$\int_{t_{k-1}}^{t_k} \sqrt{v_s} \, dZ_s \; = \frac{1}{\xi}\left(v_k - v_{k-1} - \kappa\theta\Delta t + \frac{\kappa}{2}\left(v_k + v_{k-1}\right)\Delta t\right). \tag{2.8}$$

Tot slot dient de laatste integraal in vergelijking (2.5) nog benaderd te worden om een volledige discretisatie te geven van vergelijking (2.1).
Uit Broadie en Kaya [3] weten we dat:

$$\int_{t_{k-1}}^{t_k} \sqrt{v_s} \, d\widetilde{B_s} \; \sim \mathcal{N}(0, \sigma^2), \quad \text{waarbij} \quad \sigma^2 = \int_{t_{k-1}}^{t_k} v_s \, ds \; \approx \frac{1}{2}(v_k + v_{k-1})\Delta t. \tag{2.9}$$

Met deze gegevens kunnen we een 'sample' trekken voor $\int_{t_{k-1}}^{t_k} \sqrt{v_s} \, d\widetilde{B_s}$ en deze samen met (2.6) en (2.8) gebruiken om vergelijking (2.5) en daarmee vergelijking (2.1) volledig te discretiseren. We krijgen:

$$y_k = \; y_{k-1} + \mu\Delta t - \frac{1}{4}(v_k + v_{k-1})\Delta t + \frac{\rho}{\xi}\left(v_k - v_{k-1} - \kappa\theta\Delta t + \frac{\kappa}{2}\left(v_k + v_{k-1}\right)\Delta t\right) + \sqrt{1-\rho^2} \cdot \zeta_1, \tag{2.10}$$

waarbij ζ_1 een $\mathcal{N}(0, \sigma^2)$-verdeelde stochatische variabele is met σ^2 zoals in (2.9).

We hebben ons model nu volledig gediscretiseerd. In hoofdstuk 3 zullen we uitleggen hoe we het particle filter zullen gaan gebruiken met dit model en hoe we hiermee de volatiliteit schatten.

Met het gediscretiseerde model kunnen we ook zelf een volatiliteit genereren en aan de hand daarvan een markt simuleren. Resultaten hiervan zijn te zien in hoofdstuk 6. Deze gesimuleerde markt zullen we in eerste instantie als input gebruiken voor ons particle filter, waarmee we de volatiliteit van deze gesimuleerde markt zullen schatten. Op deze manier kunnen we onze resultaten vergelijken met de gegenereerde volatiliteit en kijken hoe goed onze schatting is. In hoofdstuk 4 leggen we uit hoe we de volatiliteit van echte markten kunnen schatten.

Hoofdstuk 3

Particle filters en het schatten van de volatiliteit

In dit hoofdstuk zullen we uitleggen hoe we de volatiliteit kunnen schatten met ons model en met behulp van 'particle filters'. Voordat we dit zullen doen is het belangrijk om te weten wat particle filters zijn en hoe deze werken. Deze theorie zullen we dan ook eerst behandelen, waarna we de theorie toepassen op onze modelveronderstelling. Vervolgens bespreken we de verschillende keuzemogelijkheden binnen het particle filter en onderzoeken we met welke van deze keuzes onze schatting van de volatiliteit het beste is.

3.1 Particle filtering

In deze paragraaf volgen wij het artikel van Petar M. Djurić et al.[6] bij het uitleggen wat particle filters zijn en hoe ze werken. Particle filters worden gebruikt wanneer men te maken heeft met twee toestanden; één waarneembare en één onwaarneembare. De waarneembare toestand is afhankelijk van de onwaarneembare toestand en de onwaarneembare toestand kan eventueel afhankelijk zijn van de waarneembare toestand. Met een particle filter kan men de onwaarneembare toestand op ieder tijdstip schatten aan de hand van waarnemingen van de waarneembare toestand en reeds geschatte waarden van de onwaarneembare toestand in het verleden, men kan in ons geval dus met een particle filter de volatiliteit van een aandeel schatten aan de hand van de aandeelprijs en de volatiliteit in het verleden. We leggen uit hoe particle filters werken aan de hand van de volgende toestandsrepresentatie, voor $k = 0, 1, 2, \ldots$:

$$\begin{aligned} x_k &= f_k(x_{k-1}, \epsilon_k), \\ y_k &= g_k(x_k, \eta_k). \end{aligned} \tag{3.1}$$

In (3.1) is y_k de waarneembare toestand en is x_k de onderliggende, onwaarneembare toestand, ϵ_k en η_k zijn storingen, welke niet onderling onafhankelijk hoeven te zijn. Het subscript k is de tijdsindex. We gaan er van uit dat alle waarden van y bekend zijn; vanaf het tijdstip 0 tot en met het tijdstip k. Nu is het dus zaak om met behulp van de waarneembare toestand y_k de onderliggende toestand x_k zo goed mogelijk te benaderen.

We bekijken de 'simultane a posteriori verdeling' van $x_0, x_1, \ldots, x_k$, de volgende evenredigheid geldt:

$$p(x_{0:k}|y_{0:k}) \propto p(x_0|y_0) \prod_{i=1}^{k} p(y_i|x_i) p(x_i|x_{i-1}). \tag{3.2}$$

Uit vergelijking (3.2) volgt volgens Petar M. Djurić et al [6] dat men $p(x_{0:k}|y_{0:k})$ kan verkrijgen uit $p(x_{0:k-1}|y_{0:k-1})$ op de volgende manier:

$$p(x_{0:k}|y_{0:k}) = \frac{p(y_k|x_k)p(x_k|x_{k-1})}{p(y_k|y_{0:k-1})} p(x_{0:k-1}|y_{0:k-1}). \tag{3.3}$$

Aangezien de transitie van $p(x_{0:k-1}|y_{0:k-1})$ naar $p(x_{0:k}|y_{0:k})$ vaak analytisch niet uit te voeren is, moeten we de oplossing zoeken in methoden die gebaseerd zijn op benaderingen. Dit is waar particle filtering ons gaat helpen; bij particle filtering worden de kansverdelingen benaderd door discrete, willekeurige metingen gedefiniëerd door 'particles' en gewichten die aan deze particles toegekend worden. Stel dat we de verdeling van $p(x)$ willen benaderen, dan gebruiken we de willekeurige metingen $\Psi = \{x^{(m)}, w^{(m)}\}_{m=1}^{M}$, waarbij $x^{(m)}$ de

particles zijn met bijbehorende gewichten $w^{(m)}$. Er worden in totaal M particles gebruikt bij de benadering. Ψ benadert de verdeling $p(x)$ door:

$$p(x) = \sum_{m=1}^{M} w^{(m)} \delta \left(x - x^{(m)} \right), \tag{3.4}$$

waarbij $\delta(\cdot)$ de Dirac-deltafunctie is. Met deze benadering worden berekeningen van verwachtingen, welke normaal gesproken moeilijke integralen bevatten, vereenvoudigd tot makkelijk uitvoerbare sommaties. Zo geven Jayesh H. Kotecha en Petar M. Djurić [8] als voorbeeld dat:

$$\mathbb{E}_{p(x)}(g(x_n)) = \int g(x_n) p(x_n)\, dx_n \tag{3.5}$$

berekend kan worden op de volgende manier:

$$\hat{\mathbb{E}}_{p(x)}(g(x_n)) = \sum_{m}^{M} w^{(m)} g(x_n^{(m)}). \tag{3.6}$$

Gebruik makend van de 'sterke wet van de grote aantallen' kan worden aangetoond dat:

$$\hat{\mathbb{E}}_{p(x)}(g(x_n)) \longrightarrow \mathbb{E}_{p(x)}(g(x_n)) \tag{3.7}$$

bijna zeker als $M \longrightarrow \infty$.

Het volgende concept wat we gebruiken is 'importance sampling'. Stel dat we de verdeling $p(x)$ willen benaderen met discrete, willekeurige metingen. Als we particles kunnen genereren van $p(x)$ zelf, kennen we ze ieder een gewicht toe van $1/M$. Als directe sampling van $p(x)$ niet mogelijk is, dan kunnen we particles $x^{(m)}$ genereren uit een verdeling $\pi(x)$, de 'importance function', en kennen we niet genormaliseerde gewichten toe aan deze particles volgens:

$$w^{*(m)} = \frac{p(x)}{\pi(x)}, \tag{3.8}$$

welke na normaliseren het volgende worden:

$$w^{(m)} = \frac{w^{*(m)}}{\sum_{m=1}^{M} w^{*(m)}}. \tag{3.9}$$

Stel nu dat de posterior verdeling $p(x_{0:k-1}|y_{0:k-1})$ benaderd wordt door de discrete, willekeurige metingen $\Psi_{k-1} = \{x_{0:k-1}^{(m)}, w_{k-1}^{(m)}\}_{m=1}^{M}$ (merk op dat de particles $x_{0:k-1}^{(m)}$ opgevat kunnen worden als particles van $p(x_{0:k-1}|y_{0:k-1})$). Gegeven Ψ_{k-1} en de observatie y_k is het nu de bedoeling om Ψ_k op te stellen. Sequentiële importance sampling methoden doen dit door particles $x_k^{(m)}$ te genereren en deze toe te voegen aan $x_{0:k-1}^{(m)}$ om $x_{0:k}^{(m)}$ te vormen en vervolgens de gewichten $w_k^{(m)}$ te updaten, zodat Ψ_k een goede schatting geeft van de gezochte kansverdeling op tijdstip k.

De importance function is vrij te kiezen, maar als deze gekozen wordt zodat deze gefactoriseerd kan worden als:

$$\pi(x_{0:k}|y_{0:k}) = \pi(x_k|x_{0:k-1}, y_{0:k})\pi(x_{0:k-1}|y_{0:k-1}), \tag{3.10}$$

en als:

$$x_{0:k-1}^{(m)} \sim \pi(x_{0:k-1}|y_{0:k-1}), \tag{3.11}$$

en:

$$w_{k-1}^{(m)} \propto \frac{p(x_{0:k-1}^{(m)}|y_{0:k-1})}{\pi(x_{0:k-1}^{(m)}|y_{0:k-1})}, \tag{3.12}$$

dan kunnen we $x_{0:k-1}^{(m)}$ uitbreiden met $x_k^{(m)}$, waarbij:

$$x_k^{(m)} \sim \pi(x_k^{(m)}|x_{0:k-1}^{(m)}, y_{0:k}). \tag{3.13}$$

De bijbehorende gewichten $w_k^{(m)}$ kunnen verkregen worden door:

$$w_k^{(m)} \propto \frac{p(y_k|x_k^{(m)})p(x_k^{(m)}|x_{k-1}^{(m)})}{\pi(x_k^{(m)}|x_{0:k-1}^{(m)}, y_{0:k})} w_{k-1}^{(m)}. \tag{3.14}$$

Vergelijking (3.14) zullen we in het vervolg de 'weight update equation' noemen.

De keuze van de importance function is belangrijk voor het functioneren van het particle filter, dit is makkelijk te zien aan de weight update equation waar de importance function een directe invloed uitoefent in de noemer. In paragraaf 3.3 gaan we in op het gebruik van verschillende importance functions en het effect hiervan op de weight update equation.

Samenvattend kunnen we x_0 tot en met x_k schatten middels 'sequential importance sampling', hierbij is i één stap waarbij we Δt (onze gekozen stapgrootte) verdergaan in de tijd. In totaal nemen we k van deze tijdstappen ($i = 0, 1, \ldots, k$). Dit gaat als volgt:

1. Trek N particles $x_0^{(m)} \sim p(x_0|y_0)$, $m = 1, 2, \ldots, N$.
2. Zet de gewichten $w_0^{(m)}$ op $1/N$.
3. De schatting van x_0 is $\sum_{j=1}^{N} w_0^{(j)} \cdot x_0^{(j)}$ (volgens (3.6)).
4. Zet $i = 1$.
5. Trek N particles $x_i^{(m)} \sim \pi(x_i|x_{0:i-1}, y_{0:i})$.
6. Bereken $w_i^{(m)}$ met behulp van (3.14).
7. De schatting van x_i is $\sum_{j=1}^{N} w_i^{(j)} \cdot x_i^{(j)}$ (volgens (3.6)).
8. Controleer of $i = k$, zo ja dan zijn we klaar. Zo nee, zet dan $i = i + 1$ en ga dan naar 5.

Wanneer we dit algoritme aanhouden is het waarschijnlijk dat op den duur een paar gewichten de overhand krijgen en heel groot worden in verhouding tot de andere gewichten, dit verschijnsel heet 'degeneracy'. Om dit tegen te gaan bestaat er 'resampling'. Dit doen we in eerste instantie als volgt:

We hebben N particles $x_k^{(m)}$ getrokken, hierbij horen de gewichten $w_k^{(m)}$, die we ook berekend hebben. We trekken nu uit alle particles $x_k^{(m)}$ één particle, de kans dat we deze particle trekken is $w_k^{(m)}$. Dit doen we N maal, vervolgens zetten we de gewichten allemaal gelijk aan $1/N$. Petar M. Djurić et al. [6] geven hier een grafische samenvatting van in figuur 3.1. In dit figuur zien we links de initiële gewichten van de particles schematisch afgebeeld in bollen die groter zijn naar mate het gewicht groter is. Vervolgens wordt er geresampled en zie je op de lijn in het midden hoevaak ieder particle gekozen wordt. Rechts van de lijn zie je dat ieder gekozen particle weer een gelijk gewicht gekregen heeft.

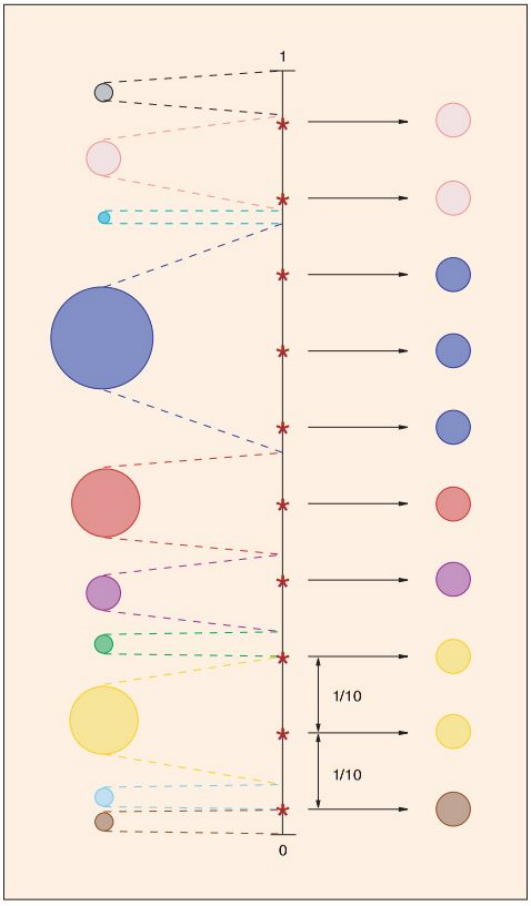
Op deze manier hebben we dus weer meer particles die een rol spelen, in plaats van een paar particles met een grote rol en een heleboel met een kleine. Het is overigens niet nodig om bij elke stap te 'resamplen', dit kost veel rekenwerk en in de praktijk geld. Een goede graadmeter om te testen of het nodig is om te resamplen is N_{eff}, de 'effective sample size':

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w_k^{(i)})^2}. \tag{3.15}$$

Er wordt een drempelwaarde N_{thr} gekozen, bijvoorbeeld $N/3$, als N_{eff} dan kleiner is dan N_{thr} wordt er geresampled en anders wordt er niet geresampled.

Er zijn meerdere manieren om te resamplen, hierboven hebben we er daar slechts één van belicht. In paragraaf 3.4 zullen we een aantal andere resamplemethoden toelichten, opdat we vervolgens kunnen gaan onderzoeken of er een optimale resamplemethode bestaat.

Nu we hebben uitgelegd hoe het particle filter in het algemeen werkt, kunnen we het particle filter toe gaan passen op ons eigen model, om vervolgens de volatiliteit van een aandeel te schatten. Hier zullen we in de volgende paragraaf over uitwijden.



Figuur 3.1: Een schematische representatie van resampling

3.2 Het schatten van de volatiliteit

Nu we weten hoe particle filters werken, kunnen we deze met ons model gaan gebruiken om de volatiliteit van een aandeel te schatten. Ons gediscretiseerde model, zoals beschreven in (2.4) en (2.10), staat in een soortgelijke toestandsrepresentatie als in (3.1). Nu passen we het particle filter toe op dit model, zodat we de volatiliteit kunnen schatten. We weten uit (3.6) dat:

$$\hat{\mathbb{E}}_{p(v_k|y_{1:k},v_{k-1})}(v_k) = \sum_{m=1}^{M} w_k^{(m)} \cdot v_k^{(m)}. \tag{3.16}$$

Uit (3.7) weten we dat:

$$\hat{\mathbb{E}}_{p(v_k|y_{1:k},v_{k-1})}(v_k) \longrightarrow \mathbb{E}_{p(v_k|y_{1:k},v_{k-1})}(v_k), \tag{3.17}$$

bijna zeker als $M \longrightarrow \infty$.

We moeten dus op zoek naar een uitdrukking voor de gewichten in deze vergelijking. Hiertoe stellen we eerst een algemene formule op voor de gebruikte gewichten $w_k^{(m)}$. Uit het artikel van Broadie en Kaya [3] en uit het artikel van Petar M. Djurić et al. [6] weten we dat

$$w_k^{(m)} = \frac{p(v_{0:k}^{(m)}|y_{1:k})}{\pi(v_{0:k}^{(m)}|y_{1:k})}, \tag{3.18}$$

merk op dat y_k pas gedefiniëerd is vanaf $k = 1$. Tevens weten we van Branko Ristic et al. [5] dat:

$$\begin{aligned} p(v_{0:k}^{(m)}|y_{1:k}) &= \frac{p(y_k|v_{0:k}^{(m)}, y_{1:k-1})p(v_{0:k}^{(m)}|y_{1:k-1})}{p(y_k|y_{1:k-1})} \\ &= \frac{p(y_k|v_{0:k}^{(m)}, y_{1:k-1}) \cdot p(v_k|v_{0:k-1}^{(m)}, y_{1:k-1}) \cdot p(v_{0:k-1}^{(m)}|y_{1:k-1})}{p(y_k|y_{1:k-1})}. \end{aligned} \tag{3.19}$$

Deze uitdrukking substitueren we in vergelijking (3.18) en krijgen:

$$\begin{aligned} w_k^{(m)} = \frac{p(v_{0:k}^{(m)}|y_{1:k})}{\pi(v_{0:k}^{(m)}|y_{1:k})} &= \frac{p(y_k|v_{0:k}^{(m)}, y_{1:k-1}) \cdot p(v_k|v_{0:k-1}^{(m)}, y_{1:k-1}) \cdot p(v_{0:k-1}^{(m)}|y_{1:k-1})}{\pi(v_k^{(m)}|v_{0:k-1}^{(m)}, y_{1:k}) \cdot \pi(v_{0:k-1}^{(m)}|y_{1:k-1}) \cdot p(y_k|y_{1:k-1})} \\ &= w_{k-1}^{(m)} \cdot \frac{p(y_k|v_{0:k}^{(m)}, y_{1:k-1}) \cdot p(v_k^{(m)}|v_{0:k-1}^{(m)}, y_{1:k-1})}{\pi(v_k^{(m)}|v_{0:k-1}^{(m)}, y_{1:k}) \cdot p(y_k|y_{1:k-1})}, \end{aligned} \tag{3.20}$$

en omdat $p(y_k|y_{1:k-1})$ onafhankelijk is van v_k geldt:

$$w_k^{(m)} \propto w_{k-1}^{(m)} \cdot \frac{p(y_k|v_{0:k}^{(m)}, y_{1:k-1}) \cdot p(v_k^{(m)}|v_{0:k-1}^{(m)}, y_{1:k-1})}{\pi(v_k^{(m)}|v_{0:k-1}^{(m)}, y_{1:k})}, \tag{3.21}$$

welke onze algemene 'weight update equation' is.

De importance function $\pi(v_k^{(m)}|v_{0:k-1}^{(m)}, y_{1:k})$ is vrij te kiezen. In eerste instantie nemen we $\pi(v_k^{(m)}|v_{0:k-1}^{(m)}, y_{1:k}) = p(v_k^{(m)}|v_{k-1}^{(m)}, y_k, y_{k-1})$, de in de theorie genoemde 'optimal importance function' (Petar M. Djurić et al. [6]). Later zullen we nog andere importance functions in ons model implementeren en onderzoeken wat hier de effecten van zijn, dit doen we in paragraaf 3.3. Tevens weten we uit de discretisatie van ons model (zie vergelijkingen (2.4) en (2.10)) dat v_k enkel direct afhankelijk is van v_{k-1} (en dus niet van meerdere, voorgaande termen van v) en dat y_k enkel direct afhankelijk is van v_k, y_{k-1} en v_{k-1} (en dus niet van meerdere, voorgaande termen van v en y). Met deze informatie en de gestelde importance function passen we onze weight update equation (3.21) aan tot:

$$w_k^{(m)} \propto w_{k-1}^{(m)} \cdot \frac{p(y_k|v_k^{(m)}, v_{k-1}^{(m)}, y_{k-1}) \cdot p(v_k^{(m)}|v_{k-1}^{(m)})}{p(v_k^{(m)}|v_{k-1}^{(m)}, y_k, y_{k-1})}, \tag{3.22}$$

welke de uitdrukking voor de gewichten is die we in eerste instantie bij het schatten van de volatiliteit zullen gebruiken. Om deze te kunnen berekenen hebben we echter nog wel uitdrukkingen voor de kansdichtheden nodig die in deze uitdrukking van de gewichten gebruikt worden.

Allereerst gaan we op zoek naar een uitdrukking voor $p(y_k|v_k^{(m)}, v_{k-1}^{(m)}, y_{k-1})$. We weten uit vergelijking (2.10) dat:

$$y_k = y_{k-1} + \mu\Delta t - \frac{1}{4}(v_k + v_{k-1})\Delta t + \frac{\rho}{\xi}(v_k - v_{k-1} - \kappa\theta\Delta t + \frac{1}{2}\kappa(v_k + v_{k-1})\Delta t) + \sqrt{1-\rho^2} \cdot \zeta_1. \tag{3.23}$$

Hier is $\zeta_1 \sim \mathcal{N}(0, \sigma^2)$ met σ^2 zoals in (2.9): $\sigma^2 = \frac{1}{2}(v_k + v_{k-1})\Delta t$. Hiermee kunnen we de verdeling van $p(y_k|v_k^{(m)}, v_{k-1}^{(m)}, y_{k-1})$ afleiden:

$$p(y_k|v_{k-1}, v_k, y_{k-1}) \sim \mathcal{N}(\mu_{c1}, \sigma_{c1}^2), \tag{3.24}$$

met de volgende μ_{c1} en σ_{c1}^2:

$$\begin{aligned} \mu_{c1} &= y_{k-1} + \mu\Delta t - \frac{1}{4}(v_k + v_{k-1})\Delta t + \frac{\rho}{\xi}(v_k - v_{k-1} - \kappa\theta\Delta t + \frac{1}{2}\kappa(v_k + v_{k-1})\Delta t), \\ \sigma_{c1}^2 &= \frac{1}{2}(v_k + v_{k-1})\Delta t \cdot (1 - \rho^2). \end{aligned} \tag{3.25}$$

Omdat (3.24) normaal verdeeld is met bekende parameters wordt de kansdichtheid gegeven door:

$$p(y_k|v_{k-1}^{(m)}, v_k^{(m)}, y_{k-1}) = \frac{1}{\sqrt{2\pi\sigma_{c1}^2}} \cdot e^{-\frac{1}{2}(\frac{y_k - \mu_{c1}}{\sigma_{c1}})^2}. \tag{3.26}$$

Vervolgens zoeken we een uitdrukking voor $p(v_k^{(m)}|v_{k-1}^{(m)})$. We weten uit (2.4) dat:

$$p(v_k^{(m)}|v_{k-1}^{(m)}) \sim C_1 \cdot \chi_d^2(\lambda). \tag{3.27}$$

De parameters zijn hetzelfde als in (2.4). We weten nu hoe $p(v_k|v_{k-1})$ verdeeld is en bekijken we zijn kansdichtheid om te gebruiken in de berekening van de gewichten $w_k^{(m)}$ zoals in (3.22). We kennen de

kansdichtheid van de non-centrale χ^2-verdeling en gebruiken deze bij het vinden van de kansdichtheid die we zoeken. We gebruiken de substitutie $Z = C \cdot X$, met $C > 0$ een constante:

$$F_Z(z) = P(Z \leq z) = P(C \cdot X \leq z) = P(X \leq \frac{z}{C}) = F_X(\frac{z}{C}), \tag{3.28}$$

waarbij F de verdelingsfunctie van de non-centrale χ^2-verdeling is. Hieruit weten we dat:

$$f_Z(z) = \frac{\partial}{\partial z} F_Z(z) = \frac{\partial}{\partial z} F_X(\frac{z}{C}) = \frac{1}{C} f_X(\frac{z}{C}), \tag{3.29}$$

waarbij f de kansdichtheid is van de non-centrale χ^2-verdeling is. Hieruit volgt het volgende:

$$p(v_k|v_{k-1}) = f_Z(v_k) = \frac{1}{C_1} f_X(\frac{v_k}{C_1}), \tag{3.30}$$

waarbij $f_X(\cdot)$ gelijk is aan de kansdichtheid van $\chi^2_d(\lambda)$, welke gegeven wordt door:

$$f_X(x; k; \lambda) = \sum_{i=0}^{\infty} \frac{e^{-\lambda/2}(\lambda/2)^i}{i!} f_{Y_{k+2i}}(x), \tag{3.31}$$

waarbij Y_q χ^2-verdeeld is met q vrijheidsgraden.

Tot slot zoeken we een uitdrukking voor $p(v_k^{(m)}|v_{k-1}^{(m)}, y_k, y_{k-1})$. Voor deze uitdrukking schrijven we eerst ons originele model op een soortgelijke manier als bij vergelijkingen (2.1) en (2.2) om naar:

$$dy_t = \left(\mu - \frac{1}{2} v_t\right) dt + \sqrt{v_t} dB_t, \tag{3.32}$$

$$dv_t = \kappa(\theta - v_t)dt + \xi\sqrt{v_t}\left[\rho dB_t + \sqrt{1-\rho^2} d\widetilde{Z_t}\right]. \tag{3.33}$$

In deze vorm van het model zijn B_t en $\widetilde{Z_t}$ onafhankelijke Brownse bewegingen. We schrijven (3.32) om tot het volgende:

$$\sqrt{v_t} dB_t = dy_t - \left(\mu - \frac{1}{2} v_t\right) dt. \tag{3.34}$$

Vervolgens willen we vergelijking (3.33) discretiseren, hiertoe schrijven we hem eerst om met behulp van (3.34):

$$\begin{aligned} dv_t &= \kappa(\theta - v_t)dt + \xi\sqrt{v_t}\left[\rho dB_t + \sqrt{1-\rho^2} d\widetilde{Z_t}\right] \\ &= \kappa(\theta - v_t)dt + \xi\rho\sqrt{v_t} dB_t + \xi\sqrt{1-\rho^2}\sqrt{v_t} d\widetilde{Z_t} \\ &= \kappa(\theta - v_t)dt + \xi\rho\left[dy_t - \left(\mu - \frac{1}{2} v_t\right) dt\right] + \xi\sqrt{1-\rho^2}\sqrt{v_t} d\widetilde{Z_t}. \end{aligned} \tag{3.35}$$

Discretiseren van (3.35) doen we als volgt, voor $t_k > t_{k-1}$, $\Delta t = t_k - t_{k-1}$ en $k = 0, 1, 2, \ldots$:

$$v_k - v_{k-1} = \kappa(\theta - v_{k-1})\Delta t + \xi\rho\left[(y_k - y_{k-1}) - (\mu - \frac{1}{2} v_{k-1})\Delta t\right] + \xi\sqrt{1-\rho^2}\sqrt{v_{k-1}}(\widetilde{Z_k} - \widetilde{Z_{k-1}}), \tag{3.36}$$

wat neerkomt op het volgende:

$$v_k = v_{k-1} + \kappa(\theta - v_{k-1})\Delta t + \xi\rho(y_k - y_{k-1}) - \xi\rho(\mu - \frac{1}{2} v_{k-1})\Delta t + \xi\sqrt{1-\rho^2}\sqrt{v_{k-1}}(\widetilde{Z_k} - \widetilde{Z_{k-1}}). \tag{3.37}$$

Uit de definitie van een Brownse beweging weten we dat $(\widetilde{Z_k} - \widetilde{Z_{k-1}}) \sim \mathcal{N}(0, \Delta t)$. We zien in vergelijking (3.37) dat v_k slechts afhankelijk is van v_{k-1}, y_k en y_{k-1} en dat deze als volgt verdeeld is:

$$p(v_k|v_{k-1}, y_k, y_{k-1}) \sim \mathcal{N}(\mu_{c2}, \sigma^2_{c2}), \tag{3.38}$$

met de volgende μ_{c2} en σ^2_{c2}:

$$\begin{aligned} \mu_{c2} &= v_{k-1} + \kappa(\theta - v_{k-1})dt + \xi\rho(y_k - y_{k-1}) - \xi\rho(\mu - \frac{1}{2} v_{k-1})\Delta t, \\ \sigma^2_{c2} &= \xi^2(1-\rho^2) v_{k-1}\Delta t, \end{aligned} \tag{3.39}$$

De kansdichtheid wordt dan als volgt gegeven:

$$p(v_k^{(m)}|v_{k-1}^{(m)}, y_k, y_{k-1}) = \frac{1}{\sqrt{2\pi\sigma_{c2}^2}} \cdot e^{\frac{1}{2}(\frac{v_k - \mu_{c2}}{\sigma_{c2}})^2}. \tag{3.40}$$

Nu we voor iedere kansdichtheid in onze weight update equation (3.21) een uitdrukking hebben, kunnen we het particle filter implementeren. We zullen een markt genereren met de theorie van hoofdstuk 2 en met het in dit hoofdstuk 3 behandelde particle filter schatten we de volatiliteit daarvan op ieder tijdstip k volgens (3.16). In hoofdstuk 5 leggen we ons gebruikte algoritme bij deze simulaties uit en in hoofdstuk 6 behandelen we de resultaten van onze simulaties. Ook willen we het effect bekijken van het gebruiken van andere importance functions en andere resamplemethoden. Hierover vertellen we in de paragrafen 3.3 en 3.4 meer.

3.3 Verschillende importance functions

In de voorgaande paragraaf hebben we een aantal aannamen moeten maken om implementatie mogelijk te maken; zo hebben we de kansdichtheden uit de formule voor de gewichten moeten benaderen, hebben we een importance function gekozen en hebben we een resamplemethode gekozen. Natuurlijk zijn er voor de keuzes die we gemaakt hebben ook alternatieven, die wellicht positief uit kunnen pakken op onze resultaten. In deze paragraaf gaan we in op de verschillende importance functions die we kunnen kiezen om te gebruiken.

3.3.1 Optimal importance function

In de voorgaande paragraaf hebben we de zogenaamde 'optimal importance function' ($\pi(v_k^{(m)}|v_{0:k-1}^{(m)}, y_{1:k}) = p(v_k^{(m)}|v_{k-1}^{(m)}, y_k, y_{k-1})$) gebruikt en hebben we laten zien dat $p(v_k^{(m)}|v_{k-1}^{(m)}, y_k, y_{k-1})$ te benaderen is met een normale verdeling. Het nadeel aan deze benadering is dat de normale verdeling ook negatieve waarden aan kan nemen; iets wat in de praktijksituatie voor de volatiliteit niet voor kan komen, aangezien deze altijd positief is. We kunnen deze kansdichtheid ook benaderen met de non-centrale χ^2-verdeling, zoals we $p(v_k^{(m)}|v_{k-1}^{(m)})$ ook benaderd hebben, welke geen negatieve waarden aan kan nemen. Deze benadering leiden we als volgt af:

We weten uit (2.3) en (2.4) dat als:

$$v_k = v_{k-1} + \int_{t_{k-1}}^{t_k} \kappa(\theta - v_s)\,\mathrm{d}s + \int_{t_{k-1}}^{t_k} \xi\sqrt{v_s}\,\mathrm{d}Z_s, \tag{3.41}$$

dat dan:

$$v_k|v_{k-1} \sim C \cdot \chi_d^2(\lambda), \tag{3.42}$$

waarbij

$$C = \frac{\xi^2(1 - e^{-\kappa\Delta t})}{4\kappa}, \quad d = \frac{4\theta\kappa}{\xi^2}, \quad \lambda = \frac{4\kappa e^{-\kappa\Delta t}}{\xi^2(1 - e^{-\kappa\Delta t})} v_{k-1}. \tag{3.43}$$

Als we (3.35) discretiseren volgt daarbij de volgende uitdrukking voor v_k, met daarin nog continue termen:

$$v_k = v_{k-1} + \int_{t_{k-1}}^{t_k} \kappa(\theta - v_s)\,\mathrm{d}s - \int_{t_{k-1}}^{t_k} \xi\rho(\mu - \frac{1}{2}v_s)\,\mathrm{d}s + \xi\rho(y_k - y_{k-1}) + \int_{t_{k-1}}^{t_k} \xi\sqrt{1-\rho^2}\sqrt{v_s}\,\mathrm{d}\widetilde{Z_s}. \tag{3.44}$$

Deze uitdrukking willen we omschrijven naar de vorm van vergelijking (3.41), want dan kunnen we concluderen dat $v_k|v_{k-1}$ non-centraal χ^2-verdeeld is. Na de substitutie $v_{k-1}^* = v_{k-1} + \xi\rho(y_k - y_{k-1})$ wordt (3.44):

$$\begin{aligned} v_k = v_{k-1}^* &+ \int_{t_{k-1}}^{t_k} \kappa(\theta - v_s)\,\mathrm{d}s - \int_{t_{k-1}}^{t_k} \xi\rho(\mu - \frac{1}{2}v_s)\,\mathrm{d}s \\ &+ \int_{t_{k-1}}^{t_k} \xi\sqrt{1-\rho^2}\sqrt{v_s}\,\mathrm{d}\widetilde{Z_s}. \end{aligned} \tag{3.45}$$

Omschrijven geeft:

$$\begin{aligned} v_k &= v_{k-1}^* + \int_{t_{k-1}}^{t_k} \left(\kappa(\theta - v_s) - \xi\rho(\mu - \frac{1}{2}v_s)\right) \mathrm{d}s \quad + \int_{t_{k-1}}^{t_k} \xi\sqrt{1-\rho^2}\sqrt{v_s}\,\mathrm{d}\widetilde{Z_s} \\ &= v_{k-1}^* + \int_{t_{k-1}}^{t_k} \left(\kappa\theta - \xi\rho\mu + (\frac{1}{2}\xi\rho - \kappa)v_s\right) \mathrm{d}s \quad + \int_{t_{k-1}}^{t_k} \xi\sqrt{1-\rho^2}\sqrt{v_s}\,\mathrm{d}\widetilde{Z_s}. \end{aligned} \tag{3.46}$$

Na de substitutie $\xi^* = \xi\sqrt{1-\rho^2}$ volgt:

$$v_k = v_{k-1}^* + \int_{t_{k-1}}^{t_k} \left(\kappa\theta - \xi\rho\mu + (\frac{1}{2}\xi\rho - \kappa)v_s\right) \mathrm{d}s + \int_{t_{k-1}}^{t_k} \xi^*\sqrt{v_s}\,\mathrm{d}\widetilde{Z_s}. \tag{3.47}$$

Deze uitdrukking is al bijna van dezelfde vorm als (3.41). Om hem op deze vorm te krijgen moeten we $\int_{t_{k-1}}^{t_k} \kappa\theta - \xi\rho\mu + (\frac{1}{2}\xi\rho - \kappa)v_s\,\mathrm{d}s$ omschrijven naar $\int_{t_{k-1}}^{t_k} \kappa^*(\theta^* - v_s)\,\mathrm{d}s$. Hieruit blijkt dat $-\kappa^* v_s = (\frac{1}{2}\xi\rho - \kappa)v_s$ en $\kappa^*\theta^* = \kappa\theta - \xi\rho\mu$. Hieruit volgt dat:

$$\kappa^* = -(\frac{1}{2}\xi\rho - \kappa) \quad \text{en} \quad \theta^* = \frac{\kappa\theta - \xi\rho\mu}{\kappa^*} = \frac{\kappa\theta - \xi\rho\mu}{-(\frac{1}{2}\xi\rho - \kappa)}. \tag{3.48}$$

Hiermee schrijven we vergelijking (3.47) om naar:

$$v_k = v_{k-1}^* + \int_{t_{k-1}}^{t_k} \kappa^*(\theta^* - v_s)\,\mathrm{d}s + \int_{t_{k-1}}^{t_k} \xi^*\sqrt{v_s}\,\mathrm{d}\widetilde{Z_s}, \tag{3.49}$$

welke dezelfde vorm heeft als vergelijking (3.41). Hierdoor weten we dat:

$$v_k | v_{k-1}^* \sim C^* \cdot \chi^2_{d^*}(\lambda^*), \tag{3.50}$$

met:

$$C^* = \frac{{\xi^*}^2(1 - e^{-\kappa^*\Delta t})}{4\kappa^*}, \quad d^* = \frac{4\theta^*\kappa^*}{{\xi^*}^2}, \quad \lambda^* = \frac{4\kappa^* e^{-\kappa^*\Delta t}}{{\xi^*}^2(1 - e^{-\kappa^*\Delta t})} v_{k-1}^*. \tag{3.51}$$

We krijgen met behulp van (3.30):

$$p(v_k^{(m)} | v_{k-1}^{(m)}, y_k, y_{k-1}) = \frac{1}{C^*} f(\frac{v_k^{(m)}}{C^*}), \tag{3.52}$$

waarbij $f(\cdot)$ de kansdichtheid is van $\chi^2_{d^*}(\lambda^*)$, zoals beschreven in (3.31).

We hebben $p(v_k^{(m)} | v_{k-1}^{(m)}, y_k, y_{k-1})$ nu benaderd met de non-centrale χ^2-verdeling. Deze benadering kunnen we in ons programma doorvoeren en vervolgens onderzoeken of de resultaten met deze bandering beter zijn dan met de normale verdeling.

3.3.2 Suboptimal importance function

Nu hebben we voor de 'optimal importance function' twee keuzes, maar er is naast deze twee keuzes nog een andere keuze voor de importance function te vinden in de literatuur: de 'suboptimal importance function', welke we kennen als:

$$\pi(v_k^{(m)} | v_{0:k-1}^{(m)}, y_{1:k}) = p(v_k^{(m)} | v_{k-1}^{(m)}). \tag{3.53}$$

Het verschil met de 'optimal importance function' is duidelijk; de afhankelijkheid van y ontbreekt in deze importance function. Omdat deze importance function een andere vorm heeft dan de 'optimal importance function' zal de weight update equation van het particle filter veranderen: we vullen (3.53) in de weight update equation (3.21) in en krijgen:

$$w_k^{(m)} \propto w_{k-1}^{(m)} \cdot \frac{p(y_k | v_k^{(m)}, v_{k-1}^{(m)}, y_{k-1}) p(v_k^{(m)} | v_{k-1}^{(m)})}{p(v_k^{(m)} | v_{k-1}^{(m)})} = w_{k-1}^{(m)} \cdot p(y_k | v_k^{(m)}, v_{k-1}^{(m)}, y_{k-1}). \tag{3.54}$$

We zien dat de term $p(v_k^{(m)} | v_{k-1}^{(m)})$ zowel boven als onder de deelstreep voorkomt, deze strepen we dus tegen elkaar weg. De kansdichtheid die overblijft is $p(y_k | v_k^{(m)}, v_{k-1}^{(m)}, y_{k-1})$, waarvan we in de voorgaande paragraaf

hebben laten zien met vergelijking (3.23) dat deze normaal verdeeld is en dus uit te drukken is als in (3.26) met μ en σ^2 als in (3.25).

Met deze uitdrukkingen kunnen we de 'suboptimal importance function' doorvoeren in ons programma en de resultaten vergelijken met de twee benaderingen van de 'optimal importance function'.

Andere keuzes voor de gebruikte importance function laten we in ons onderzoek buiten beschouwing.

3.4 Verschillende resamplemethoden

Als de gewichten die gebruikt worden bij particle filtering in grootte teveel van elkaar gaan verschillen en sommige particles gewichten van bijna 0 hebben, dient er geresampled te worden. Een van de aannamen die we in paragraaf 3.2 gemaakt hebben, is de manier waarop we in eerste instantie resamplen. Deze methode heet 'Multinomial resampling', maar er zijn andere methoden die gebruikt kunnen worden. Van deze methoden zullen we er in deze paragraaf drie bespreken, daar te weten 'Residual resampling', 'Stratified resampling', 'Systematic resampling' en als eerste zullen we 'Multinomial resampling' nogmaals beknopt bespreken. Voordat we dit doen geven we eerst wat meer uitleg over de variabelen die we gebruiken: eerst noemen we de oude particles $v_{oud}^{(i)}$ en de nieuwe particles, dus na resampling, $v_{nieuw}^{(i)}$.

3.4.1 Multinomial Resampling

Multnomial resampling is de meest eenvoudige resampling methode. Eerst wordt er een cumulatieve array Q van de genormaliseerde gewichten w gemaakt, zodat $Q(0) = 0$, $Q(1) = w^{(1)}$, $Q(2) = w^{(1)} + w^{(2)}$ enzovoort, tot slot is $Q(N) = 1$. We trekken nu $T(i) \in \mathcal{U}(0, 1)$, waarbij $i = 1, \ldots, N$. Voor $j = 1, \ldots, N$ voeren we de volgende controle uit: als $Q(j-1) < T(i) \leq Q(j)$, dan wordt de waarde van het nieuwe particle $v_{nieuw}^{(i)} = v_{oud}^{(j)}$. Als alle particles getrokken zijn worden de gewichten $w^{(i)}$ allemaal op $\frac{1}{N}$ gezet.

3.4.2 Residual Resampling

Residual resampling verloopt in twee stappen: de deterministische en de stochastische stap. Als eerste moet de deterministische stap worden uitgevoerd. Hier berekent men het aantal keer dat een particle met zekerheid getrokken wordt met $\lfloor w^{(i)} \cdot N \rfloor$.

Het stochastische deel gaat als volgt: Stel dat uit de deterministische stap k particles getrokken zijn, dan moeten er nog $N-k$ particles getrokken worden. Bereken hiertoe $\tilde{w}^{(i)} = w^{(i)} - \lfloor w^{(i)} \cdot N \rfloor$ en tel deze kansen op in de vector Q zodat $Q(0) = 0$, $Q(1) = \tilde{w}^{(1)}$, enzovoort, tot $Q(N) = \sum_{i=1}^{N} \tilde{w}^{(i)}$. Trek nu $T(i) \in \mathcal{U}(0, Q(N))$, waarbij $i = 1, \ldots, N$. Voor $j = 1, \ldots, N$ voeren we de volgende controle uit: als $Q(j-1) < T(i) \leq Q(j)$, dan wordt de waarde van het nieuwe particle $v_{nieuw}^{(i)} = v_{oud}^{(j)}$. Als alle particles getrokken zijn worden de gewichten $w^{(i)}$ allemaal op $\frac{1}{N}$ gezet.

3.4.3 Stratified Resampling

Bij stratified resampling trekken we willekeurige waarden $T(i) \in \mathcal{U}(\frac{i-1}{N}, \frac{i}{N})$, waarbij $i = 1, \ldots, N$. We berekenen Q op dezelfde manier als bij multinomial resampling. Voor $j = 1, \ldots, N$ voeren we de volgende controle uit: als $Q(j-1) < T(i) \leq Q(j)$, dan wordt de waarde van het nieuwe particle $v_{nieuw}^{(i)} = v_{oud}^{(j)}$. Als alle particles getrokken zijn worden de gewichten $w^{(i)}$ allemaal op $\frac{1}{N}$ gezet.

3.4.4 Systematic Resampling

Systematic resampling lijkt op stratified resampling. We berekenen Q op dezelfde manier maar berekenen $T(i)$ anders. $T(1) \in \mathcal{U}(0, \frac{1}{N})$, waarna $T(i)$ als volgt wordt berekend:

$$T(i) = T(1) + \frac{1}{N}(i-1), \tag{3.55}$$

met $i = 2, \ldots, N$.
Nu gaan we voor $i = 1, \ldots, N$ en $j = 1, \ldots, N$ het volgende na: als $Q(j-1) < T(i) \leq Q(j)$, dan wordt de waarde van het nieuwe particle $v_{nieuw}^{(i)} = v_{oud}^{(j)}$. Als alle particles getrokken zijn worden de gewichten $w^{(i)}$

allemaal op $\frac{1}{N}$ gezet.

We hebben nu vier methoden beschreven om te resamplen. Deze methoden zullen we in ons programma implementeren en vervolgens zullen we onderzoeken of er een methode aan te wijzen is die de beste resultaten oplevert. Hierbij moet wel rekening gehouden worden met N_{eff}, welke bij overschrijding van de drempel N_{thr} aangeeft dat er geresampled moet worden. We zullen na het testen van de resamplemethoden ook testen in hoeverre de kwaliteit van de resultaten afhangt van deze drempel.

Hoofdstuk 4

Model met onbekende parameters

Tot nu toe hebben we in ons model de parameters μ, θ, κ, ξ en ρ als bekend verondersteld. In de praktijk zijn deze parameters niet bekend en is er geen analytische formule om deze uit te rekenen. Als we met ons model de volatiliteit van een echt aandeel willen schatten, zullen we ons model aan moeten passen op een manier dat μ, θ, κ, ξ en ρ niet meer bekend verondersteld hoeven te worden. In dit hoofdstuk zullen we eerst een nieuw model afleiden, waarin de parameters μ, θ, κ, ξ en ρ net als de volatiliteit geschat worden met behulp van een particle filter. Vervolgens zullen we de toepassingen van dit model op de aandelenmarkt behandelen.

4.1 Het model met onbekende parameters opstellen

Bij het opstellen van het model met onbekende parameters zullen we het artikel van Shin Ichi Aihara et al. [1] volgen. We gaan uit van ons gediscretiseerde model, zoals beschreven in vergelijkingen (2.4) en (2.10). Dit model herschrijven we naar een nieuw discreet model, waar we vervolgens een particle filter op kunnen toepassen zodat naast de volatiliteit ook de parameters van het model geschat worden.

Allereerst construeren we een vector $z_k = [v_k, \alpha_k]$ met $\alpha_k = [\kappa_k \ \theta_k \ \mu_k \ \xi_k \ \rho_k]$. Hierin veronderstellen we niet meer dat de parameters bekend zijn en nemen we ze mee in de toestandsrepresentatie van ons model, *gegeven* α_{k-1}, als zijnde:

$$\begin{aligned}
\kappa_k &\sim \mathcal{N}(\kappa_{k-1}, \epsilon_1), \\
\theta_k &\sim \mathcal{N}(\theta_{k-1}, \epsilon_2), \\
\mu_k &\sim \mathcal{N}(\mu_{k-1}, \epsilon_3), \\
\xi_k &\sim \mathcal{N}(\xi_{k-1}, \epsilon_4), \\
\rho_k &\sim \mathcal{N}(\rho_{k-1}, \epsilon_5).
\end{aligned} \tag{4.1}$$

Of analoog:

$$\begin{aligned}
\kappa_k &= \kappa_{k-1} + \epsilon_{1k}, \ \epsilon_{1k} \sim \mathcal{N}(0, \epsilon_1), \\
\theta_k &= \theta_{k-1} + \epsilon_{2k}, \ \epsilon_{2k} \sim \mathcal{N}(0, \epsilon_2), \\
\mu_k &= \mu_{k-1} + \epsilon_{3k}, \ \epsilon_{3k} \sim \mathcal{N}(0, \epsilon_3), \\
\xi_k &= \xi_{k-1} + \epsilon_{4k}, \ \epsilon_{4k} \sim \mathcal{N}(0, \epsilon_4), \\
\rho_k &= \rho_{k-1} + \epsilon_{5k}, \ \epsilon_{5k} \sim \mathcal{N}(0, \epsilon_5),
\end{aligned} \tag{4.2}$$

met:

$$\begin{aligned}
\kappa_0 &\sim \mathcal{U}(L_\kappa, R_\kappa), \\
\theta_0 &\sim \mathcal{U}(L_\theta, R_\theta), \\
\mu_0 &\sim \mathcal{U}(L_\mu, R_\mu), \\
\xi_0 &\sim \mathcal{U}(L_\xi, R_\xi), \\
\rho_0 &\sim \mathcal{U}(L_\rho, R_\rho).
\end{aligned} \tag{4.3}$$

We merken op dat α_k slechts afhankelijk is van α_{k-1} en dat op deze manier de parameters ook niet meer constant zijn, maar iedere tijdstap opnieuw geschat worden. Dit doen we door ze mee te nemen in het particle filter voor het schatten van de volatiliteit. We trekken de beginwaarden van de elementen van α uniform binnen grenzen die hoogstwaarschijnlijk de echte waarden van de elementen van α zullen omvatten.

Als we ons gediscretiseerde model herschrijven met de elementen van α krijgen we het volgende:

$$\begin{aligned} y_k = y_{k-1} + \mu_k \Delta t - \frac{1}{4}(v_k + v_{k-1})\Delta t \\ + \frac{\rho_k}{\xi_k}\left(v_k - v_{k-1} - \kappa_k\theta_k\Delta t + \frac{\kappa_k}{2}(v_k + v_{k-1})\Delta t\right) + \sqrt{1-\rho_k^2}\cdot\zeta_k, \end{aligned} \tag{4.4}$$

$$v_k|v_{k-1} \sim C_k \cdot \chi^2_{d_k}(\lambda_k), \tag{4.5}$$

waarbij ζ_k een $\mathcal{N}(0,\sigma^2)$-verdeelde stochatische variabele is met σ^2 zoals in (2.9) en:

$$C_k = \frac{\xi_k^2\left(1-e^{-\kappa_k\Delta t}\right)}{4\kappa_k}, \quad \lambda_k = \frac{4\kappa_k e^{-\kappa_k\Delta t}}{\xi_k^2\left(1-e^{-\kappa_k\Delta t}\right)}\cdot v_{k-1}, \quad d_k = \frac{4\theta_k\kappa_k}{\xi_k^2}, \tag{4.6}$$

waarbij $\chi^2_{d_k}(\lambda_k)$ een stochastische variabele is die non-centraal χ^2-verdeeld is met d_k vrijheidsgraden en parameter λ_k, zoals in (3.31).

Vervolgens willen we met dit nieuwe model de volatiliteit en de parameters, dus z_k, gaan schatten middels particle filtering. Dit doen we net als in (3.6):

$$\hat{\mathbb{E}}_{p(z_k|y_{1:k},z_{k-1})}(z_k) = \sum_m^M w_k^{(m)} z_k^{(m)}. \tag{4.7}$$

En we weten dat:

$$\hat{\mathbb{E}}_{p(z_k|y_{1:k},z_{k-1})}(z_k) \longrightarrow \mathbb{E}_{p(z_k|y_{1:k},z_{k-1})}(z_k) \tag{4.8}$$

bijna zeker als $M \longrightarrow \infty$.

Voor dit model moeten we dus het particle filter herontwerpen. We vullen z_k in plaats van v_k in de weight update equation (3.21) in en we gebruiken als importance function $\pi(z_k^{(m)}|z_{0:k-1}^{(m)}, y_{1:k}) = p(z_k^{(m)}|z_{k-1}^{(m)}, y_k, y_{k-1})$ en verkrijgen:

$$w_k^{(m)} \propto w_{k-1}^{(m)} \cdot \frac{p(y_k|z_k^{(m)}, z_{k-1}^{(m)}, y_{k-1})p(z_k^{(m)}|z_{k-1}^{(m)})}{p(z_k^{(m)}|z_{k-1}^{(m)}, y_k, y_{k-1})}. \tag{4.9}$$

Om de kansdichtheden te berekenen in deze vergelijking, zullen we z_k weer gaan splitsen in zijn componenten v_k en α_k en verkrijgen we:

$$w_k^{(m)} \propto w_{k-1}^{(m)} \cdot \frac{p(y_k|v_k^{(m)}, v_{k-1}^{(m)}, y_{k-1}, \alpha_k^{(m)}, \alpha_{k-1}^{(m)})p(v_k^{(m)}, \alpha_k^{(m)}|v_{k-1}^{(m)}, \alpha_{k-1}^{(m)})}{p(v_k^{(m)}, \alpha_k^{(m)}|v_{k-1}^{(m)}, y_k, y_{k-1}, \alpha_{k-1}^{(m)})} \tag{4.10}$$

We constateren uit (4.4) dat y_k niet afhankelijk is van α_{k-1} en kunnen deze term in de afhankelijkheid dus wegstrepen, wat resulteert in de kansdichtheid $p(y_k|v_k^{(m)}, v_{k-1}^{(m)}, y_{k-1}, \alpha_k^{(m)})$, welke we normaal kunnen benaderen zoals we dit voorheen ook gedaan hebben.

Vervolgens schrijven we de twee andere kansdichtheden uit, met de kennis dat alle elementen in α_k slechts afhankelijk zijn van hun voorganger in α_{k-1} (uit (4.1)) en dat v_k niet direct afhankelijk is van α_{k-1} (uit (4.5)):

$$\begin{aligned} &p(v_k, \alpha_k|v_{k-1}, \alpha_{k-1}) = p(v_k, \kappa_k, \theta_k, \mu_k, \xi_k, \rho_k|v_{k-1}, \kappa_{k-1}, \theta_{k-1}, \mu_{k-1}, \xi_{k-1}, \rho_{k-1}) \\ &= p(\kappa_k|v_{k-1}, \alpha_{k-1})p(v_k, \theta_k, \mu_k, \xi_k, \rho_k|\kappa_k, v_{k-1}, \alpha_{k-1}) \\ &= p(\kappa_k|\kappa_{k-1})p(\theta_k|\kappa_k, v_{k-1}, \alpha_{k-1})p(v_k, \mu_k, \xi_k, \rho_k|\kappa_k, \theta_k, v_{k-1}, \alpha_{k-1}) \\ &= p(\kappa_k|\kappa_{k-1})p(\theta_k|\kappa_k, \theta_{k-1})p(\mu_k|\kappa_k, \theta_k, v_{k-1}, \alpha_{k-1})p(v_k, \xi_k, \rho_k|\kappa_k, \theta_k, \mu_k, v_{k-1}, \alpha_{k-1}), \end{aligned} \tag{4.11}$$

We zien bij het uitwerken dat er een afhankelijkheid van κ_k verschijnt in de term $p(\theta_k|\kappa_k, \alpha_{k-1})$ en een afhankelijkheid van θ_k en κ_k in de term $p(\mu_k|\kappa_k, \theta_k, v_{k-1}, \alpha_{k-1})$. We weten echter dat μ_k enkel afhankelijk is van μ_{k-1} en dat θ_k enkel afhankelijk is van θ_{k-1}, zoals ieder element van α_k enkel afhankelijk is van zijn eigen voorganger. De niet bestaande afhankelijkheden in de uitdrukkingen zullen we in het vervolg direct wegstrepen:

$$\begin{aligned} &= p(\kappa_k|\kappa_{k-1})p(\theta_k|\theta_{k-1})p(\mu_k|\mu_k)p(\xi_k|v_{k-1}, \alpha_{k-1})p(v_k, \rho_k|\kappa_k, \theta_k, \mu_k, \xi_k, v_{k-1}, \alpha_{k-1}) \\ &= p(\kappa_k|\kappa_{k-1})p(\theta_k|\theta_{k-1})p(\mu_k|\mu_k)p(\xi_k|\xi_k)p(\rho_k|v_{k-1}, \alpha_{k-1})p(v_k|\alpha_k, v_{k-1}, \alpha_{k-1}) \\ &= p(\kappa_k|\kappa_{k-1})p(\theta_k|\theta_{k-1})p(\mu_k|\mu_k)p(\xi_k|\xi_k)p(\rho_k|\rho_{k-1})p(v_k|v_{k-1}, \alpha_k). \end{aligned} \tag{4.12}$$

Op een soortgelijke manier van splitsen krijgen we:

$$\begin{aligned} &p(v_k, \alpha_k | v_{k-1}, y_k, y_{k-1}, \alpha_{k-1}) \\ &= p(\kappa_k|\kappa_{k-1}, y_k, y_{k-1}) p(\theta_k|\theta_{k-1}, y_k, y_{k-1}) p(\mu_k|\mu_{k-1}, y_k, y_{k-1}) p(\xi_k|\xi_{k-1}, y_k, y_{k-1}) \\ &\cdot p(\rho_k|\rho_{k-1}, y_k, y_{k-1}) p(v_k|v_{k-1}, y_k, y_{k-1}, \alpha_k). \end{aligned} \tag{4.13}$$

We zien dat y_k en y_{k-1} zich nog voordoen in termen als $p(\kappa_k|\kappa_{k-1}, y_k, y_{k-1})$. Aangezien we geen uitdrukkingen hebben om dit te berekenen en y_k wel gegevens heeft over α_k zullen we de aanname moeten maken dat we in deze uitdrukkingen de afhankelijkheid van y_k en y_{k-1} moeten verwaarlozen. We krijgen dan de volgende uitdrukking:

$$\begin{aligned} &p(v_k, \alpha_k | v_{k-1}, y_k, y_{k-1}, \alpha_{k-1}) \\ &= p(\kappa_k|\kappa_{k-1}) p(\theta_k|\theta_{k-1}) p(\mu_k|\mu_{k-1}) p(\xi_k|\xi_{k-1}) p(\rho_k|\rho_{k-1}) p(v_k|v_{k-1}, y_k, y_{k-1}, \alpha_k). \end{aligned} \tag{4.14}$$

Met het voorgaande wordt de weight update equation als volgt:

$$w_k^{(m)} \propto w_{k-1}^{(m)} \cdot \frac{p(y_k|v_k^{(m)}, v_{k-1}^{(m)}, y_{k-1}, \alpha_k^{(m)}) p(v_k^{(m)}|v_{k-1}^{(m)}, \alpha_k^{(m)})}{p(v_k^{(m)}|v_{k-1}^{(m)}, y_k, y_{k-1}, \alpha_k^{(m)})}. \tag{4.15}$$

In deze weight update equation staan alleen nog maar kansdichtheden waarvan we uit voorgaande hoofdstukken weten hoe we ze kunnen berekenen:

- $p(y_k|v_k^{(m)}, v_{k-1}^{(m)}, y_{k-1}, \alpha_k^{(m)})$ wordt gegeven door (3.26)
- $p(v_k^{(m)}|v_{k-1}^{(m)}, \alpha_k^{(m)})$ wordt gegeven door (3.30) en (3.31)
- $p(v_k^{(m)}|v_{k-1}^{(m)}, y_k, y_{k-1}, \alpha_k^{(m)})$ wordt gegeven door (3.40)

Bij deze gegevens moet er wel opgemerkt worden dat bij iedere gebruikte uitdrukking de parameters μ, θ, κ, ξ en ρ vervangen moeten worden door μ_k, θ_k, κ_k, ξ_k en ρ_k, om de uitdrukkingen aan te laten sluiten op dit model.

Nu we voor iedere kansdichtheid een expliciete uitdrukking hebben, hebben we alle benodigde gegevens om het model met onbekende parameters te programmeren en de toestand van z_k, dus de volatiliteit en de modelparameters, te schatten middels (4.7). In hoofdstuk 5 leggen we ons algoritme waarmee we deze schattingen maken uit.

4.2 Toepassingen op de beurs

Nu we een model hebben waarin de parameters μ, θ, κ, ξ en ρ niet meer als bekend verondersteld worden, kunnen we met dit model ook echte marktdata gaan analyseren. Bij onze voorgaande simulaties hebben we eerst zelf een volatiliteit en een markt gegenereerd, maar nu zullen we deze stap overslaan en als input voor ons model beursdata uit het verleden gebruiken. Aan de hand van deze beursdata kunnen we het verloop van de volatiliteit van specifieke aandelen schatten, evenals de verschillende onderliggende modelparameters. Hiertoe gebruiken we in ons programma ons gediscretiseerde model zoals gegeven in (4.4) en (4.5) en voor ons particle filter de daaruit afgeleide weight update equation (4.15) om middels (4.7) een schatting te maken.

In paragraaf 6.2.2 presenteren we een analyse van de volatiliteit van de AEX-index met ons model. Aangezien we de volatiliteit nu niet meer zelf genereren, kunnen we onze geschatte volatiliteit niet meer vergelijken met de 'echte' volatiliteit. Wel kunnen we de door ons geschatte volatiliteit vergelijken met een door iemand anders geschatte volatiliteit: in 2009 voerden Shin Ichi Aihara et al. [1] een soortgelijk onderzoek als ons uit en presenteerden een schatting van de volatiliteit van de AEX-index. In paragraaf 6.2.3 gebruiken wij in ons programma dezelfde startwaarden van ons model als dat zij deden en vergelijken we onze schatting van de volatiliteit met die van hun.

De resultaten van het gebruik van dit model op de aanedelenmarkt is waar we in ons onderzoek naar op zoek zijn: als de resultaten goed blijken te zijn, kan ons programma gebruikt worden om de volatiliteit van echte aandelen op de markt te schatten.

Hoofdstuk 5

Algoritmen

In dit hoofdstuk behandelen we de algoritmen die we geprogrammeerd hebben voor ons onderzoek. Als eerste zullen we stap voor stap uitleggen hoe we een markt genereren, waar we vervolgens de volatiliteit van gaan schatten. Hoe we de volatiliteit schatten leggen we uit in paragraaf 5.2 voor het model met bekende parameters. In paragraaf 5.3 gaan we in op het algoritme wat we gebruiken om met ons model met onbekende parameters de volatiliteit te schatten van een echt aandeel.
In appendix A staat de MATLAB-code die in de volgende paragrafen beschreven wordt.

5.1 Marktsimulatie

Stap 1 We definiëren κ, θ, μ, ρ en ξ. Verder definiëren we t als tijd waarover we gaan simuleren en Δt als stapgrootte.

Stap 2 Zet $T = t(\frac{1}{\Delta t}) + 1$, zet $\lambda_2 = \frac{4\kappa e^{-\kappa\Delta t}}{\xi^2(1-e^{-\kappa\Delta t})}$ en zet $d = \frac{4\theta\kappa}{\xi^2}$.

Stap 3 Maak en vector $v1$ en y aan, beide bevatten T elementen.

Stap 4 Definiëer $v1(1)$ als beginwaarde van de volatiliteit en $y(1)$ als beginwaarde van de logprijs van het aandeel als volgt: $v1(1) = \theta$ en $y(1) = 1$.

Stap 5 Zet $i = 2$.

Stap 6 Stel $\lambda = \lambda_2 \cdot v1(i-1)$, stel $v1(i) = \frac{\xi^2(1-e^{-\kappa\Delta t})}{4\kappa} \cdot \text{ncx2rnd}(d, \lambda)$ (hierin is ncx2rnd een functie in MATLAB welke een sample trekt uit de noncentrale χ^2-verdeling met parameters d en λ). Stel $\sigma_2 = \frac{v1(i)\Delta t}{2}$ en stel $y(i) = y(i-1) + \mu\Delta t - \frac{1}{4}(v1(i,1) + v1(i-1,1))\Delta t + \frac{\rho}{\xi}(v1(i,1) - v1(i-1,1)) - \kappa\theta\Delta t + \frac{1}{2}\kappa(v1(i,1) + v1(i-1,1))\Delta t + \sqrt{1-\rho^2}\mathcal{N}(0, \frac{1}{2}(v1(i,1) + v1(i-1,1)\Delta t)$.

Stap 7 Als $i < T$ verhogen we i met 1 en gaan we terug naar stap 6. Als $i = T$ dan zijn we klaar.

We hebben nu de volatiliteit $v1(i)$ en de logprijs $y(i)$ van de markt gesimuleerd, waarbij $i = 1, \ldots, T$

5.2 Volatiliteit schatten met gesimuleerde marktdata

In deze paragraaf lichten we ons algoritme toe wat de volatiliteit schat met het model met bekende parameters en hoe we dit in MATLAB geprogrammeerd hebben, hierbij maken we gebruik van de vector y uit paragraaf 5.1, bestaande uit T elementen, deze bevat namelijk de waarden van de logprijs van de markt.
Zoals beschreven staat in paragraaf 3.3 en 3.4 zijn er verschillende importance functions en resample methoden te kiezen, die elk een ander resultaat geven. In dit stappenplan is uitgegaan van de normale benadering voor de importance function, welke invloed heeft op de stappen 6 en 7 en de Multinomial Resampling methode, welke invloed heeft op stap 13. Als voor een andere importance function of resample methode gekozen wordt moeten deze stappen aangepast worden.
Om de fout van onze schatting van de volatiliteit te meten gebruiken we de vector $v1$ uit paragraaf 5.1, welke de echte waarden van de volatiliteit bevat, en we meten de fout van onze schattingen met behulp van de 'Root Mean Squared Error', ofwel de 'RMSE'. De RMSE is als volgt gedefiniëerd, voor $i = 1, 2, \ldots, T$:

$$\sqrt{\frac{1}{T}\sum_{i=1}^{T}(x_i - \widetilde{x_i})^2}, \tag{5.1}$$

waarbij x_i de schatting is van $\widetilde{x_i}$. In ons geval bekijken we dus de wortel van het gemiddelde kwadratische verschil van onze schatting van de volatiliteit op ieder tijdstip met zijn echte waarde als maatstaaf voor hoe goed onze schatting van de volatiliteit over het gehele tijdspan is.

Voor onze resultaten is de RMSE van eenmaal het programma draaien niet voldoende om conclusies te trekken, daarom willen we het programma meerdere keren draaien en kijken wat de gemiddelde RMSE over dat aantal keren draaien is. We berekenen:

$$RMSE_{av} = \frac{1}{M}\sum_{j=1}^{M} RMSE_j, \tag{5.2}$$

waarbij $RMSE_{av}$ het gemiddelde is van de RMSE van de schattingen van de volatiliteit van M maal het programma draaien.
In appendix A.1 staat vanaf regel 48 de code die hiervoor gebruikt is, welke we hieronder per stap zullen beschrijven.

Stap 1 Definiëer M als het aantal keer dat het gehele programma wordt doorlopen. Maak een vector RMSE aan met M elementen, die we op nul zetten. Zet l op 1.

Stap 2 Voer de marktsimulatie uit zoals beschreven in paragraaf 5.1 met de parameters κ, θ, μ, ξ, t, Δt en T. Definiëer N en N_{thr}. We maken de vector RMSE aan met M elementen, die we op nul zetten. Ook definiëren we het minimum van de volatiliteit, deze is v_{min}.

Stap 3 Stel de tijdsindex k gelijk aan 1. Maak een $2 \times N$ matrix en noem deze $v2$. Maak een $1 \times N$ matrix en noem deze $\tilde{w}$. Maak een $1 \times N$ matrix en noem deze w.

Stap 4 Vul rij 1 van $v2$ met nullen. De waarden van rij 2 worden getrokken uit $\mathcal{N}(\theta, \xi^2)$. Vul rij 1 van $\tilde{w}$ met nullen. Vul rij 1 van w met de waarde $\frac{1}{N}$ voor alle elementen.

Stap 5 Maak een vector V aan, bestaande uit T elementen met als eerste element $V(1) = \sum_{i=1}^{N} w(1,i) \cdot v2(2,i)$. Maak een getal $sumv_{squared}$ aan en zet deze op $(v1(1) - V(1))^2$.

Stap 6 Verhoog k met 1. Verplaats de tweede rij van $v2$ naar de eerste rij van $v2$. Trek de waarden van de tweede rij uit de normale verdeling $\mathcal{N}(\mu_c, \sigma_c^2)$, waarbij μ_c en σ_c^2 gegeven zijn door:

$$\mu_c = v2(1,i) + \kappa\left(\theta - v2(1,i)\right)(\Delta t) + \xi\rho(y(k) - y(k-1)) - \xi\rho\left[\mu - \frac{1}{2}v2(1,i)\right](\Delta t),$$

$$\sigma_c^2 = \xi^2(1-\rho^2)v2(1,i) \cdot \Delta t,$$

waarbij $i = 1, \ldots, N$. Als deze trekking een negatieve waarde geeft voor $v2(i)$ dan zetten we hem in plaats daarvan op v_{min}

Stap 7 Bereken $a_i = f(y(k))$ voor $i = 1, \ldots, N$, met $f(\cdot)$ de kansdichtheid van de normale verdeling met verwachting $\mu_{a,i}$ en variantie $\sigma_{a,i}^2$ als volgt gegeven:

$$\begin{aligned}\mu_{a,i} =& y(k-1) + \mu \cdot \Delta t - \frac{1}{4}\left(v2(2,i) + v2(1,i)\right) \cdot \Delta t \\ &+ \frac{\rho}{\xi}\left(v2(2,i) - v2(1,i) - \kappa\theta \cdot \Delta t + \kappa\frac{1}{2}(v2(2,i) + v2(1,i)) \cdot \Delta t\right),\end{aligned}$$

$$\sigma_{a,i}^2 = \frac{1}{2}\left(v2(2,i) + v2(1,i)\right) \cdot \Delta t \cdot (1-\rho^2).$$

Bereken $b_i = \frac{1}{C}g(\frac{v2(2,i)}{C})$ voor $i = 1, \ldots, N$, met $g(\cdot)$ de kansdichtheid van de noncentrale χ^2-verdeling met d vrijheidsgraden en parameter λ_i als volgt gegeven:

$$d = \frac{4\theta\kappa}{\xi^2}, \quad \lambda_i = \left(\frac{4\kappa e^{-\kappa\cdot\Delta t}}{\xi^2(1-e^{-\kappa\cdot\Delta t}}\right)v2(1,i), \quad \text{en} \quad C = \frac{\xi^2(1-e^{-\kappa\cdot\Delta t})}{4\kappa}. \tag{5.3}$$

Bereken $c_i = h(v(1,i))$ voor $i = 1, \ldots, N$, met $h(\cdot)$ de kansdichtheid van de normaal verdeelde benadering van de importance function met verwachting $\mu_{c,i}$ en variantie $\sigma^2_{c,i}$ als volgt gegeven:

$$\mu_{c,i} = v2(1,i) + \kappa(\theta - v2(1,i)) \cdot \Delta t + \xi\rho(y(k) - y(k-1)) - \xi\rho[\mu - \frac{1}{2}v2(1,i)] \cdot \Delta t, \tag{5.4}$$

$$\sigma^2_{c,i} = \xi^2(1-\rho^2)v2(1,i) \cdot \Delta t. \tag{5.5}$$

Stap 8 Bereken $\tilde{w}(1,i) = w(1,i) \cdot \frac{a_i \cdot b_i}{c_i}$ voor $i = 1, \ldots, N$.

Stap 9 Bereken $w(1,i) = \frac{\tilde{w}(1,i)}{\sum_{i=1}^{N} \tilde{w}(1,i)}$ voor $i = 1, \ldots, N$.

Stap 10 Stel $V(k)$ gelijk aan $\sum_{i=1}^{N} w(1,i) \cdot v2(2,i)$.

Stap 11 Verhoog $sumv_{squared}$ met $(v1(k) - V(k))^2$.

Stap 12 We berekenen $N_{eff} = \frac{1}{w \cdot w'}$. Wanneer $N_{eff} < N_{thr}$ is resampling nodig en gaan we naar stap 13, wanneer $N_{eff} \geq N_{thr}$ is dit niet het geval en gaan we naar stap 14.

Stap 13 Hier treedt het resampling proces op. De methode die hier is beschreven is multinomial resampling. We zien de matrix w als een matrix gevuld met kansen, deze tellen op tot 1. We gaan met behulp van de kansen $w(1,i)$ nieuwe waarden bepalen voor $v2(2,i)$. Dit doen we als volgt: Wordt het j-de element getrokken uit de matrix w (dit heeft kans $w(1,j)$), dan stellen we $v2(1,i)$ gelijk aan $v2(2,j)$, dit doen we voor $i = 1, \ldots, N$. Nu verplaatsen we de eerste rij van $v2$ naar de tweede rij van $v2$, hierbij overschrijven de oude waarden. Tot slot vervangen we alle waarden in de matrix w met $\frac{1}{N}$. Ga naar stap 14.

Stap 14 Is k gelijk aan T? Zo ja, ga dan naar stap 15. Zo nee, ga dan naar stap 6.

Stap 15 Stel het l-de element van $RMSE$ gelijk aan $\sqrt{\frac{sumv_{squared}}{T}}$. Als $l < M$, verhoog l dan met 1 en ga naar stap 2. Als $l = M$, ga dan naar stap 16.

Stap 16 Bereken de gemiddelde RMSE als volgt:

$$RMSE_{av} = \frac{1}{M}\sum_{l=1}^{M} RMSE(l). \tag{5.6}$$

We hebben nu de markt en een geschatte volatiliteit M keer berekend. Deze resultaten zijn niet bewaard, dat hoeft ook niet, want we zijn alleen geinteresseerd in het verschil met de werkelijke volatiliteit, oftewel, de RMSE. De gemiddelde RMSE van onze schattingen van de volatiliteit over M maal het programma draaien is gegeven in (5.6).

5.3 Volatiliteit schatten met echte marktdata

Het doel van ons onderzoek is om een schatting te geven van de volatiliteit van een aandeel uit de aandelenmarkt. We laden dus het verloop van de aandeelprijs van een echt bestaand aandeel over een bepaald tijdsinterval en schatten daar vervolgens de volatiliteit en de parameters van met ons model met onbekende paramaters.
Net zoals in paragraaf 5.2 zijn ook hier verschillende importance functions en resample methoden te kiezen, die elk een ander resultaat geven. In dit stappenplan is weer uitgegaan van de normale benadering voor de importance function, welke invloed heeft op de stappen 9 en 10 en de Multinomial Resampling methode, welke invloed heeft op de stappen 15 t/m 19. Als voor een andere importance function of resample methode gekozen wordt moeten deze stappen aangepast worden.
Eerst zal het verloop van de aandeelprijs gedownload moeten worden van internet, dit kan eenvoudig met MATLAB met de functie 'hist_stock_data.m', waaruit de aandelprijs van een specifiek aandeel op een specifiek interval wordt gegeven als een vector.

Eerst moet dit aandeel natuurlijk worden gedownload van een server, we gaan ervan uit dat dat hier al is gebeurd en dat de marktdata in de vorm staat waar t het aantal jaar is, $\Delta t = \frac{1}{252}$ en het aantal tijdstappen $T = \frac{t}{\Delta t}$. S_k is de aandeelprijs op tijdstip k, met $k = 1, \ldots, T$.

In appendix A.3 staat de code die hiervoor gebruikt is, welke we hieronder per stap zullen beschrijven.

Stap 1 Definiëer en bepaal alle preliminaries die nodig zijn: N als het aantal particles, N_{thr} als de drempel voor resampling, v een 2xN matrix met nullen, w een vector met lengte N gevuld met nullen, $\widetilde{w}$ een vector met lengte N gevuld met nullen en V een vector met lengte T gevuld met nullen. Definiëer ook μ_{min}, μ_{max}, θ_{min}, θ_{max}, κ_{min}, κ_{max}, ξ_{min}, ξ_{max} en ρ_{min}, ρ_{max} als onder- en bovengrenzen van de beginwaarden van de respectievelijke parameters. Definiëer tegelijk μ_{restr}, θ_{restr}, κ_{restr}, ξ_{restr} en ρ_{restr} als algemene ondergrens van de parameters. Definiëer ook σ_μ, σ_κ, σ_ξ, σ_θ en σ_ρ als middel om de parameters te variëren. Definiëer ten slotte v_{min} als de minimum volatiliteit.

Stap 2 Eerst moet de aandeelprijs omgevormd worden naar de logprijs waarmee we werken: $y_k = \ln(S_k)$ voor $k = 1, \ldots, T$.

Stap 3 In dit geval zijn de parameters onbekend, dus ook deze moeten we schatten. Definïeer α als een 5xN matrix. Iedere rij is een verzameling particles van een parameter, rij 1 is θ, rij 2 is ρ, rij drie is ξ, rij vier is κ en rij vijf is μ. Iedere parameter wordt N maal uniform getrokken tussen zijn onder- en bovengrenzen zoals in stap 1 gedefiniëerd en in α geplaatst.

Stap 4 Nu trekken we waarden voor N particles van de geschatte volatiliteit aan de hand van de zojuist geschatte parameters. Laat $j = 1, \ldots, N$ en trek v zodanig dat $v(2,j) \sim \mathcal{N}\left(\alpha(1,j), (\alpha(3,j))^2\right)$, waarbij $v(2,j) = v_{min}$ als hij kleiner zou zijn dan nul. Zet alle gewichten op $w(:) = \frac{1}{N}$ en zet de gemiddelde volatiliteit op $V(1) = \frac{sum(v(2,:))}{N}$.

Stap 5 Zet nu de tijdsindex i op 2.

Stap 6 Kopieer de tweede rij van v naar de eerste rij van v zodat $v(1,:) = v(2,:)$.

Stap 7 Zet de particle index j op 1.

Stap 8 Er worden nieuwe waarden voor α getrokken, waarbij $\alpha(1,j)$ wordt verhoogd met een willekeurige trekking uit $\mathcal{N}\left(0, \frac{\sigma_\theta}{\sqrt{i-1}}\right)$. Hetzelfde doen we voor $\alpha(2,j)$, welke we verhogen met $\mathcal{N}\left(0, \frac{\sigma_\rho}{\sqrt{i-1}}\right)$, $\alpha(3,j)$, verhogen we met $\mathcal{N}\left(0, \frac{\sigma_\xi}{\sqrt{i-1}}\right)$, $\alpha(4,j)$, verhogen we met $\mathcal{N}\left(0, \frac{\sigma_\kappa}{\sqrt{i-1}}\right)$, $\alpha(5,j)$, verhogen we met $\mathcal{N}\left(0, \frac{\sigma_\mu}{\sqrt{i-1}}\right)$. Hierbij worden de volgende randvoorwaarden gesteld: als $\alpha(1,j) < \theta_{restr}$, dan wordt $\alpha(1,j) = \theta_{restr}$, als $\alpha(3,j) < \xi_{restr}$, dan wordt $\alpha(3,j) = \xi_{restr}$, als $\alpha(2,j) > 1 - \rho_{restr}$, dan wordt $\alpha(2,j) = 1 - \rho_{restr}$, als $\alpha(3,j) < 1 + \rho_{restr}$, dan wordt $\alpha(3,j) = -1 + \rho_{restr}$ en als $\alpha(4,j) < \kappa_{restr}$, dan wordt $\alpha(4,j) = \kappa_{restr}$.

Stap 9 Trek nu een nieuw particle voor $v(2,j)$ uit $\mathcal{N}(\mu, \sigma^2)$. waarbij

$$\begin{aligned} \mu =& v(1,j) + \alpha(4,j)\left(\alpha(1,j) - v(1,j)\right)\Delta t + \alpha(3,j)\alpha(2,j)\cdot \\ & (y(i) - y(i-1)) - \alpha(3,j)\alpha(2,j)\left(\alpha(5,j) - \frac{1}{2}v(1,j)\right)\Delta t, \\ \sigma^2 =& \alpha(3,j)^2\left(1 - \alpha(2,j)^2\right)v(1,j)\Delta t, \end{aligned} \tag{5.7}$$

en als $v(2,j) < 0$, dan wordt $v(2,j) = v_{min}$.

Stap 10 Nu gaan we de gewichten bijstellen voor de nieuwe particles. Dit doen we door eerst de volgende berekeningen te maken:

$$\begin{aligned}
\mu_a =& y(i-1) + \alpha(5,j)\Delta t - \frac{1}{4}\left(v(2,j)+v(1,j)\right)\Delta t + \frac{\alpha(2,j)}{\alpha(3,j)}\cdot \\
& \left(v(2,j) - v(1,j) - \alpha(4,j)\alpha(1,j)\Delta t + \alpha(4,j)\frac{1}{2}\left(v(2,j)+v(1,j)\right)\Delta t\right), \\
\sigma_a =& \sqrt{\frac{1}{2}\left(v(2,j)+v(1,j)\right)\Delta t(1-\alpha(2,j)^2)}, \\
\mu_c =& v(1,j) + \alpha(4,j)(\alpha(1,j) - v(1,j))\Delta t + \alpha(3,j)\alpha(2,j)\cdot \\
& (y(i) - y(i-1)) - \alpha(3,j)\alpha(2,j)(\alpha(5,j) - \frac{1}{2}v(1,j))\Delta t, \\
\sigma_c =& \sqrt{\alpha(3,j)^2(1-\alpha(2,j)^2)v(1,j)\Delta t}, \\
C_b =& \frac{\alpha(3,j)^2\left(1-e^{-\alpha(4,j)\Delta t}\right)}{4\alpha(4,j)}, \\
d_b =& \frac{4\alpha(1,j)\alpha(4,j)}{\alpha(3,j)^2}, \\
\lambda_b =& \frac{4\alpha(4,j)e^{-\alpha(4,j)\Delta t}}{\alpha(3,j)^2\left(1-e^{-\alpha(4,j)\Delta t}\right)}\cdot v(1,j),
\end{aligned} \tag{5.8}$$

en deze vervolgens te gebruiken om de gewichten te berekenen op de volgende manier:

$$\begin{aligned}
a &= \text{normpdf}(y(i), \mu_a, \sigma_a), \\
b &= \frac{1}{C_b}\text{ncx2pdf}(\frac{v(2,j)}{C_b}, d_b, \lambda_b), \\
c &= \text{normpdf}(v(2,j), \mu_c, \sigma_c), \\
\widetilde{w(j)} &= w(j)\cdot\frac{ab}{c}.
\end{aligned} \tag{5.9}$$

Stap 11 Als j gelijk is aan N, ga dan naar stap 12, anders verhoog j met 1 en ga dan naar stap 8.

Stap 12 Nu normaliseren we de gewichten $\widetilde{w(j)}$. Hiertoe delen we ieder element uit $\widetilde{w(j)}$ door de som van alle elementen met de volgende formule:

$$w(j) = \frac{\widetilde{w(j)}}{sum(\widetilde{w(j)})}, \tag{5.10}$$

waarbij $j = 1, \ldots, N$.

Stap 13 We hebben nu N particles van de volatiliteit en hun bijbehorende gewichten. Deze combineren we tot een gemiddelde geschatte volatiliteit op tijdstap i:

$$V(i) = w \cdot v(2,:)'. \tag{5.11}$$

Ook van de parameters kunnen we nu een schatting maken met $A(:,i) = \alpha \cdot w'$.

Stap 14 Bereken nu of er geresampled moet worden met $N_{eff} = \frac{1}{w\cdot w'}$.
Als $N_{eff} < N_{thr}$ moet er geresampled worden, ga dan naar stap 15. Ga anders naar stap 20.

Stap 15 Hier treedt het resampling proces op. In de stappen 15 t/m 19 wordt gebruik gemaakt van de Multinomial resampling methode. Definiëer de volgende parameters:

$$\begin{aligned}
Q =& \text{cumsum}(w), \\
v_{new} =& \text{zeros}(N), \\
\alpha_{new} =& \text{zeros}(5, N), \\
k =& 1.
\end{aligned} \tag{5.12}$$

Stap 16 Trek een willekeurige waarde tussen 0 en 1, noem deze waarde "*sampl*". Zet l op 1.

Stap 17 Als $Q(l) < sampl$, verhoog l met 1 en doe stap 17 nog een keer. Ga anders naar stap 18.

Stap 18 Zet $v_{new}(k) = v(2, l)$ en $\alpha_{new}(:, k) = \alpha(:, l)$.
Als $k \leq N$, verhoog k met 1 en ga naar stap 16, ga anders naar stap 19.

Stap 19 Het resamplen is klaar met deze laatste stap:
$v(2, :) = v_{new}(:)$ en $\alpha = \alpha_{new}$. Ook worden alle gewichten weer even groot gesteld: $w(:) = \frac{1}{N}$.

Stap 20 Als $i < T$, verhoog i dan met 1 en ga naar stap 6. Anders zijn we klaar en hebben we V en A als respectievelijk de geschatte volatiliteit en de geschatte verzameling van parameters.

We kunnen uiteraard weer V en A in een grafiek uitzetten tegen de tijd om de schattingen visueel te maken.

We hebben nu toegelicht hoe we ons model met onbekende parameters kunnen gebruiken om de volatiliteit van een echte markt te schatten. We kunnen natuurlijk ook dit algoritme gebruiken om de volatiliteit van een gegenereerde markt te schatten, dan verandert enkel de input y van het model. We kunnen als we dit doen de RMSE van onze schattingen weer bijhouden op dezelfde manier als in paragraaf 5.2. Bij het genereren van de resultaten zullen we eerst dit doen om te kijken hoe goed ons model met onbekende parameters de volatiliteit en de parameters schat, vervolgens zullen we het bovenstaande algoritme gebruiken om resultaten te genereren met een echte markt.

Hoofdstuk 6

Resultaten

In dit hoofdstuk zullen we de resultaten van ons onderzoek bespreken. We zullen eerst de resultaten van ons model met bekende parameters bespreken en daarna de resultaten die verkregen zijn met het model met onbekende parameters.

Voor ons model met bekende parameters beginnen we met het simuleren van de markt: hiervoor zullen we met ons model een volatiliteit genereren en aan de hand daarvan genereren we het verloop van de logprijs van een aandeel. Vervolgens zullen we de gegenereerde volatiliteit met ons model proberen te schatten. Hiertoe zullen we eerst de prestaties (qua kwaliteit van de schatting van de volatiliteit en qua snelheid) van ons programma moeten optimaliseren. We moeten onderzoeken welke importance function het beste werkt en vervolgens moeten we met de beste hiervan testen welke resamplemethode het beste werkt en bij welke waarde van N_{thr} deze de beste resultaten geeft. Vervolgens zullen we nog een gevoeligheidsanalyse uitvoeren op de parameters κ, θ, μ, ξ en ρ om te kijken of een verstoring in deze parameters bij het schatten van de volatiliteit slechtere resultaten oplevert.

Als we alle resultaten van ons model met bekende parameters hebben besproken, zullen we overgaan op het bespreken van de resultaten van het model met onbekende parameters. We zullen bij het genereren van deze resultaten de importance function gebruiken die het beste functioneert bij ons model met bekende parameters. We werken in dit model met een vector als input voor het particle filter, dus zullen we het programma moeten draaien met meer particles dan bij het programma met bekende parameters en een resamplemethode die kan werken met onze vector als input. We beginnen weer met het genereren van het verloop van de volatiliteit en de bijbehorende aandeelprijs, waarna we met ons model de volatiliteit en de parameters κ, θ, μ, ξ en ρ gaan schatten. Vervolgens passen we ons model toe op een echt aandeel en presenteren we hiervan de resultaten. Deze resultaten zullen we tot slot nog vergelijken met een soortgelijk onderzoek.

6.1 Resultaten voor het model met bekende parameters

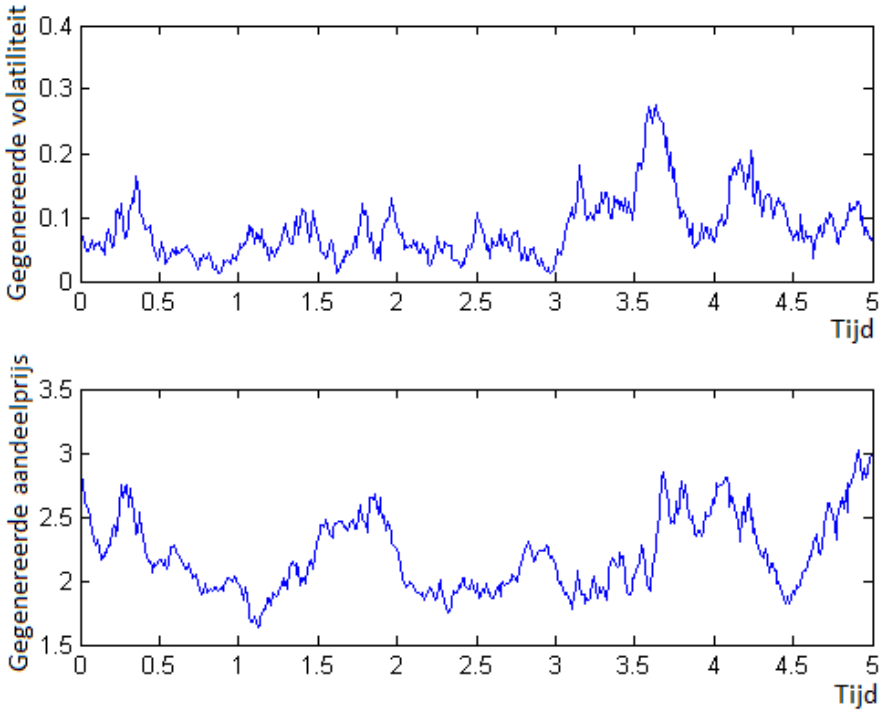
In deze paragraaf bespreken we de resultaten die we verkregen hebben met ons model met bekende parameters. We zullen eerst naar de marktgeneratie kijken en vervolgens zullen we onderzoeken welke importance function, welke resamplemethode en welke waarde voor N_{thr} het beste werkt voor onze simulaties. Tot slot voeren we een gevoeligheidsanalyse uit op de parameters κ, θ, μ, ξ en ρ en bespreken we hiervan de resultaten.

6.1.1 Simuleren van de markt

In paragraaf 5.1 is stap voor stap uitgelegd hoe we een marktaandeel simuleren aan de hand van de theorie uit hoofdstuk 2. In deze paragraaf simuleren we een markt en bespreken we hiervan de resultaten.

In figuur 6.1 staat in de bovenste grafiek de door ons gegenereerde volatiliteit en daaronder de hiermee verkregen aandeelprijs van het aandeel wat we simuleren. Bij deze simulatie hebben we de volgende parameters gebruikt:

$$\kappa = 3 \quad \theta = 0.1 \quad \mu = 0.1 \quad \rho = -0.2 \quad \xi = 0.5 \quad t = 5 \quad \Delta t = 0.01$$



Figuur 6.1: Gegenereerde volatiliteit en aandeelprijs

We zien dat onze volatiliteit en markt netjes gegenereerd worden en de figuren lijken op wat we er van verwachten; schommelende grafieken met veel pieken en weinig regelmaat. Dit is echter maar één gesimuleerde markt, in de volgende paragrafen gaan we verschillende keuzes binnen het particle filter vergelijken en hun prestaties meten over een aantal verschillende markten. Het is natuurlijk van belang dat als we gaan vergelijken wel steeds dezelfde verschillende markten gebruiken voor iedere keuze binnen ons particle filter. Een markt wordt door ons algoritme gegenereerd door willekeurige trekkingen, welke bij simulaties afhankelijk zijn van de stand interne klok van de computer waarop gesimuleerd wordt. Als we steeds dezelfde markt willen simuleren, moeten we hier iets op verzinnen. MATLAB biedt hiervoor de oplossing in de vorm van de formule:

$$RandStream.setDefaultStream(RandStream(`mt19937ar', `Seed', e)),$$

waarin e de 'seed' is van de 'random stream'. Dat wil zeggen dat e het beginpunt vastlegt van de random stream die we gebruiken om de volatiliteit en de markt te genereren. MATLAB zal dus altijd dezelfde volatiliteit en markt genereren voor een vaste e. Als we dus meerdere waarden kiezen van e krijgen we meerdere marktsimulaties die we steeds opnieuw kunnen gebruiken als input voor het model om keuzes te vergelijken, zonder dat we de complete marktsimulatie moeten bewaren in het geheugen van MATLAB.

6.1.2 Resultaten verschillende importance functions

In deze paragraaf bespreken we de effecten van de drie verschillende importance functions op de kwaliteit van de schatting van volatiliteit. De drie importance functions die we testen zijn:

- $\pi(v_k^{(m)}|v_{0:k-1}^{(m)}, y_{1:k}) = p(v_k^{(m)}|v_{k-1}^{(m)})$ de 'suboptimal importance function'.
- $\pi(v_k^{(m)}|v_{0:k-1}^{(m)}, y_{1:k}) = p(v_k^{(m)}|v_{k-1}^{(m)}, y_k, y_{k-1})$ de normaal verdeelde 'optimal importance function'.
- $\pi(v_k^{(m)}|v_{0:k-1}^{(m)}, y_{1:k}) = p(v_k^{(m)}|v_{k-1}^{(m)}, y_k, y_{k-1})$ de non-centraal χ^2 verdeelde 'optimal importance function'.

Om te kijken welke functie de beste resultaten oplevert draaien we het programma voor iedere functie met 20 verschillende markten (gegenereerd met 20 verschillende seeds), met zowel 30, 100 als 500 particles. Vervolgens kijken we per markt, per functie en per aantal particles wat de 'Root mean squared error', oftewel de 'RMSE', is van de geschatte volatiliteit zoals uitgelegd in hoofdstuk 5, hoe vaak er geresampled wordt en

hoe lang het programma nodig gehad heeft om te draaien. Uiteindelijk berekenen we per functie en per aantal particles de gemiddelde RMSE van de volatiliteit, het gemiddeld aantal keren dat er geresampled wordt en de gemiddelde tijd die het programma nodig heeft om te draaien over de 20 markten. Uiteindelijk zullen we aan de hand van hoe laag de RMSE is en hoe snel het programma draait kiezen welke importance function het beste functioneert in ons model en waar we de rest van de resultaten mee zullen genereren. Het aantal keren resamplen houden we bij om te onderzoeken of het programma trager wordt als er vaker geresampled wordt. Bij iedere test gebruiken we $N_{thr} = \frac{N}{3}$ (met N het aantal gebruikte particles) en multinomial resampling en de volgende parameters:

$$\kappa = 3 \quad \theta = 0.1 \quad \mu = 0.1 \quad \rho = -0.2 \quad \xi = 0.5 \quad t = 5 \quad \Delta t = 0.01$$

Hieronder staan per importance function de gemiddelden gegeven.

De 'suboptimal importance function' $\pi(v_k^{(m)}|v_{0:k-1}^{(m)}, y_{1:k}) = p(v_k^{(m)}|v_{k-1}^{(m)})$

De tabel met alle gegevens die gebruikt zijn voor het berekenen van deze gemiddelden staat in Appendix B.3.

	Gemiddelde		
Aantal particles	RMSE	Tijd (in sec)	Aantal keren geresampled
30	0.08843	6.16	39.7
100	0.088337	18.54	45.15
500	0.088287	109.24	47.25

Tabel 6.1: Gemiddelden voor de 'suboptimal importance function'

De normaal verdeelde 'optimal importance function' $\pi(v_k^{(m)}|v_{0:k-1}^{(m)}, y_{1:k}) = p(v_k^{(m)}|v_{k-1}^{(m)}, y_k, y_{k-1})$

De tabel met alle gegevens die gebruikt zijn voor het berekenen van deze gemiddelden staat in Appendix B.2.

	Gemiddelde		
Aantal particles	RMSE	Tijd (in sec)	Aantal keren geresampled
30	0.045343	8.42	29.4
100	0.043569	25.38	33.25
500	0.046833	121.74	39.6

Tabel 6.2: Gemiddelden voor de normaal verdeelde 'optimal importance function'

De non-centraal χ^2 verdeelde 'optimal importance function' $\pi(v_k^{(m)}|v_{0:k-1}^{(m)}, y_{1:k}) = p(v_k^{(m)}|v_{k-1}^{(m)}, y_k, y_{k-1})$

De tabel met alle gegevens die gebruikt zijn voor het berekenen van deze gemiddelden staat in Appendix B.1.

	Gemiddelde		
Aantal particles	RMSE	Tijd (in sec)	Aantal keren geresampled
30	0.046465	19.50	36.5
100	0.04564	62.70	40.65
500	0.044382	310.84	43.85

Tabel 6.3: Gemiddelden voor de non-centraal χ^2 verdeelde 'optimal importance function'

Als we de gemiddelde uitkomsten van de drie importance functions bekijken, zien we dat het gebruikte aantal particles niet veel uitmaakt voor de RMSE, maar wel voor de tijd. Dit is te verklaren door het aantal keren dat er een kansdichtheid van de non-centrale χ^2-verdeling berekend moet worden. Dit kunnen we zien met behulp van de MATLAB profiler, welke bijhoudt hoe lang het programma doet over ieder onderdeel. Het aantal keer dat er geresampled wordt neemt ook toe met het aantal particles, maar volgens de MATLAB profiler neemt dit niet veel tijd in beslag.

De verschillende importance functions met elkaar vergeleken

We hebben bij iedere importance function gezien dat het gebruikte aantal particles niet heel veel uitmaakt voor de gemiddelde RMSE van de schatting van de volatiliteit. De gemiddelde RMSE bij de 'suboptimal importance function' is wel bijna tweemaal zo groot als de gemiddelde RMSE van de 'optimal importance function', zowel de normaal verdeelde als de non-centraal χ^2-verdeelde variant, welke dan ook een ongeveer gelijke gemiddelde RMSE hebben. De tijd die het programma met de 'suboptimal importance function' gemiddeld nodig heeft om te draaien is wel heel veel korter dan de tijd die het programma gemiddeld nodig heeft om te draaien met de non-centraal χ^2-verdeelde 'optimal importance function', dit komt doordat er bij gebruik van de 'suboptimal importance function' twee termen in de weight update equation tegen elkaar weggedeeld worden, zoals te zien is in vergelijking (3.54). De normaal verdeelde 'optimal importance function' is ook duidelijk sneller dan de non-centraal χ^2-verdeelde 'optimal importance function' en maar iets trager dan de 'suboptimal importance function', wederom zijn de verschillen in snelheid van het programma te verklaren door het aantal keren dat er kansdichtheden van de non-centrale χ^2-verdeling berekend moeten worden. We zien dat het gemiddeld aantal keren dat er geresampled wordt met gebruik van de 'suboptimal importance function' en de non-centraal χ^2-verdeelde 'optimal importance function' ongeveer gelijk is. Met gebruik van de normaal verdeelde 'optimal importance function' zien we dat er minder vaak geresampled wordt. Het aantal keren dat er geresampled wordt maakt volgens de MATLAB profiler echter niet veel uit voor de tijd die het programma er over doet om te draaien.

Uit de bovenstaande analyse concluderen we dat we met ons programma het beste de normaal verdeelde 'optimal importance function' kunnen gebruiken, deze levert qua gemiddelde RMSE goede resultaten en is daarbij redelijk snel. Bij het testen van de verschillende resamplemethoden, de verschillende waarden voor N_{thr} en de gevoeligheidsanalyse zullen we dan ook deze importance function gebruiken.

6.1.3 Resultaten verschillende resamplemethoden

In deze paragraaf bespreken we de effecten van het gebruik van ieder van de vier verschillende resamplemethoden op de kwaliteit van de schatting van de volatiliteit. De vier resamplemethoden zijn:

- Multinomial resampling
- Residual resampling
- Stratified resampling
- Systematic resampling

Per resamplemethode bekijken we we weer over 20 verschillende markten per markt de RMSE van de schatting van de volatiliteit, de tijd die het programma nodig heeft om te draaien en het aantal keren dat er geresampled wordt. Uit deze metingen berekenen we per resamplemethode de gemiddelde RMSE, de tijd die het programma gemiddeld nodig heeft om te draaien en het aantal keren wat er geresampled wordt en kijken dan aan de hand hiervan of er een beste resamplemethode aan te wijzen is. Het programma laten we 500 particles gebruiken, de normaal verdeelde 'optimal importance function', $N_{thr} = \frac{N}{3} = \frac{500}{3}$ en de volgende parameters:

$$\kappa = 3 \quad \theta = 0.1 \quad \mu = 0.1 \quad \rho = -0.2 \quad \xi = 0.5 \quad t = 5 \quad \Delta t = 0.01$$

De gemiddelden over de 20 markten staan weergegeven in tabel 6.4, de tabel met alle gegevens die gebruikt zijn voor het berekenen van deze gemiddelden staat in Appendix B.4.

	Resamplemethode	Gemiddelde
RMSE	Multinomial	0.045268
	Residual	0.045031
	Stratified	0.045979
	Systematic	0.046542
Tijd (in sec)	Multinomial	120.48
	Residual	122.02
	Stratified	120.32
	Systematic	120.52
Aantal keren geresampled	Multinomial	40.68
	Residual	40.59
	Stratified	40.16
	Systematic	41.05

Tabel 6.4: Gemiddelden verschillende resamplemethoden

We zien dat de gemiddelde RMSE niet veel verschilt tussen de verschillende resamplemethoden, de tijd die het programma gemiddeld nodig heeft om te draaien ook nagenoeg gelijk is en dat er gemiddeld ook ongeveer even vaak geresampled wordt. Op basis van deze bevindingen kunnen we niet een overtuigend beste of slechtste resamplemethode aanwijzen en concluderen we dat ze alle vier even goed functioneren in ons programma. We zullen de multinomial resamplemethode gebruiken voor het testen van de waarden voor N_{thr} en voor de gevoeligheidsanalyse.

6.1.4 Resultaten voor verschillende waarden van N_{thr}

In deze paragraaf analyseren we wat voor effect het aanpassen van de waarde N_{thr} heeft op de geschatte volatiliteit. De waarde N_{thr} is de waarde die aangeeft wanneer er geresampled moet worden; als:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}(w_k^{(i)})^2} < N_{thr},$$

dan wordt er geresampled. We testen de volgende waarden voor N_{thr}: $\frac{N}{6}$, $\frac{N}{3}$, $\frac{N}{2}$ en $\frac{2N}{3}$, met N het totaal aantal gebruikte particles. We testen door het programma weer voor 20 verschillende markten te laten draaien en daarbij per markt en per waarde van N_{thr} de RMSE van de schatting van de volatiliteit te berekenen, de tijd die het programma nodig heeft om te draaien bij te houden en bij te houden hoevaak er geresampled wordt. Over deze 20 markten berekenen we per waarde van N_{thr} de gemiddelde RMSE, de gemiddelde tijd die het programma nodig heeft om te draaien en het gemiddeld aantal keren dat er geresampled wordt. Aan de hand van deze gegevens zullen we kijken of er een beste waarde voor N_{thr} aan te wijzen is. Het programma draait met 500 particles, de normaal verdeelde 'optimal importance funcion', multinomial resampling en de volgende parameters

$$\kappa = 3 \quad \theta = 0.1 \quad \mu = 0.1 \quad \rho = -0.2 \quad \xi = 0.5 \quad t = 5 \quad \Delta t = 0.01$$

De gemiddelden over de 20 markten zijn weergegeven in tabel 6.5, de tabel met alle gegevens die gebruikt zijn voor het berekenen van deze gemiddelden staat in Appendix B.5.

	N_{thr}	Gemiddelde
RMSE	$\frac{N}{6}$	0.045925
	$\frac{N}{3}$	0.046403
	$\frac{N}{2}$	0.044062
	$\frac{2N}{3}$	0.044129
Tijd (in sec)	$\frac{N}{6}$	121.30
	$\frac{N}{3}$	121.13
	$\frac{N}{2}$	121.10
	$\frac{2N}{3}$	121.17
Aantal keren geresampled	$\frac{N}{6}$	27.05
	$\frac{N}{3}$	39.9
	$\frac{N}{2}$	54.95
	$\frac{2N}{3}$	83.5

Tabel 6.5: Gemiddelden verschillende waarden N_{thr}

Aan de hand van deze resultaten kunnen we zien dat de gemiddelde waarden van de RMSE voor de verschillende waarden van N_{thr} niet veel van elkaar verschillen. Ook heeft het programma voor iedere waarde gemiddeld even lang de tijd nodig om te draaien, we zien enkel dat als N_{thr} groter wordt, er vaker geresampled wordt. Dat er vaker geresampled wordt als N_{thr} groter wordt, is logisch, aangezien N_{eff} dan makkelijker kleiner is dan N_{thr}. Aan de hand van deze resultaten kunnen we geen optimale waarde voor N_{thr} aanwijzen en concluderen we dat iedere waarde een even goede schatting van de volatiliteit oplevert.

6.1.5 Gevoeligheidsanalyse parameters

Vervolgens voeren we een gevoeligheidsanalyse uit voor de parameters κ, θ, μ, ξ en ρ. Dit houdt in dat we onderzoeken in hoeverre het model nog goede schattingen van de volatiliteit oplevert op het moment dat we deze parameters bij het schatten ongelijk kiezen aan de waarden van deze parameters waarmee we de markt hebben gesimuleerd. We testen dit door het programma weer over 20 verschillende markten te laten draaien en bij iedere markt iedere parameter een aantal keren te verstoren, terwijl de rest van de parameters gelijk is aan hun ware waarde. We kijken bij deze 20 markten weer naar de RMSE van de schatting van de volatiliteit en nemen vervolgens over deze 20 markten de gemiddelde RMSE. Deze zullen we hieronder weergeven. Als we een parameter verstoren, maken we hem x maal zo groot als zijn werkelijke waarde, met x= $1, 2, \ldots, 10$. Het programma draait iedere keer met de normaal verdeelde 'optimal importance function', 500 particles, multinomial resampling, $N_{thr} = N/3$ en de volgende waarden van de parameters:

$$\kappa = 3 \quad \theta = 0.1 \quad \mu = 0.1 \quad \rho = -0.1 \quad \xi = 0.5 \quad t = 5 \quad \Delta t = 0.01$$

	Gemiddelde RMSE bij verstoring met factor x				
Verstoring van:	x=1	x=2	x=3	x=4	x=5
κ	0,047333	0,049531	0,047501	0,046423	0,047594
θ	0,046876	0,056804	0,08627	0,138476	0,187595
μ	0,047329	0,047671	0,047839	0,047768	0,046499
ρ	0,045747	0,04994	0,043158	0,047277	0,049536
ξ	0,046136	0,053203	0,080543	0,10737	0,122589
	x=6	x=7	x=8	x=9	x=10
κ	0,047959	0,04831	0,047692	0,049589	0,04994
θ	0,25911	0,330563	0,41127	0,477421	0,553892
μ	0,047497	0,046658	0,049626	0,050115	0,049885
ρ	0,047083	0,049802	0,051612	0,056419	0,058184
ξ	0,127986	0,151043	0,177291	0,205602	0,209244

Tabel 6.6: Gemiddelden gevoeligheidsanalyse

In tabel 6.6 zien we dat het verstoren van κ niet heel veel invloed heeft op de prestaties van ons particle filter, de gemiddelde RMSE van de volatiliteit gaat maar neemt maar iets toe.

Verstoren van θ heeft echter wel grote gevolgen. We zien dat bij het verstoren van θ de gemiddelde RMSE van de schatting van de volatiliteit bijna met de factor x toeneemt.

We zien dat het verstoren van μ niet heel erg veel invloed heeft. Bij het tienmaal verstoren van deze factor is de gemiddelde RMSE maar iets groter dan de originele waarde.

Omdat ρ geen -1 kan worden hebben we deze in plaats daarvan op -0.99 gezet voor x=10. We zien dat een verstoring van ρ niet veel invloed heeft op de RMSE van de schatting van de volatiliteit.

Tot slot zien we dat ξ verstoren wel al snel grote gevolgen heeft. Bij tienmaal verstoren is de gemiddelde RMSE van de schatting van de volatiliteit bijna vijf keer zo groot.

Uit deze resultaten kunnen we opmaken dat het verstoren van θ en ξ de grootste gevolgen hebben op de kwaliteit van onze resultaten. Het verstoren van de rest van de parameters valt hier redelijk bij in het niet. We moeten met ons model met onbekende parameters er dus goed op letten dat θ en ξ goed geschat worden, als dit niet gebeurt, dan zal dit grote gevolgen hebben op de kwaliteit van de schatting van de volatiliteit.

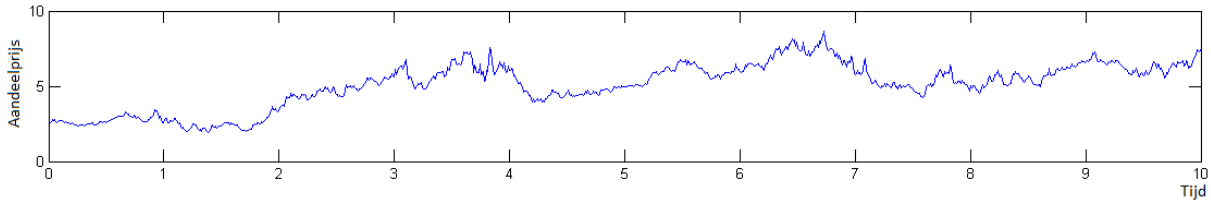
6.1.6 Visualisatie

Tot slot hebben we met ons model met bekende parameters nog een drietal grafieken gegenereerd om de schatting van de volatiliteit te visualiseren. We hebben dit gedaan met de volgende parameters:

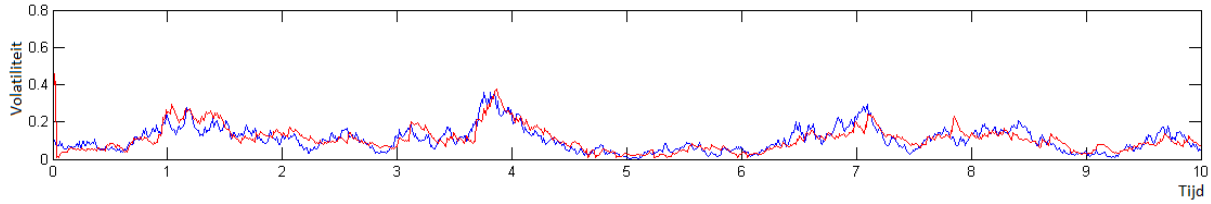
$$\kappa = 3 \quad \theta = 0.1 \quad \mu = 0.1 \quad \rho = -0.2 \quad \xi = 0.5 \quad t = 10 \quad \Delta t = 0.01$$

We gebruiken de normaal verdeelde optimal importance function, Multinomial resampling, $N_{thr} = N/3$ en een seed van 7.

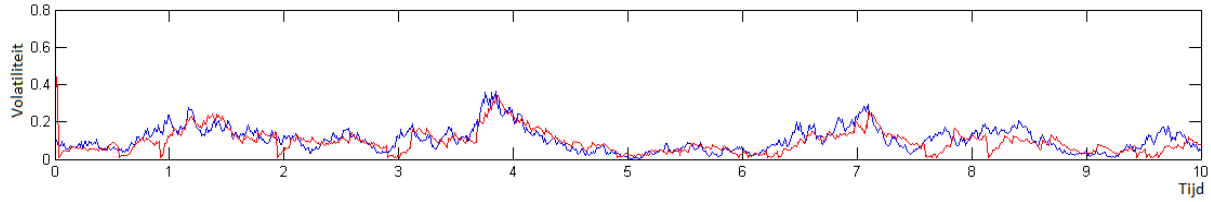
Eerst genereren we zelf een volatiliteit en simuleren we aan de hand hiervan een markt. Vervolgens schatten we het verloop van de volatiliteit een keer met 500 particles en een keer met 2000 particles. In het eerste figuur staat de gegenereerde markt, in het tweede en het derde figuur staat de geschatte volatiliteit als rode lijn weergegeven en de gegenereerde volatiliteit als blauwe.



Figuur 6.2: Gegenereerde marktdata



Figuur 6.3: Geschatte volatiliteit met 500 particles



Figuur 6.4: Geschatte volatiliteit met 2000 particles

We zien dat de volatiliteit zowel bij 500 particles als bij 2000 particles goed benaderd wordt. Opmerkelijk is dat de geschatte volatiliteit met 500 particles op veel punten beter lijkt dan de geschatte volatiliteit met 2000 particles.

6.2 Resultaten voor het model met onbekende parameters

In deze paragraaf bekijken we de resultaten die we gegenereerd hebben met ons model met onbekende parameters. We genereren eerst zelf een volatiliteit en aan de hand daarvan een aandeelprijs. Vervolgens schatten we met ons model met onbekende parameters de volatiliteit van dit aandeel en de bijbehorende parameters en onderzoeken we hoe goed onze schattingen zijn. Tot slot zullen we met ons model een echt aandeel analyseren en de door ons geschatte volatiliteit vergelijken met de door Shin Ichi Aihara et al.[1] gegenereerde volatiliteit van hetzelfde aandeel.

6.2.1 Resultaten voor gegenereerde markten

In deze paragraaf genereren we eerst een volatiliteit met een daarbij horende markt met bekende parameters en gaan vervolgens de volatiliteit en de parameters schatten met ons model. De waarden van de parameters waarmee we de volatiliteit en de markt gegenereerd hebben zijn:

$$\theta = 0.1 \quad \rho = -0.2 \quad \xi = 0.5 \quad \kappa = 3 \quad \mu = 0.1$$

Bij het schatten van de volatiliteit en de onbekende parameters hebben we voor alle parameters op tijdstip nul beginwaarden gekozen uit een uniform verdeeld interval. Voor deze resultaten hebben we de volgende intervallen gebruikt:

$$\theta \in \mathcal{U}[0.01, 2.01] \quad \rho \in \mathcal{U}[-0.5, 0] \quad \xi \in \mathcal{U}[0.01, 0.91] \quad \kappa \in \mathcal{U}[1, 9] \quad \mu \in \mathcal{U}[0.05, 0.5]$$

Voor de geschatte volatiliteit op tijdstip 0 hebben we de volatiliteit getrokken uit $\mathcal{N}(0.27, 0.02^2)$.

We hebben gebruik gemaakt van $N = 500$ particles, de normaal verdeelde 'optimal importance function', systematic resampling en een N_{thr} van $N/3$.
We hebben wederom tijdstappen genomen van $\Delta t = 0.01$ en als eindtijd hebben we $t = 5$ gebruikt.
We hebben 20 maal een verschillende markt gesimuleerd, de resultaten die we hier presenteren zijn de gemiddelde waarden van deze 20 markten. De 20 gebruikte marktsimulaties zijn overigens dezelfde 20 markten als die we in de voorgaande paragrafen gebruikt hebben. Hieronder staan de gemiddelde RMSE van de schatting van de volatiliteit over de 20 markten, de gemiddelde tijd die het programma nodig heeft om te draaien en het gemiddeld aantal keren dat het nodig was om te resamplen:

	Gemiddelde
RMSE	0.059577876
Tijd (in seconden)	335.8118373
Aantal keren geresampled	52.8

Tabel 6.7: Resultaten schatten van volatiliteit met onbekende parameters

De resultaten uit tabel 6.7 zijn te vergelijken met de resultaten van Systematic resampling in tabel 6.4 aangezien we daar dezelfde importance function en resamplemethode hebben gebruikt met hetzelfde aantal particles en dezelfde N_{thr}, wanneer we dit doen kunnen we zeggen dat het model met bekende parameters de volatiliteit beter schat dan het model met onbekende parameters, wat natuurlijk niet verrassend is. De gemiddelde RMSE van de schatting van de volatiliteit is met het model met onbekende parameters echter niet heel veel slechter dan die van de schatting van de volatiliteit met het model met bekende parameters. We zien ook dat het programma meer tijd nodig heeft om te draaien wanneer de parameters onbekend zijn dan wanneer ze bekend zijn. Dit komt omdat in het model met onbekende parameters naast de volatiliteit ook de parameters geschat worden. Hiervoor moeten er vaker kansdichtheden berekend worden van verdelingen dan bij het model met bekende parameters en dus heeft het programma meer tijd nodig om te draaien. We kunnen uit deze resultaten concluderen dat ons model de volatiliteit van het aandeel in ieder geval goed benadert binnen afzienbare tijd.

We hebben ook bijgehouden wat de gemiddelde waarden van de geschatte parameters waren aan het begin en aan het einde van het draaien van het programma. Ook hebben we bijgehouden hoevaak we de parameters bijgesteld hebben omdat ze onder, danwel boven een bepaalde grens kwamen. Deze waarden zijn dus het gemiddeld aantal keren dat de waarde van één particle onder de ondergrens of boven de bovengrens

komt. We hebben de volgende grenzen gehanteerd, voor μ zijn alle waarden toegestaan:

$$\theta \geq 0.0001 \quad -0.999 \leq \rho \leq 0.999 \quad \xi \geq 0.001 \quad \kappa \geq 0.001$$

θ, ξ en κ kunnen volgens de theorie niet gelijk of kleiner zijn dan 0 en ρ ligt tussen de 1 en de -1, als ρ echter gelijk aan -1 of 1 is, levert dit problemen in de programmatuur op.

Hier volgt de tabel met hierin de beginwaarde en eindwaarde van de parameters, evenals het aantal maal dat een parameter is bijgesteld, de tabel met alle gegevens die gebruikt zijn voor het berekenen van deze gemiddelden staat in Appendix B.6.

	Beginwaarde	Eindwaarde	Aantal keren bijgesteld
θ	1.011916934	0.097389543	170.9
ρ	-0.250831509	-0.211472827	0
ξ	0.461949277	0.679100831	272.95
κ	5.008538584	4.982528771	38.35
μ	0.272382718	0.219748193	n.v.t.

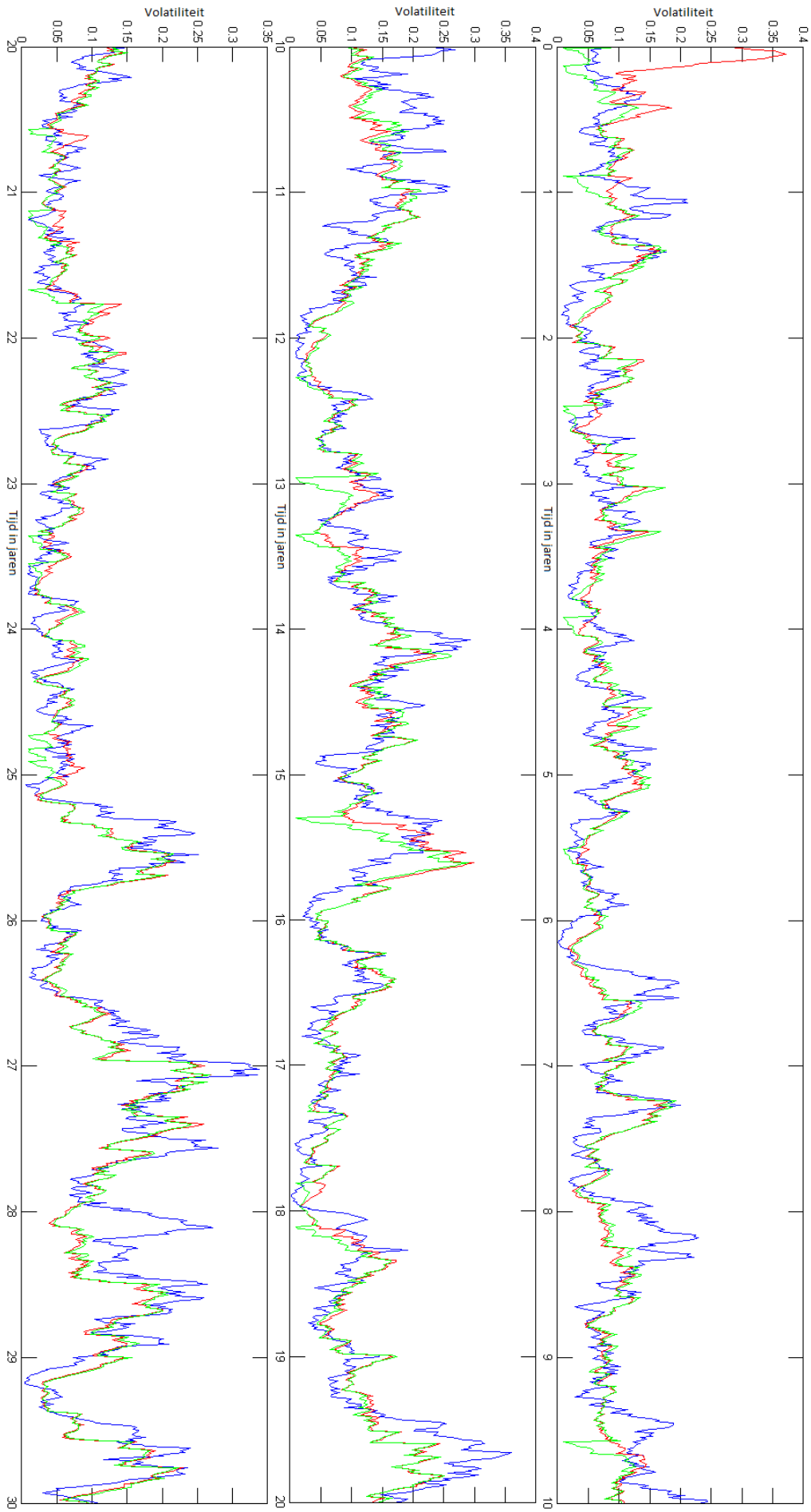
Tabel 6.8: Verandering van de onbekende parameters

We zien dat de gemiddelde eindwaarden van θ en ρ dicht in de buurt van hun echte waarde liggen, maar dat de benaderingen van ξ, κ en μ minder goed lijken op hun echte waarden.

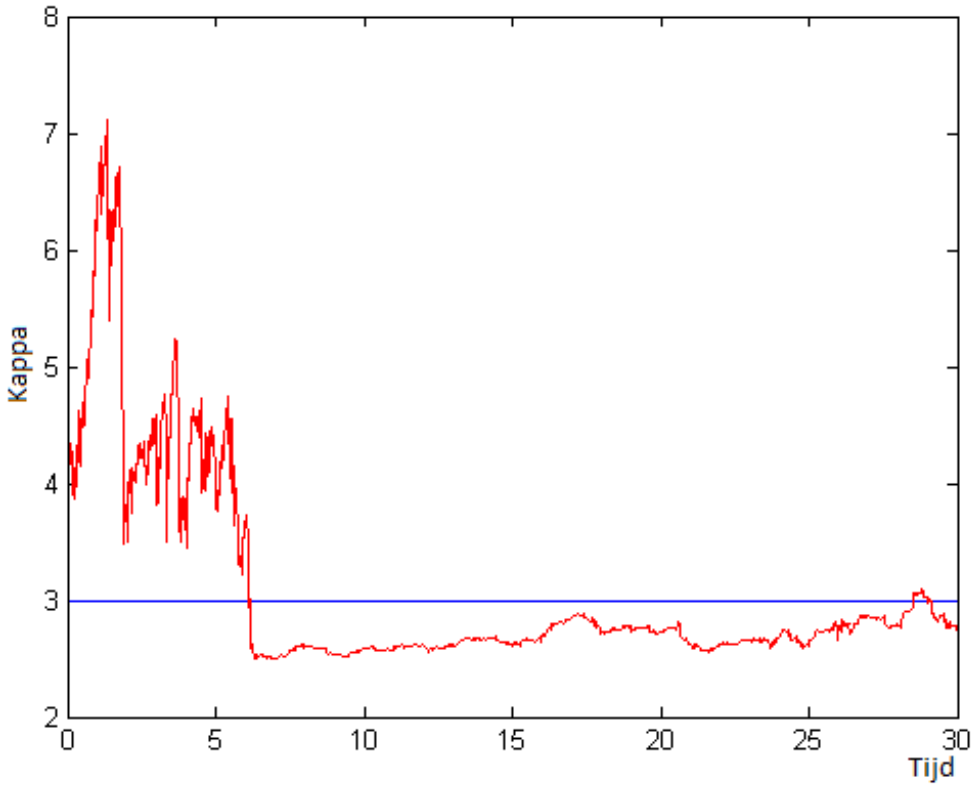
We hebben ook nog éénmaal het programma gedraaid om visueel het model met onbekende parameters te vergelijken met het model met bekende parameters, hierbij simuleren we van $t = 0$ tot $t = 30$ en met verder dezelfde parameters als aan het begin van deze paragraaf. In de grafieken op de volgende pagina staat de schatting van de volatiliteit uitgezet tegen de tijd. De grafieken vormen samen één geheel, maar om deze goed te kunnen lezen is het nodig geweest ze op te splitsen in drie delen die onder elkaar zijn gezet. De blauwe lijn is de gegenereerde, dus de echte volatiliteit, de rode lijn is de geschatte volatiliteit met het model met onbekende parameters en de groene lijn is de geschatte volatiliteit met het model met bekende parameters. De seed die we hier hebben gebruikt is $e = 1234$.

We zien in figuur 6.5 dat de groene lijn al heel al heel vlug de echte volatiliteit ongeveer benadert. De rode lijn is hier iets trager in, maar niet heel veel trager dan de groene. Na $t = 1$ zijn de rode en de groene lijn al nagenoeg gelijk en wijken maar heel af en toe nog significant van elkaar af. Het model met onbekende parameters benadert de volatiliteit dus al heel snel net zo goed als het model met bekende parameters. Dit is een uiterst positief resultaat.

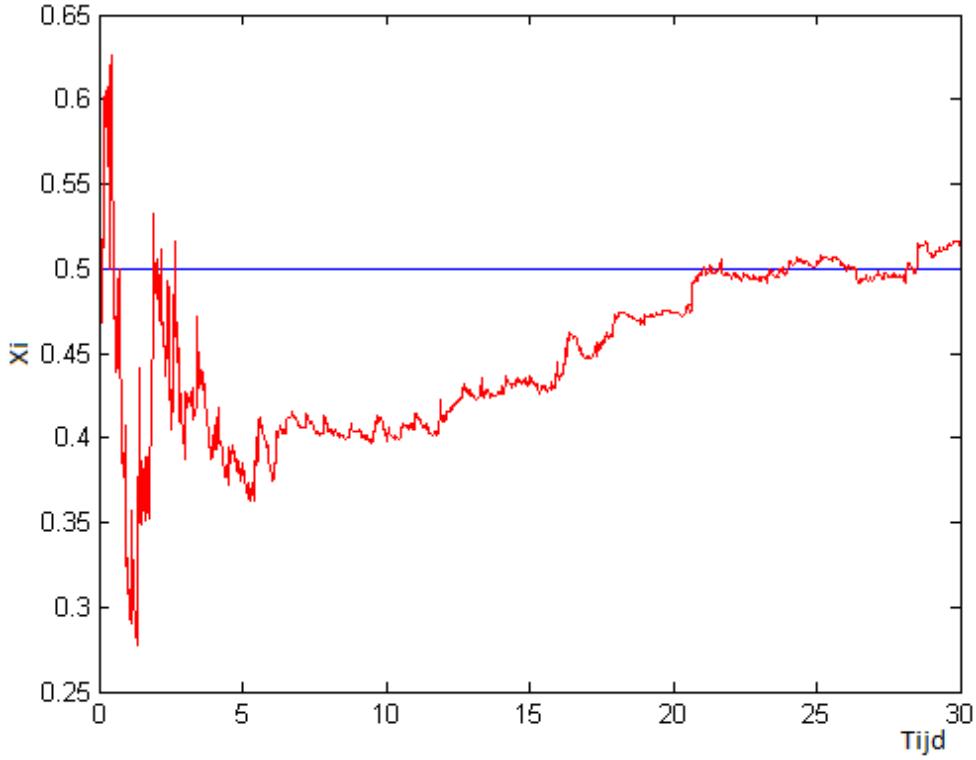
We hebben ook gekeken naar de schatting van de parameters κ, θ, μ, ξ en ρ met ons model met onbekende parameters. In de figuur 6.6 t/m 6.10 ziet u de schatting van de parameters uitgezet tegen de tijd. De blauwe, constante lijnen in deze figuren zijn de echte waarden van de parameters, de rode lijnen zijn de benaderingen hiervan.



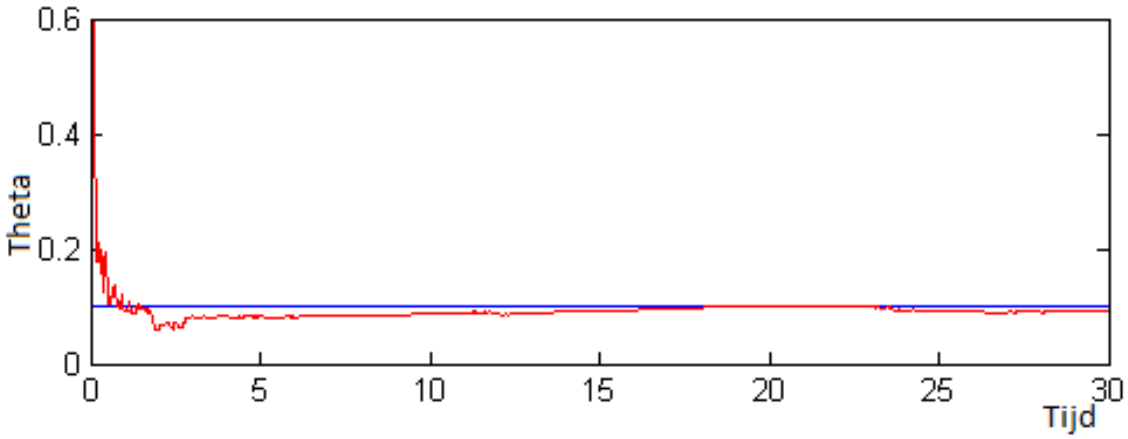
Figuur 6.5: Volatiliteit van $t = 0$ tot $t = 30$



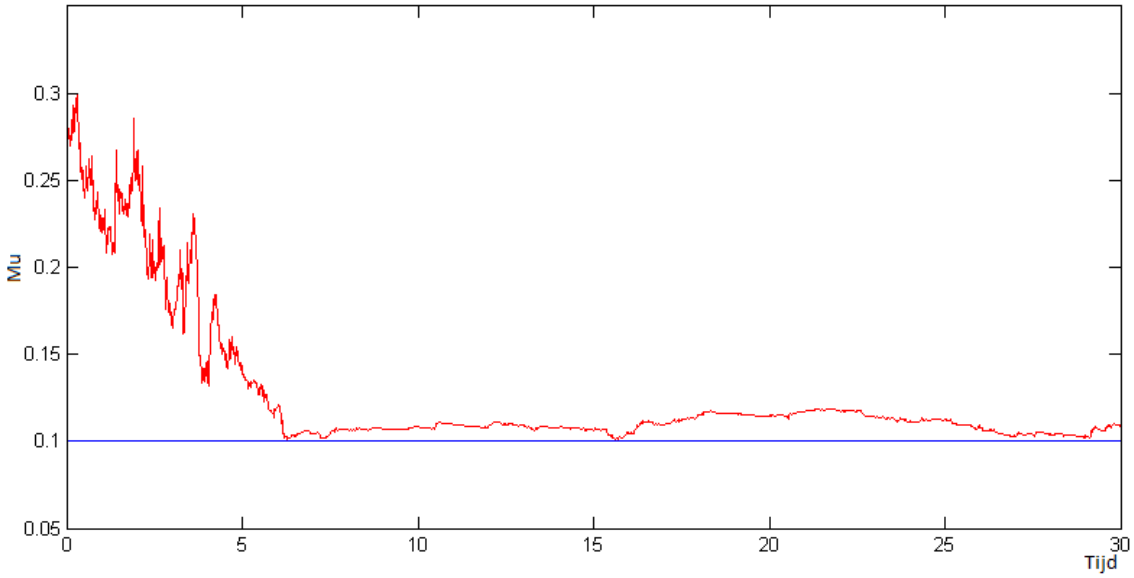
Figuur 6.6: κ als onbekende parameter



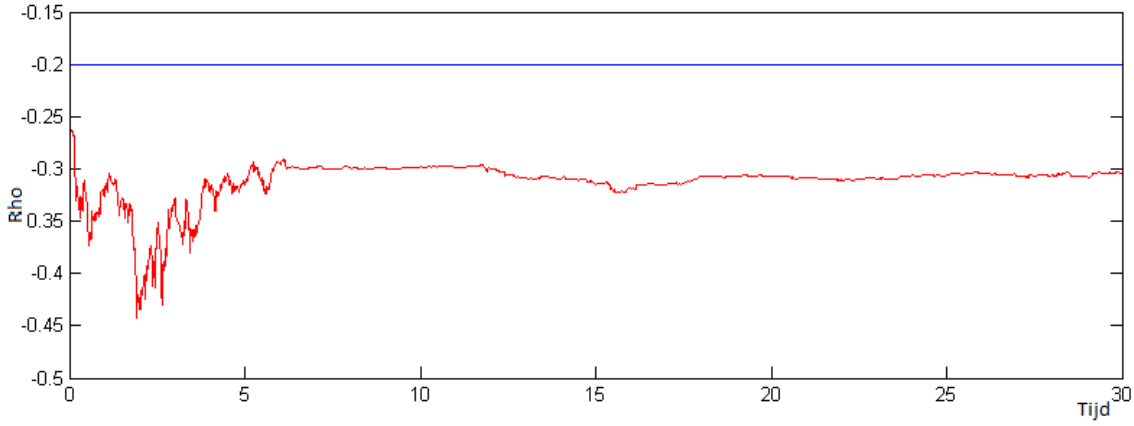
Figuur 6.7: ξ als onbekende parameter



Figuur 6.8: θ als onbekende parameter



Figuur 6.9: μ als onbekende parameter



Figuur 6.10: ρ als onbekende parameter

We zien in figuur 6.6 dat κ na $t = 5$ rond een bepaalde waarde blijft schommelen, maar wel rond een waarde die relatief ver van zijn echte waarde afligt (met een verschil van zo ongeveer 0.5). κ wordt dus niet heel erg goed benaderd. In figuur 6.7 zien we dat ξ in het begin heel erg schommelt, maar uiteindelijk wel de echte waarde benadert. Deze wordt dus ook goed benaderd, maar wel na een wat langere tijd. In figuur 6.8 zien we dat θ zich al heel snel instelt op een evenwicht rondom zijn echte waarde, deze wordt dus goed benaderd. Uit figuur 6.9 maken we op dat ook μ goed benaderd wordt, deze heeft ongeveer net zoveel tijd nodig om in te stellen als κ en blijft vervolgens de hele tijd net iets boven zijn echte waarde hangen. Tot slot zien we in figuur 6.10 dat ρ zich instelt op een evenwicht wat ongelijk is aan zijn echte waarde en dit ongeveer net zo snel doet als κ en μ. Hij zit echter niet heel ver van zijn echte waarde af, zoals κ. Al met al kunnen we tevreden zijn met de schattingen van de parameters bij deze simulatie.

6.2.2 Resultaten voor een echte markt

In deze paragraaf bespreken we de resultaten die we met ons model krijgen als we een echte markt analyseren. We gebruiken hiervoor de AEX-index en analyseren deze van 12 oktober 1992 tot op heden. De reden waarom we 12 oktober gebruiken, is omdat er vanaf die datum gegevens bekend zijn van de AEX-index. Omdat er in een beursjaar geen 365 dagen zitten (in het weekend is de beurs niet open en op feestdagen ook niet) hebben we aangenomen dat er in één beursjaar 252 dagen zitten. We kiezen daarom als stapgrootte $\Delta t = \frac{1}{252}$, op deze manier berekenen we de 'annualized volatility', dit is de meest gangbare vorm om de volatiliteit te meten.

De onbekende parameters van het model trekken we wederom uit een uniforme verdeling op tijdstip 0, we gebruiken nu grotere intervallen dan in het geval waarin we de parameters schatten terwijl we deze eigenlijk al kennen van het genereren van de markt. We trekken de parameters op tijdstip 0 als volgt:

$$\kappa \in \mathcal{U}[1, 10] \quad \mu \in \mathcal{U}[-0.2, 0.3] \quad \xi \in \mathcal{U}[0.1, 0.6] \quad \rho \in \mathcal{U}[-0.8, -0.1] \quad \theta \in \mathcal{U}[0.01, 2.01]$$

We trekken de volatiliteit op tijdstip nul uit $\mathcal{N}(0.27, 0.02^2)$, we gebruiken de normaal verdeelde 'optimal importance function', 500 particles en $N_{thr} = \frac{2N}{3}$. Ook hebben we wederom grenzen gesteld aan de meeste parameters, zodat ze geen waarden aan kunnen nemen die ze in de praktijk niet kunnen aannemen, deze grenzen zijn als volgt:

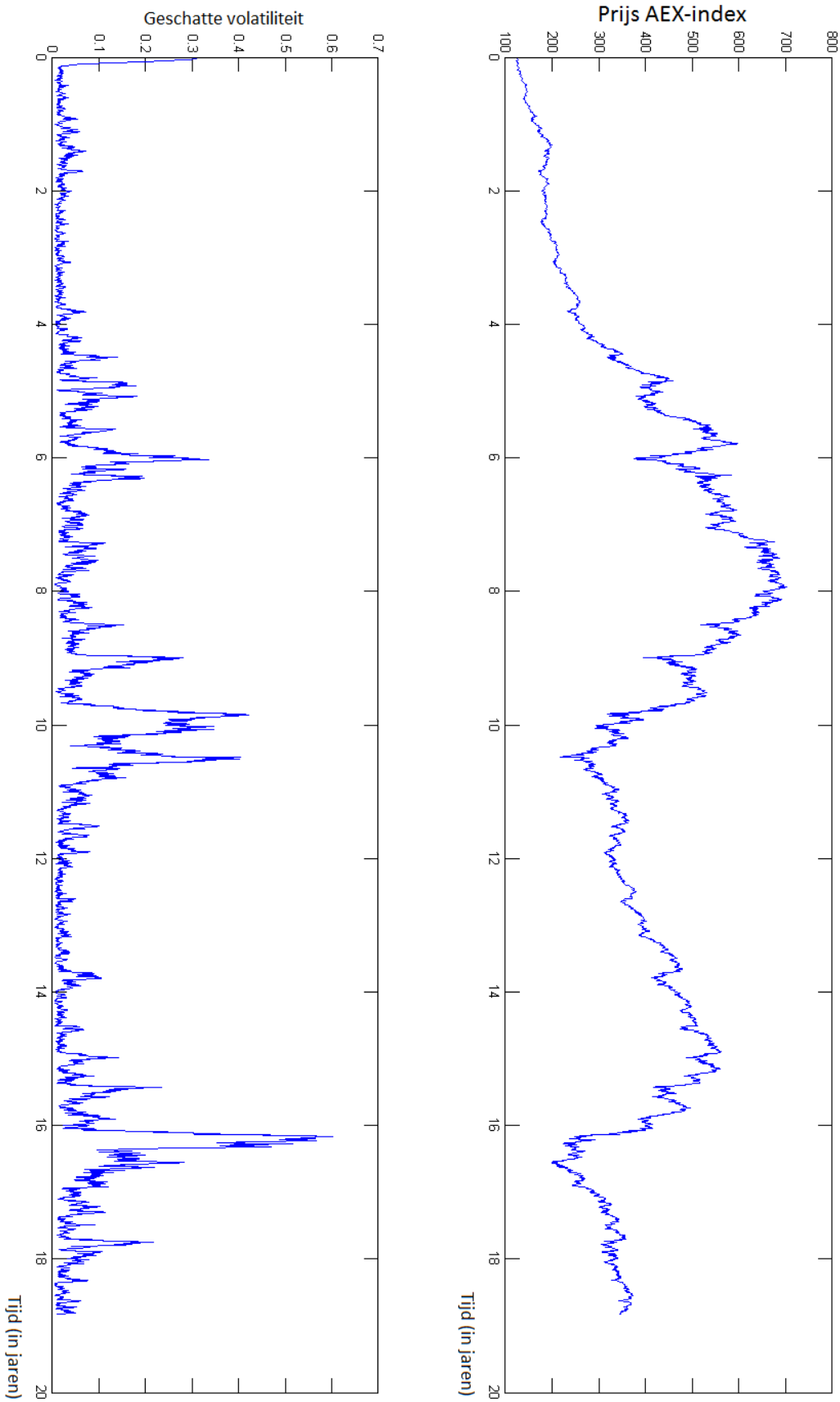
$$\theta \geq 0.0001 \quad -0.999 \leq \rho \leq 0.999 \quad \xi \geq 0.00001 \quad \kappa \geq 0.001$$

In figuur 6.11 t/m 6.16 zijn de resultaten te zien van de geschatte volatiliteit en de geschatte waarden van de parameters. Tijdstip 0 staat telkens gelijk aan 12 oktober 1992, tijdstip 20 staat telkens gelijk aan 12 oktober 2012.

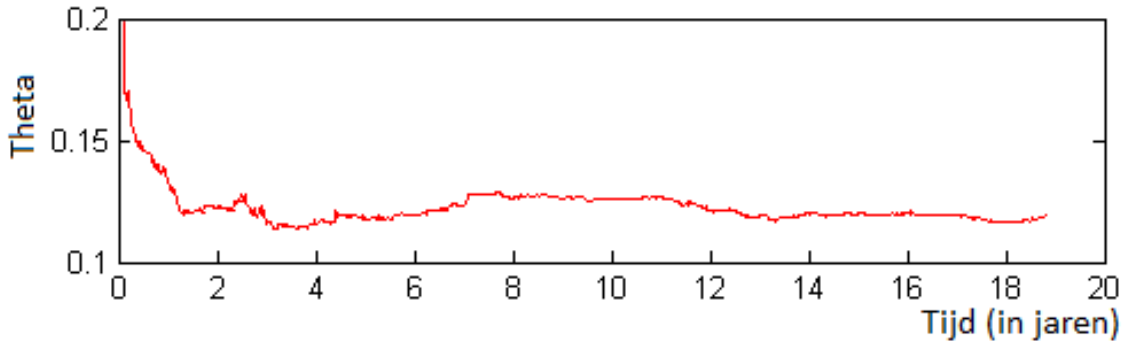
In figuur 6.11 zien we zowel de aandeelprijs van de AEX-index van de afgelopen 20 jaar, als onze schatting van de volatiliteit over de afgelopen 20 jaar. We kunnen duidelijk zien dat in tijden van economische crisis, rond de tijdstippen $t = 10$ en $t = 16$ welke corresponderen met 2001 en eind 2006, de volatiliteit omhoog schiet ten opzichte van andere perioden.

In figuur 6.12 zien we dat θ zich al redelijk snel instelt rond een evenwicht van ongeveer 0.125.
In figuur 6.13 zien we dat ρ wat langer de tijd nodig heeft om in te stellen dan θ, maar op den duur rond de -0.1 blijft schommelen. Uit figuur 6.14 kunnen we niet concluderen dat ξ convergeert naar een vaste waarde. Het zou kunnen zijn dat hij meer tijd nodig heeft om zich in te stellen rond een evenwicht. In figuur 6.15 zien we dat ook κ niet duidelijk convergeert, maar met grote pieken schommelt rond de 3.8. In figuur 6.16 zien we dat μ rond de 0.325 schommelt, maar ook weer met pieken.

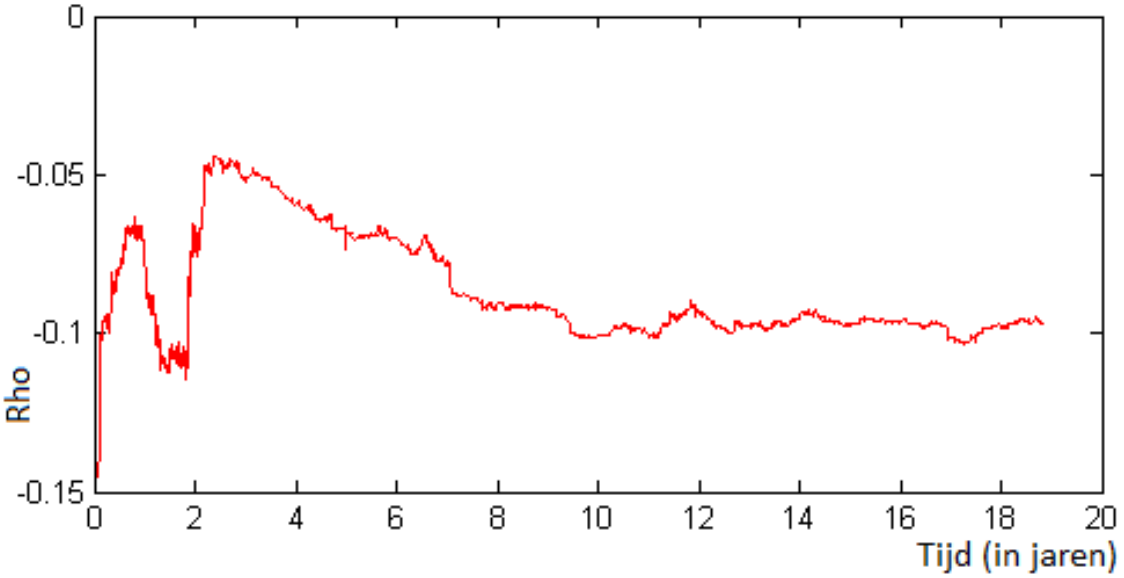
De echte waarden van de volatiliteit en de parameters van het model zijn onbekend, dus we kunnen de door ons geschatte waarden niet vergelijken met de 'echte' waarden, zoals we voorheen deden in ons onderzoek, om de kwaliteit van onze schattingen te meten. Wel kunnen we onze schattingen naast de schattingen van het onderzoek van andere onderzoekers leggen. Dit zullen we dan ook doen in de volgende paragraaf.



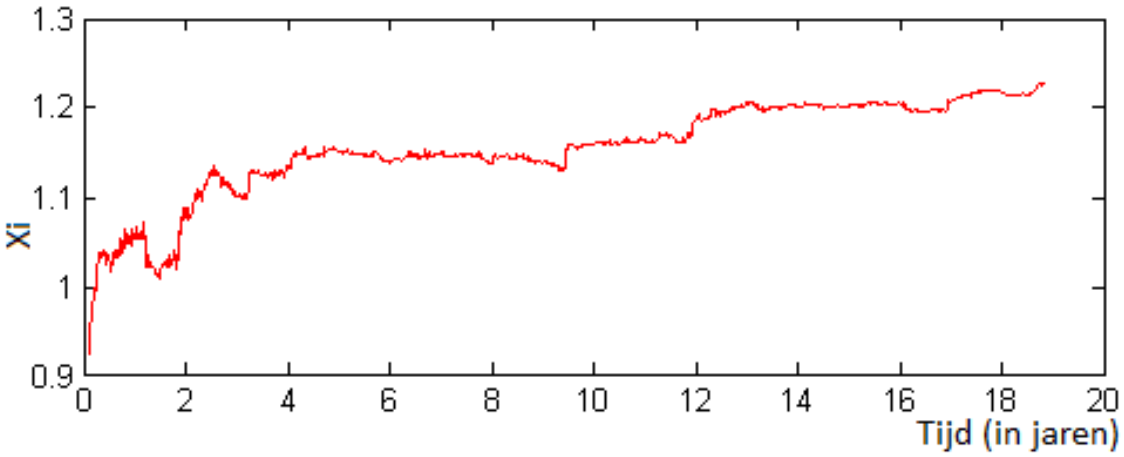
Figuur 6.11: Aandeelprijs en geschatte volatiliteit voor de AEX-index sinds okt 1992



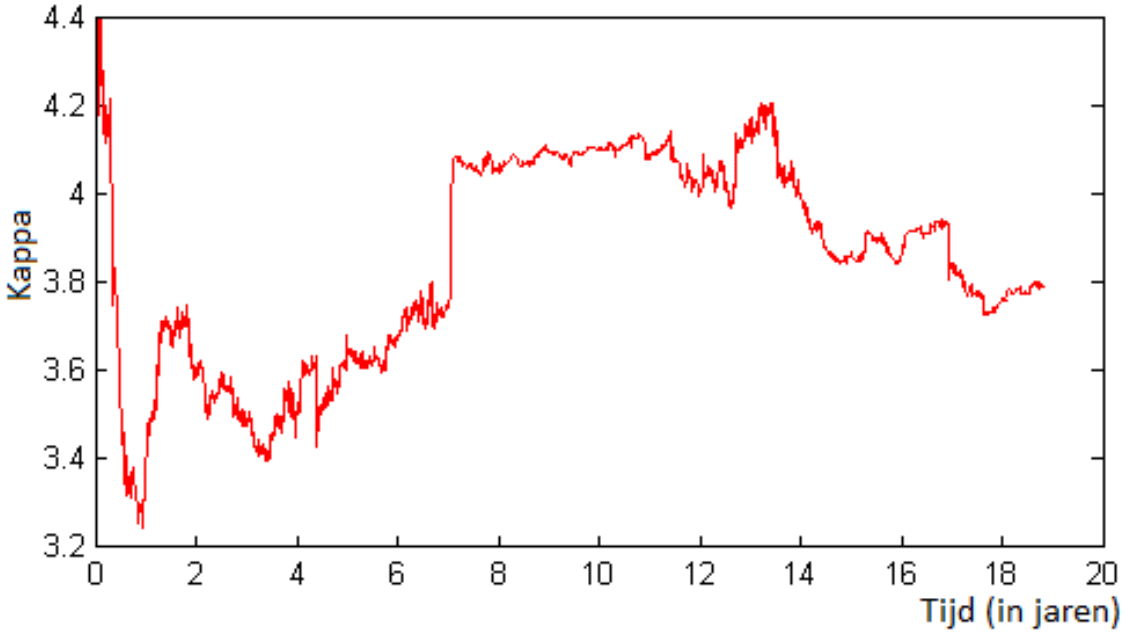
Figuur 6.12: θ voor de AEX-index sinds okt 1992



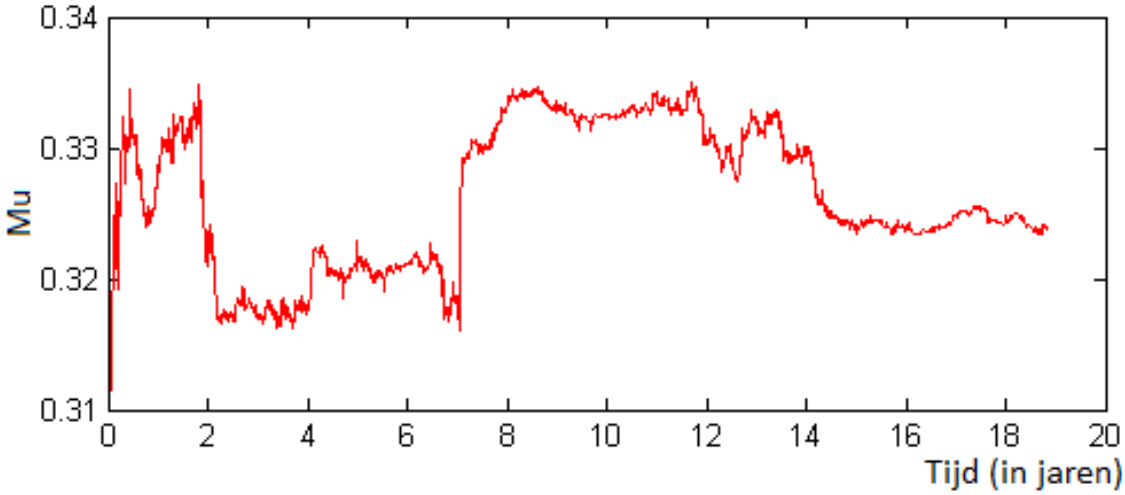
Figuur 6.13: ρ voor de AEX-index sinds okt 1992



Figuur 6.14: ξ voor de AEX-index sinds okt 1992



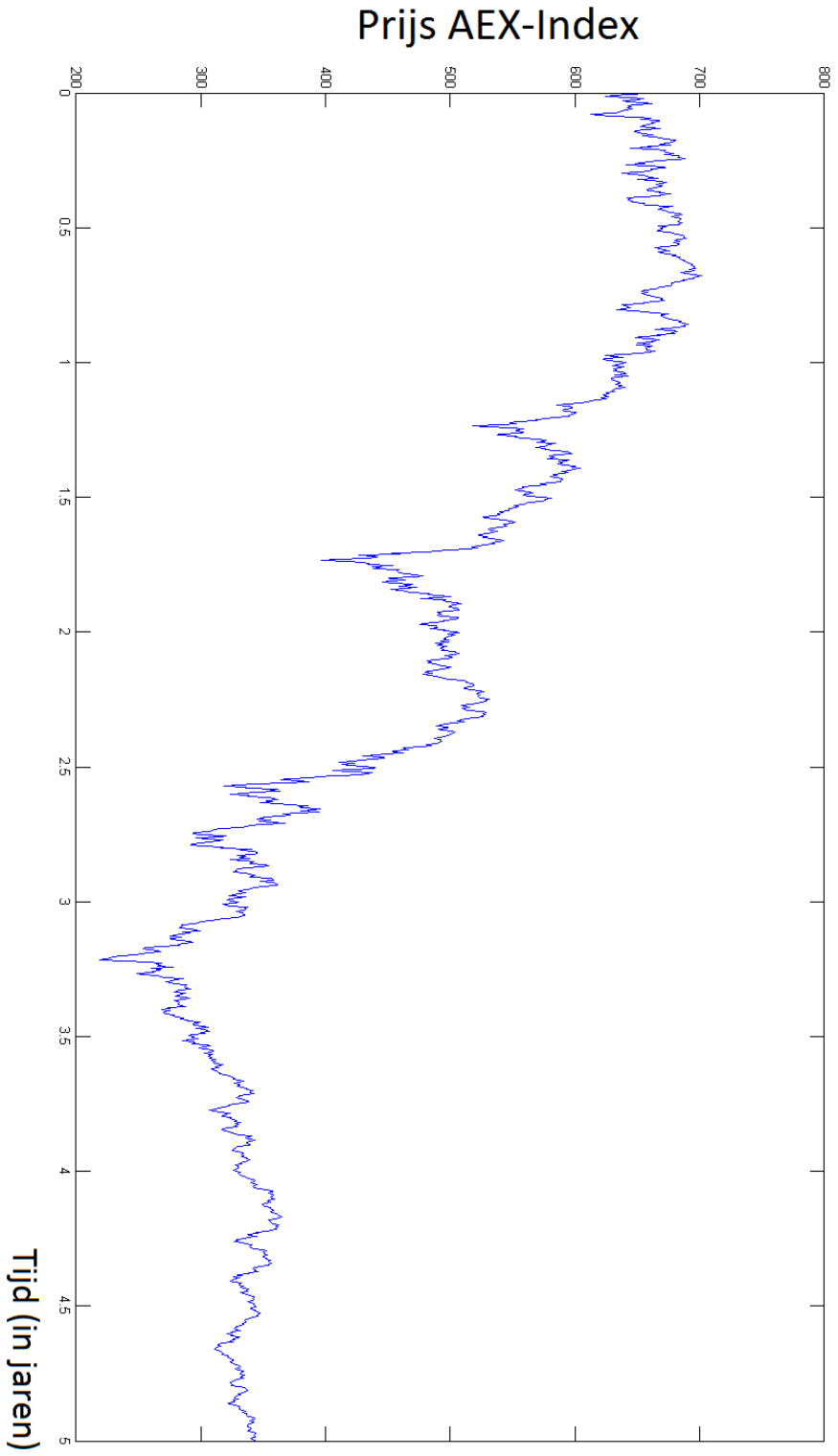
Figuur 6.15: κ voor de AEX-index sinds okt 1992



Figuur 6.16: μ voor de AEX-index sinds okt 1992

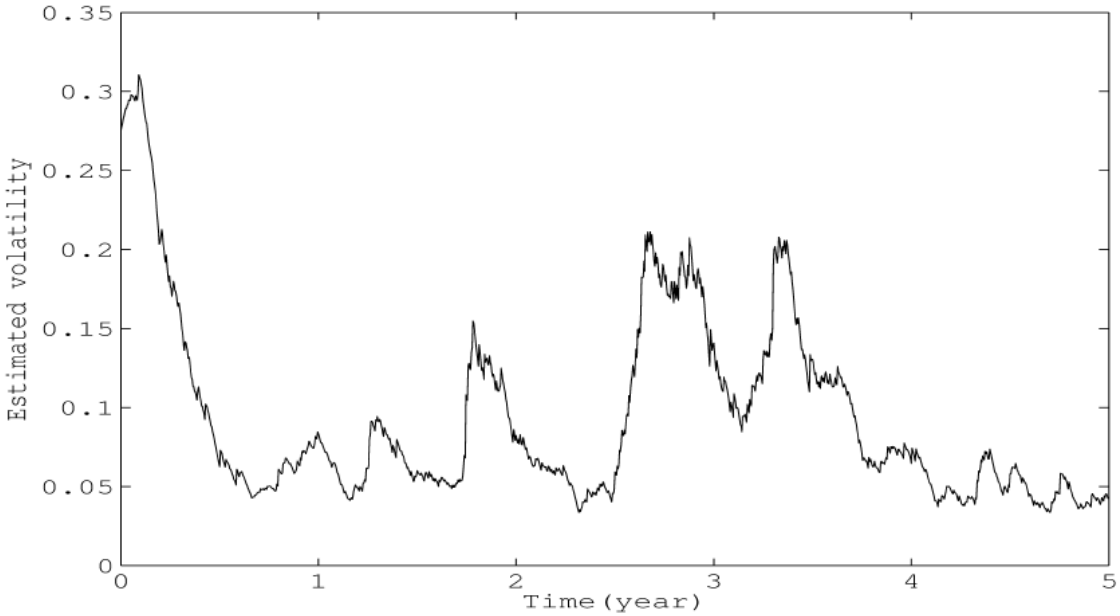
6.2.3 Resultaten vergeleken met ander onderzoek

Shin Ichi Aihara heeft samen met Arunabha Bagchi en Saikat Saha [1] in 2009 een soortgelijk onderzoek uitgevoerd als ons. Uit hun artikel hebben wij dan ook de basis gehaald voor ons model met onbekende parameters. Ons model wijkt iets af van het model dat zij hebben opgesteld (de elementen van α zijn iets anders gedefiniëerd), maar verder lijkt het er genoeg op om de resultaten naast elkaar te leggen en met elkaar te vergelijken. Ook zij hebben de AEX-index gebruikt om resultaten te genereren met hun model, zij hebben het alleen met een ander tijdsinterval gedaan dan wij in paragraaf 6.2.2. In het onderzoek van Shin Ichi Aihara et al. [1] wordt de volatiliteit van de AEX-index geschat vanaf 3 januari 2000 tot en met 3 januari 2005. Daarom voeren wij ons programma nogmaals uit, zoals wij dat in paragraaf 6.2.2 ook gedaan hebben, maar dan op het tijdsinterval van het onderzoek van Shin Ichi Aihara et al. [1] en met $\xi \in \mathcal{U}[0.25, 0.6]$ op tijdstip 0 (dit is anders dan de waarden van Shin Ichi Aihara et al. [1], maar dit voorkomt heel erg veel rekenwerk en scheelt uren in tijd, omdat ξ anders heel vaak bijgesteld moet worden, wat enorm veel tijd kost) . De resultaten staan weergegeven samen met die van Shin Ichi Aihara et al. [1] in figuur 6.17 t/m 6.27, opdat we makkelijk kunnen vergelijken. In de grafieken staat tijdstip 0 gelijk aan 3 januari 2000 en tijdstip 5 gelijk aan 3 januari 2005.

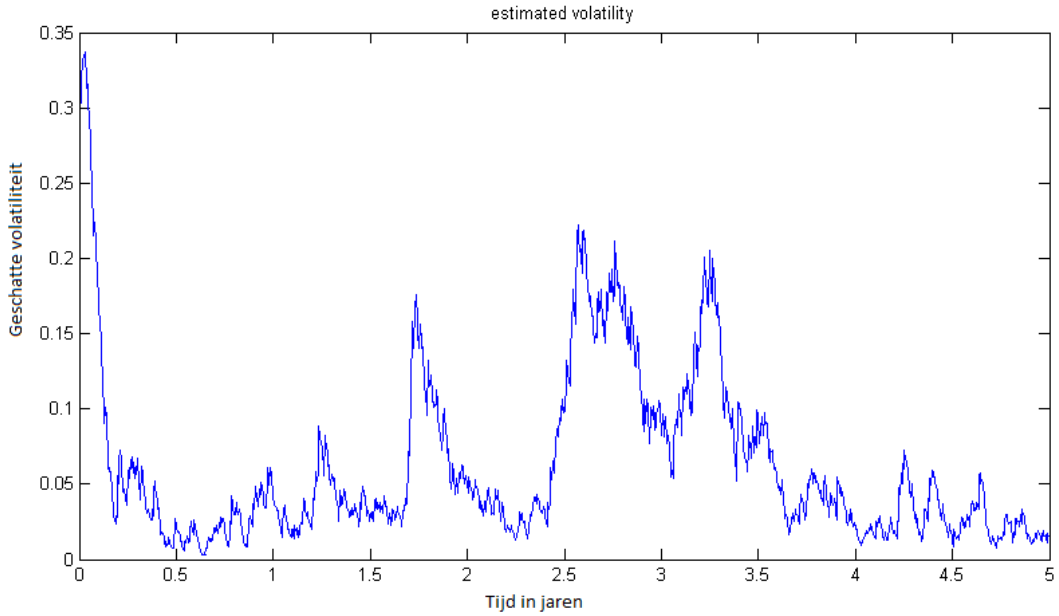


Figuur 6.17: Aandeelprijs van de AEX index vanaf jan 2000

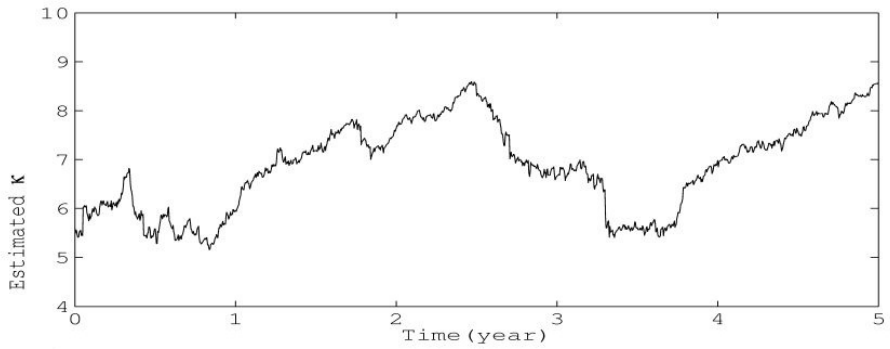
Van het verloop van de aandeelprijs van de AEX-index presenteren we slechts ons eigen figuur, aangezien het figuur van Shin Ichi Aihara et al. [1] precies hetzelfde is, omdat ze precies dezelfde data gebruikt hebben om de logprijs als input voor hun model te berekenen.



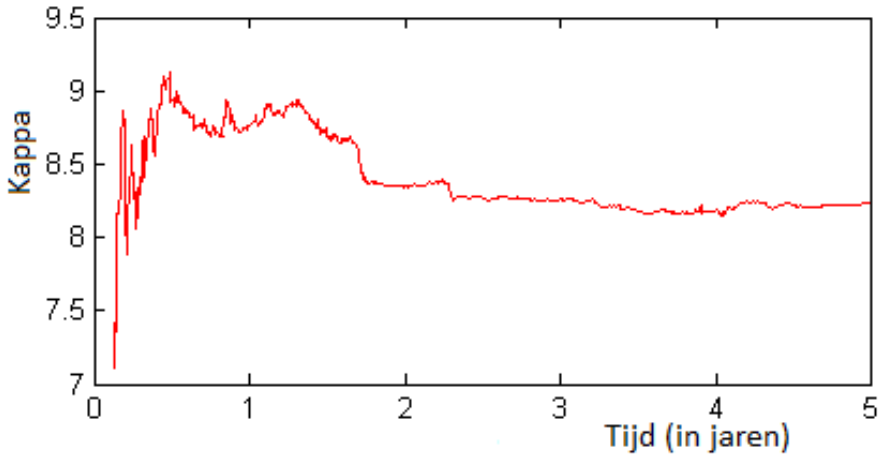
Figuur 6.18: Geschatte volatiliteit door Shin Ichi Aihara et al. [1] van de AEX-index vanaf jan 2000



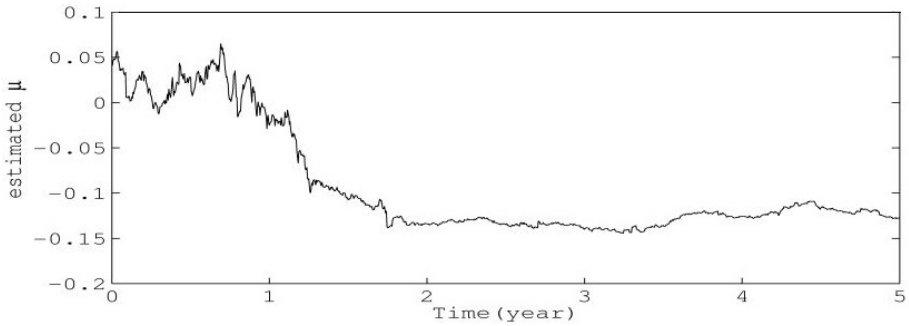
Figuur 6.19: Onze geschatte volatiliteit van de AEX-index vanaf jan 2000



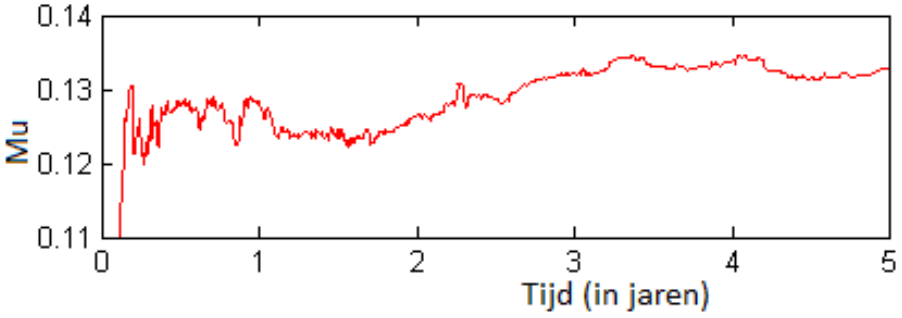
Figuur 6.20: Geschatte κ door Shin Ichi Aihara et al. [1] van de AEX-index vanaf jan 2000



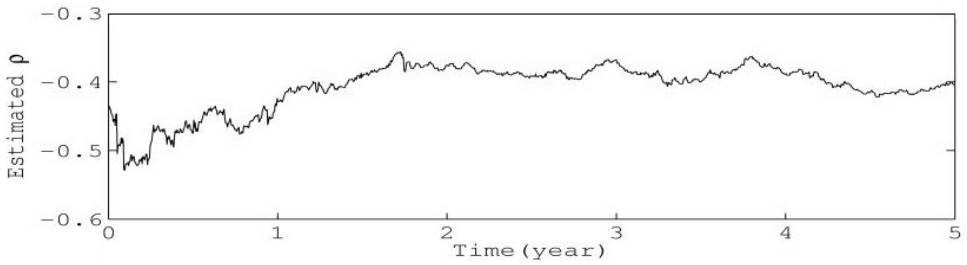
Figuur 6.21: Onze geschatte κ van de AEX-index vanaf jan 2000



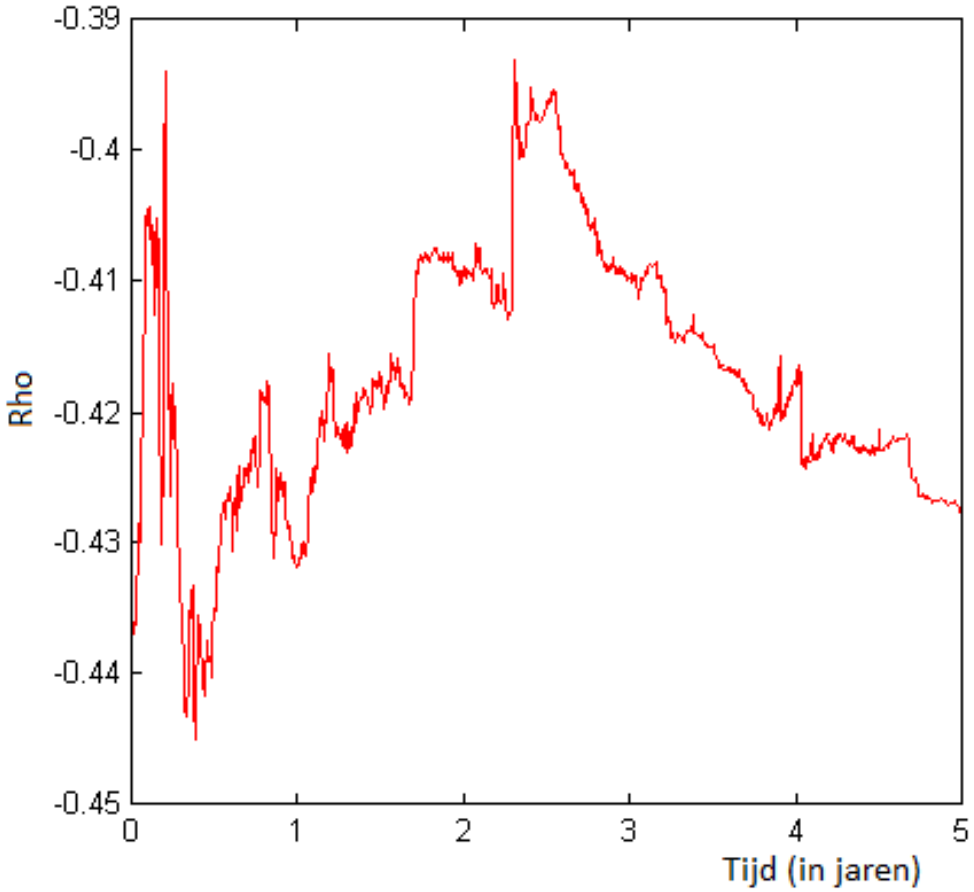
Figuur 6.22: Geschatte μ door Shin Ichi Aihara et al. [1] van de AEX-index vanaf jan 2000



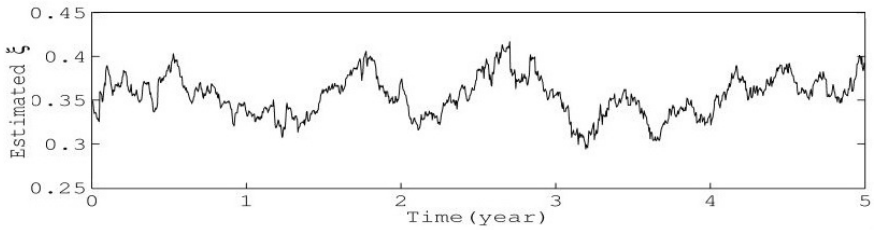
Figuur 6.23: Onze geschatte μ van de AEX-index vanaf jan 2000



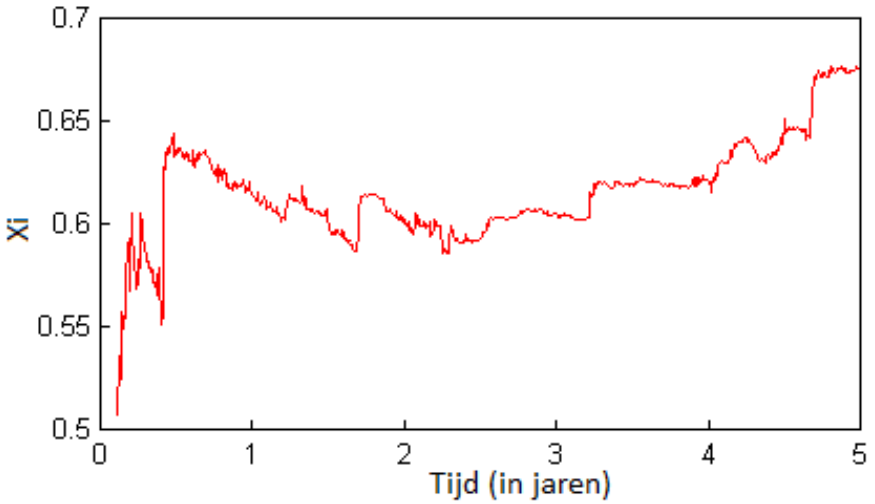
Figuur 6.24: Geschatte ρ door Shin Ichi Aihara et al. [1] van de AEX-index vanaf jan 2000



Figuur 6.25: Onze geschatte ρ van de AEX-index vanaf jan 2000



Figuur 6.26: Geschatte ξ door Shin Ichi Aihara et al. [1] van de AEX-index vanaf jan 2000



Figuur 6.27: Onze geschatte ξ van de AEX-index vanaf jan 2000

Voordat we de resultaten vergelijken merken we op dat onze simulatie gedaan is met 500 particles om de rekentijd in te korten en dat Shin Ichi Aihara et al. [1] er 2000 gebruiken per 'augmented state'. Hierna zullen we nog resultaten bespreken van een simulatie met 2000 particles.

Als we de resultaten vergelijken zien we direct dat onze schatting van de volatiliteit sterk overeen komt met die van Shin Ichi Aihara et al. [1]. De pieken zitten op ongeveer dezelfde plaatsen en schatten over het algemeen ongeveer dezelfde waarden voor de volatiliteit. Alleen in het begin wijken ze nog een beetje van elkaar af.

Als we naar de schattingen van κ kijken, zien we dat deze minder goed overeen komen. Het verloop is bij ons heel anders en bij ons stelt κ zich op den duur in op een evenwicht rond de 8.25, dit doet hij bij de simulatie van Shin Ichi Aihara et al. [1] niet.

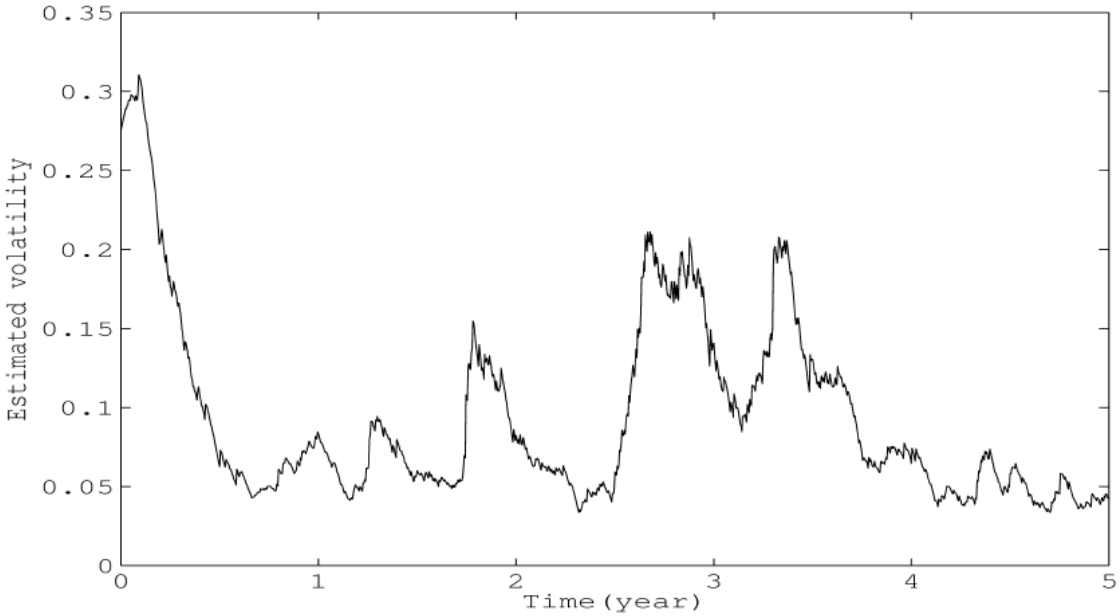
Ook onze resultaten van μ wijken sterk af van de simulatie van Shin Ichi Aihara et al. [1]. Waar hij bij ons zich op een positief evenwicht instelt, doet hij dit bij de simulatie van Shin Ichi Aihara et al. [1] op een negatief evenwicht.

Als we naar de resultaten van ρ kijken, zien we dat deze enigszins vertekend zijn door de schaalaanduiding, dus het plaatje lijkt niet op dat van Shin Ichi Aihara et al. [1], de evenwichtwaarden van beide simulaties komen echter wel ongeveer overeen.

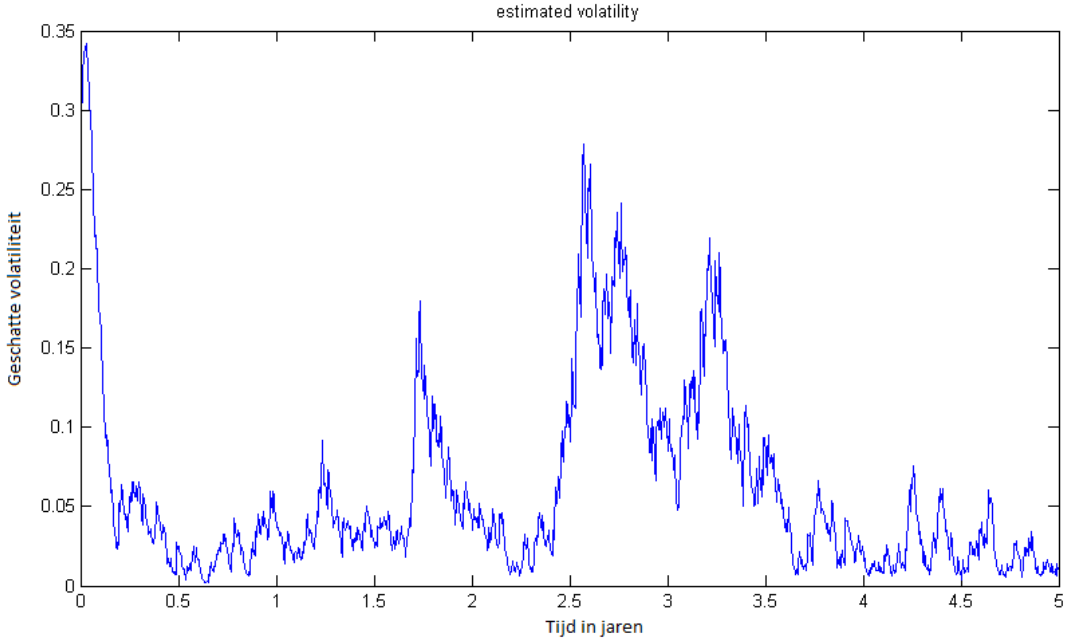
Het verloop van ξ in onze simulatie is ook anders dan bij Shin Ichi Aihara et al. [1]. Bij ons schiet hij omhoog en blijft hij daarna omhoog lopen, bij Shin Ichi Aihara et al. [1] blijft hij schommelen rond een evenwichtswaarde die niet in de buurt komt van onze geschatte waarden.

Helaas kunnen we θ niet vergelijken met het onderzoek van Shin Ichi Aihara et al. [1], aangezien zij in α de parameter $\kappa\theta$ hebben gebruikt.

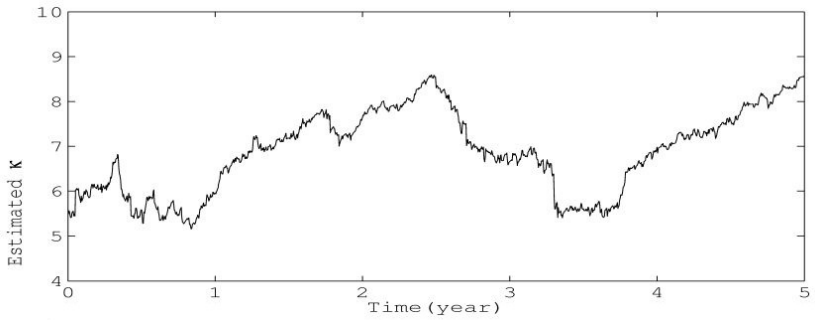
Omdat Shin Ichi Aihara et al. [1] hun model met 2000 particles hebben getest, draaien wij ook ons programma eenmaal met 2000 particles om te kijken wat voor effecten dit heeft op de schattingen van de volatiliteit en de parameters en vergeliken deze nogmaals met de resultaten van Shin Ichi Aihara et al. [1]. De rest van de instellingen van het programma houden we hetzelfde. Wederom is tijdstip 0 3 januari 2000 en tijdstip 5 3 januari 2005.



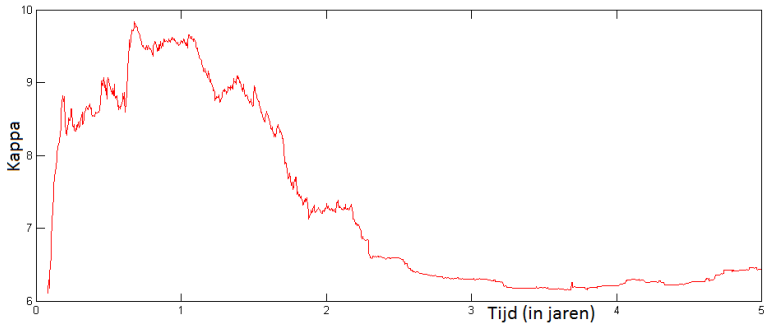
Figuur 6.28: Geschatte volatiliteit door Shin Ichi Aihara et al. [1] van de AEX-index vanaf jan 2000



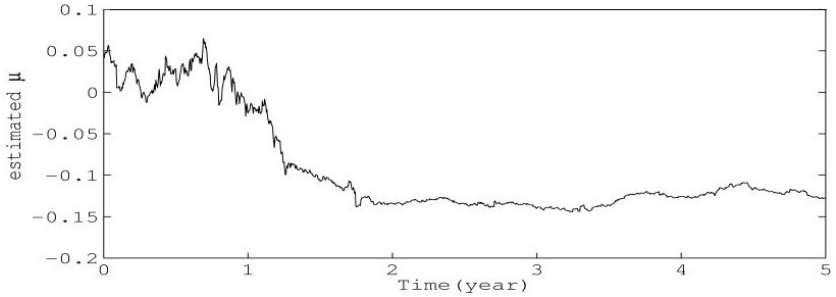
Figuur 6.29: Onze geschatte volatiliteit van de AEX-index vanaf jan 2000 met 2000 particles



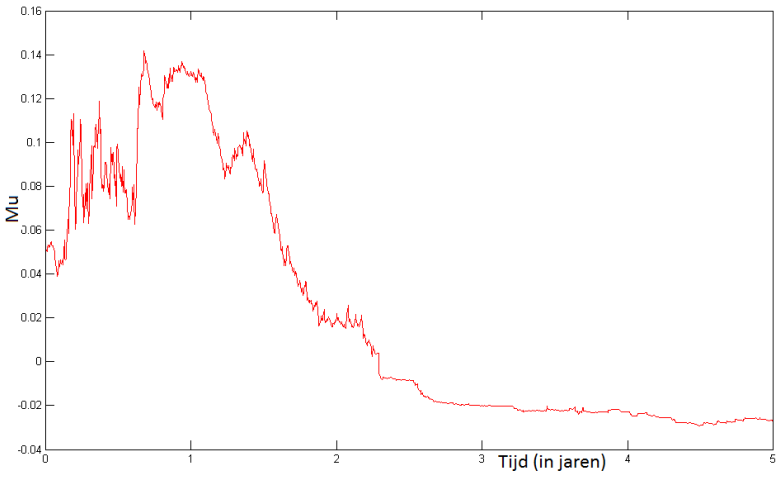
Figuur 6.30: Geschatte κ door Shin Ichi Aihara et al. [1] van de AEX-index vanaf jan 2000



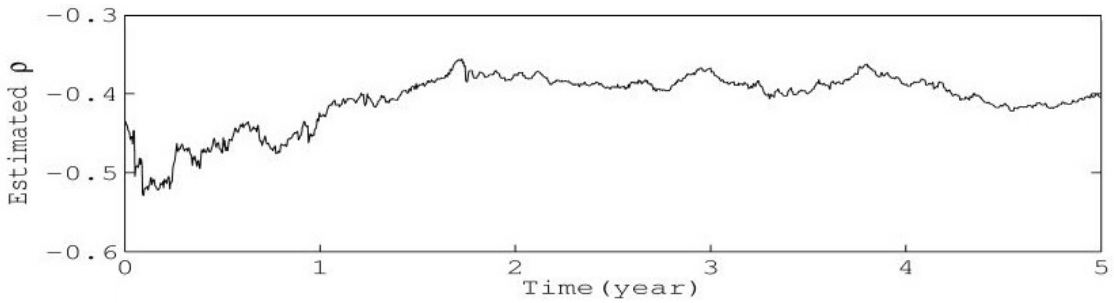
Figuur 6.31: Onze geschatte κ van de AEX-index vanaf jan 2000 met 2000 particles



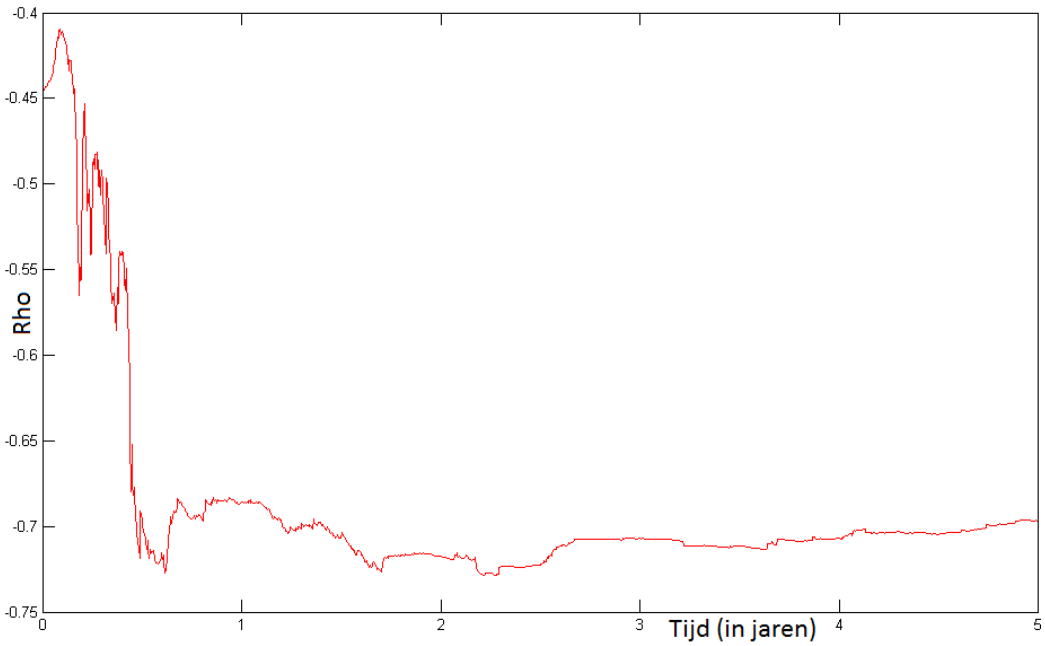
Figuur 6.32: Geschatte μ door Shin Ichi Aihara et al. [1] van de AEX-index vanaf jan 2000



Figuur 6.33: Onze geschatte μ van de AEX-index vanaf jan 2000 met 2000 particles



Figuur 6.34: Geschatte ρ door Shin Ichi Aihara et al. [1] van de AEX-index vanaf jan 2000



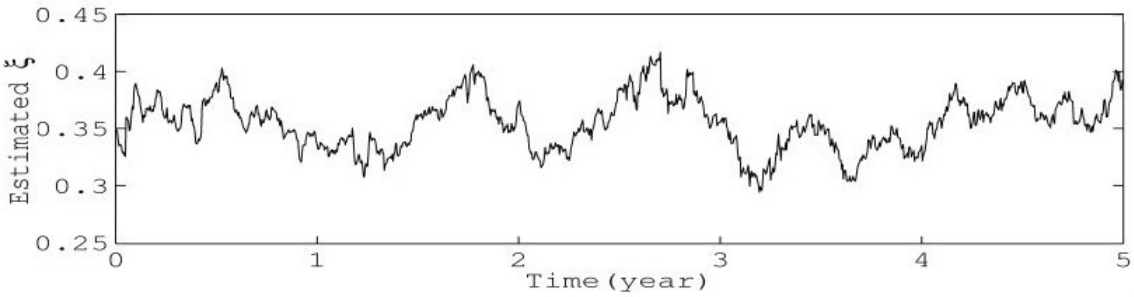
Figuur 6.35: Onze geschatte ρ van de AEX-index vanaf jan 2000 met 2000 particles

Als we naar de resultaten van de volatiliteit kijken, zien we dat onze simulatie in het begin nog steeds een beetje afwijkt van die van Shin Ichi Aihara et al. [1], maar dat na korte tijd de grafieken weer sterk op elkaar lijken. Opmerkelijk is dat de pieken van onze geschatte volatiliteit in deze simulatie hoger liggen dan in de simulatie met 500 particles en daarmee ook hoger dan die van Shin Ichi Aihara et al. [1]. Aan het einde van de vijf jaar zien we dat onze volatiliteit wat lager is dan die van Shin Ichi Aihara et al. [1].

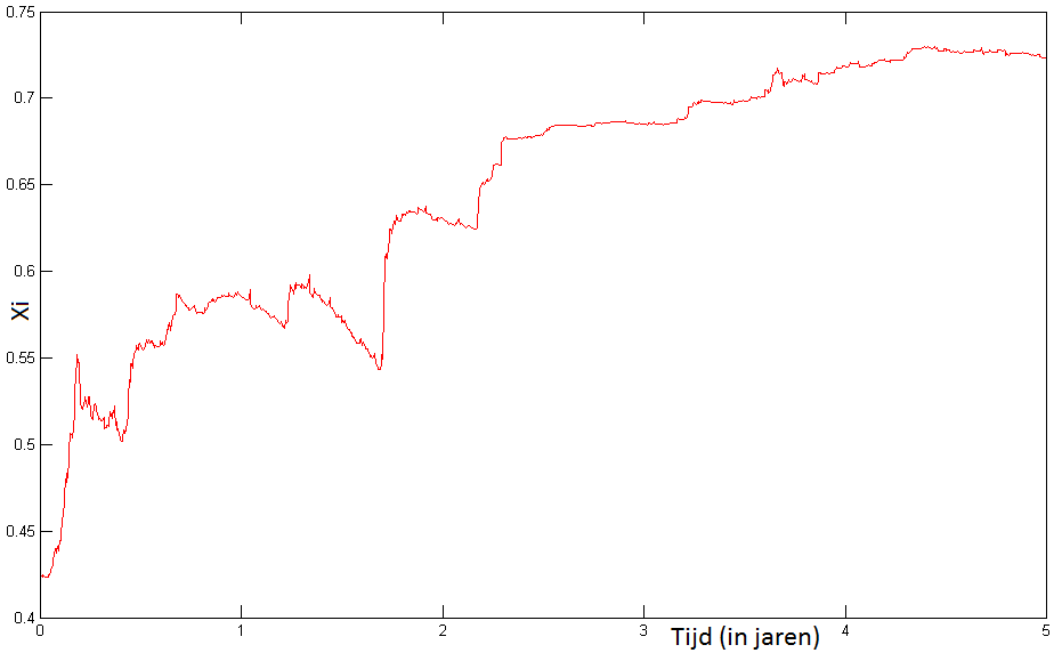
Bij κ zien we dat onze schatting ook nog steeds anders is dan die van Shin Ichi Aihara et al. [1]. In het begin gedraagt hij zich ongeveer hetzelfde als onze simulatie met 500 particles, maar al snel stelt hij zich weer in op een lager evenwicht. Dit evenwicht is ook lager dan dat van Shin Ichi Aihara et al. [1].

Als we kijken naar μ zien we dat de grafieken wel enigszins op elkaar lijken, maar dat die van ons wat grotere extreme waarden heeft. In het begin ligt de top heel wat hoger dan die van Shin Ichi Aihara et al. [1], dan verloopt hij hetzelfde als die van Shin Ichi Aihara et al. [1], maar stelt zich weer in op een lager evenwicht. Het schattingen van μ zijn in deze simulatie wel sterk anders dan in de simulatie met 500 particles.

Waar onze schatting van ρ in onze simulatie met 500 particles nog redelijk leek op de schatting van ρ van Shin Ichi Aihara et al. [1], wijkt hij er in deze simulatie met 2000 particles sterk van af. Hij neemt in het begin sterk af, in plaats van dat hij rond een evenwicht blijft schommelen, en stelt zich daarna in op een evenwicht wat lager ligt dan de waarde van Shin Ichi Aihara et al. [1].



Figuur 6.36: Geschatte ξ door Shin Ichi Aihara et al. [1] van de AEX-index vanaf jan 2000



Figuur 6.37: Onze geschatte ξ van de AEX-index vanaf jan 2000 met 2000 particles

Over ξ kunnen we opmerken dat bij deze simulatie het verloop van onze schatting van ξ lijkt op de schatting die we gedaan hebben met 500 particles. Ook in deze simulatie wordt ξ sterk anders geschat dan door Shin Ichi Aihara et al. [1].

Wederom hebben we geen vergelijking voor θ kunnen maken.

Als we 2000 particles gebruiken, zien we dat onze schatting van de volatiliteit toch minder sterk op die van Shin Ichi Aihara et al. [1] lijkt dan als we 500 particles gebruiken. Ook zien we dat onze de schattingen van de parameters zich voor een groot deel anders gedragen dan met 500 particles, maar dat ze geen van allen significant lijken op de schattingen van de parameters van Shin Ichi Aihara et al. [1].

Hoofdstuk 7

Conclusies en aanbevelingen

In dit hoofdstuk zullen we allereerst een samenvatting presenteren van ons onderzoek en het verloop daarvan. Daarna zullen we onze conclusies presenteren en tot slot zullen we een aantal aanbevelingen voor vervolgonderzoek doen.

7.1 Samenvatting

Aan het eind van het eerste semester van collegejaar 2010-2011 kregen we de presentaties van de verschillende bacheloropdrachten te zien. Wij hadden alledrie deze opdracht als eerste keuze opgegeven toen we een opdracht uit moesten kiezen, we waren dan ook blij dat we deze opdracht kregen toebedeeld. Nadat we hadden kennisgemaakt met onze begeleiders en de opdracht ons duidelijk werd gemaakt, konden we aan de slag.

We zijn eerst begonnen met het zoeken naar literatuur over het Heston model en over particle filtering. We hebben enkele artikelen gevonden waarin ons redelijk duidelijk werd wat er werd bedoeld met deze begrippen. Toen we er vervolgens ook een duidelijke uitleg bij hebben gekregen van onze begeleiders werd ons helemaal duidelijk hoe we te werk moesten gaan.

We hebben allereerst marktdata gesimuleerd aan de hand van een vanuit ons model gegenereerde volatiliteit. Hierbij hebben we de parameters μ, θ, κ, ξ en ρ van het model bekend verondersteld. Aan de hand van deze gesimuleerde markt hebben we de volatiliteit geschat op de manier zoals beschreven in hoofdstuk 3. In het kort ging dit als volgt: om de volatiliteit te schatten maken we gebruik van particle filtering, aan de hand van de waarden van de logprijs van het aandeel en de waarden van de voorgaande schattingen van de volatiliteit trekken we N particles van de volatiliteit op elke tijdstap. Aan elk particle geven we vervolgens een gewicht, dat bepaald wordt door middel van de zogeheten 'importance function' en de 'weight update equation', dit heet 'importance sampling'. De naam 'importance sampling' zegt het al, het gewicht geeft aan hoe belangrijk een bepaald particle is. Als we vervolgens een gewogen gemiddelde nemen van de gewichten met de bijbehorende geschatte volatiliteit, krijgen we onze geschatte volatiliteit op een bepaald tijdstip. Vervolgens kunnen we deze geschatte volatiliteit vergelijken met de gegenereerde volatiliteit en met de 'Root Mean Squared Error' meten hoe goed onze schatting is.

In de praktijk bleek het voor te komen dat er een paar gewichten heel groot zijn en de meeste andere gewichten bijna nul zijn, dit verschijnsel heet 'degeneracy'. Om dit tegen te gaan kunnen we 'resampling' toepassen. Bij resampling trekken we N nieuwe particles uit onze oude particles, hierbij kan het voorkomen dat we bepaalde particles vaker trekken. De kans dat we een bepaald particle trekken is gelijk aan het bijbehorende genormaliseerde gewicht. Wanneer we klaar zijn met resamplen geven we alle nieuwe particles dezelfde gewichten, te weten $1/N$. Op deze manier zijn we minder afhankelijk van slechts een paar particles met hele grote gewichten en meer afhankelijk van een geleidelijkere verdeling van particles. We willen niet bij elke stap resamplen en we willen ook niet dat we helemaal niet resamplen, daarom stellen we een criterium aan wanneer er geresampled wordt en wanneer niet. We stellen een drempel aan het aantal de 'effective sample size', deze drempel noemen we N_{thr}. We meten of het nodig is om te resamplen als volgt, als:

$$N_{eff} = \frac{1}{\sum_{i=1}^{N} (w_k^{(i)})^2} < N_{thr},$$

dan dient er geresampled te worden en anders niet.

Bij het schatten van de volatiliteit met dit particle filter kunnen we verschillende importance functions (de 'suboptimal importance function' en de twee verschillende benaderingen van de 'optimal importance

function'), verschillende resamplemethoden (Multinomial, Residual, Stratified en Systematic) en verschillende waarden voor N_{thr} gebruiken. We hebben dan ook onderzocht welke keuzes de beste schattingen van de volatiliteit opleverden met ons algoritme.

Vervolgens hebben we ons model aangepast, opdat de parameters μ, θ, κ, ξ en ρ niet meer bekend verondersteld hoeven te worden. Het doel bij het opstellen van dit model is geweest om echte aandelen te kunnen analyseren. We gaan met het particle filter naast de volatiliteit dan ook deze parameters schatten. We gaan hiermee hetzelfde te werk als bij de volatiliteit schatten; we gebruiken particles van deze parameters. We kiezen op tijdstip nul waarden voor de parameters uit een uniforme verdeling, deze krijgen in eerste instantie allemaal het gewicht $1/N$. Later gaan we deze gewichten net als voorheen met behulp van een importance function bijwerken. Ook hier kunnen we resamplen. Dit alles gaat analoog aan het schatten van de volatiliteit.

In eerste instantie deden we dit ook weer voor een gegenereerde markt. Op deze manier zijn de echte waarden van de volatiliteit en alle parameters ons bekend en kunnen we onze schattingen dus met de echte waarden vergelijken. Zo kunnen we zien of ons algoritme een goede schatting geeft van de volatiliteit en de parameters van het model.

Vervolgens hebben we dit aangepaste model gebruikt om een echt aandeel te analyseren. Hiervoor gebruikten we de AEX-index. Omdat van echte aandelen noch de volatiliteit, noch de parameters bekend zijn, hebben we onze schattingen niet kunnen vergelijken met 'echte' waarden. Wel hebben we onze resultaten kunnen vergelijken met die van Shin Ichi Aihara et al. [1] die een soortgelijk onderzoek hebben uitgevoerd.

7.2 Conclusies

In deze paragraaf bespreken we de conclusies die we uit ons onderzoek kunnen trekken. We bespreken eerst de conclusies uit ons onderzoek met ons model met bekende parameters en dan bespreken we de conclusies die we met ons model met onbekende parameters kunnen trekken. De conclusies staan puntsgewijs genoemd, met daaronder een toelichting hoe we tot deze conclusies zijn gekomen.

- *De markt wordt naar tevredenheid gesimuleerd vanuit een gegenereerde volatiliteit*

Met ons model met bekende parameters hebben we allereerst gekeken naar onze marktsimulaties. Uit de resultaten van paragraaf 6.1.1 kunnen we concluderen dat ons model de markt naar tevredenheid simuleert vanuit een gegenereerde volatiliteit; we krijgen te zien wat we verwachten, namelijk een onregelmatig verloop van de markt.

- *Bij particle filtering kunnen we het beste de normaal verdeelde 'optimal importance function' en 500 particles gebruiken*

Vervolgens hebben we onderzocht hoe we de importance function het beste kunnen kiezen. Uit de resultaten van paragraaf 6.1.2 kunnen we concluderen dat de normaal verdeelde 'optimal importance function' het beste functioneert in ons algoritme. Gebruik van deze levert nagenoeg even accurate schattingen van de volatiliteit als bij gebruik van zijn non-centraal χ^2-verdeelde variant, maar het programma levert bij gebruik van de normaal verdeelde variant deze resultaten een stuk sneller. Ook zijn de resultaten bij gebruik van deze een stuk beter dan bij gebruik van de 'suboptimal importance function', terwijl de tijd waarin deze resultaten verkregen worden niet heel veel groter is dan de tijd die nodig is om resultaten te genereren met gebruik van de 'suboptimal importance function'.

Hoewel de kwaliteit van de schattingen van de volatiliteit bij gebruik van 100 of 500 particles niet veel verschil maakt en hoewel het programma trager wordt naar mate er meer particles gebruikt worden, trekken we de conclusie dat we het beste met 500 particles kunnen werken. Dit omdat in theorie gebruik van meer particles betere schattingen oplevert en omdat we vinden dat het bij deze test nog te vroeg is om ons al vast te pinnen op gebruik van weinig particles, wat bij latere tests grote gevolgen kan hebben.

- *Het maakt niet significant uit welke resamplemethode we gebruiken bij particle filtering, maar de Systematic methode is het beste te implementeren in ons algoritme bij gebruik van het model met onbekende parameters*

Met de gevonden beste importance function zijn we op zoek gegaan naar een beste resamplemethode. We hebben hiervoor per resamplemthode (Multinomial, Residual, Stratified en Systematic) gekeken hoe goed de schatting van de volatiliteit was. Uit de resultaten van paragraaf 6.1.3 blijkt dat bij gebruik van

iedere resamplemethode ongeveer even goede resultaten van de volatiliteit verkregen worden en dat ons programma met iedere resamplemethode nagenoeg even snel draait. Er valt dus geen overduidelijk beste of slechtste resamplemethode aan te wijzen.

Wel kunnen we de Systematic resamplemethode het beste implementeren in ons algoritme voor het schatten van de volatiliteit met ons model met onbekende parameters. Deze zullen we daarvoor dan ook gebruiken.

- *Er is geen overduidelijke beste waarde voor N_{thr}*

Nu we een beste importance function gevonden hadden en geconcludeerd hadden dat het niet uit maakte welke resamplemethode we gebruikten, zijn we gaan onderzoeken of de keuze van de waarde van N_{thr} invloed had op onze schattingen van de volatiliteit. Uit de resultaten van paragraaf 6.1.4 blijkt echter ook dat er uit de onderzochte waarden geen beste N_{thr} aan te wijzen is, omdat de schatting van de volatiliteit ongeveer even goed is en dat ook ons programma met gebruik van ieder van de waarden nagoeg even snel draait.

- *Vaker resamplen maakt het programma niet veel trager*

Bij voorgaande tests hebben we bijgehouden hoevaak er geresampled werd en hoe lang het programma er over deed om te draaien. Uit sommige resultaten lijkt het alsof het programma er langer over doet om te draaien als er vaker geresampled wordt, maar bij nader onderzoek met de MATLAB profiler blijkt dit niet aan het aantal keren resamplen te liggen, maar aan het aantal berekeningen van non-centrale χ^2-verdelingen die er gedaan moeten worden, welke onafhankelijk zijn van het aantal keren dat er geresampled wordt.

- *Het verstoren of verkeerd schatten van θ en ξ heeft de grootste invloeden op de schattingen van de volatiliteit*

Uit de resultaten van paragraaf 6.1.5 kunnen we opmaken dat het verstoren van de parameters θ en ξ de grootste negatieve invloed heeft op de kwaliteit van de schattingen van de volatiliteit. Heel verrassend is dit niet, aangezien deze parameters direct met de volatiliteit te maken hebben, als zijnde de langetermijnsvolatiliteit en de volatiliteit van de volatiliteit. Het verstoren van de andere parameters heeft weinig tot geen gevolgen op de kwaliteit van de schattingen van de volatiliteit. Hieruit concluderen we tevens dat het dus erg belangrijk is om θ en ξ in de praktijk goed te schatten met ons model met onbekende parameters, gebeurt dit namelijk niet, dan zal ook de geschatte volatiliteit van het aandeel niet erg goed zijn.

- *Met ons model met onbekende parameters kunnen we de volatiliteit van een gegenereerde markt goed schatten en op den duur zelfs bijna net zo goed als met ons model met bekende parameters*

In paragraaf 6.2.1 vergelijken we onze schattingen van de volatiliteit opgesteld met ons model met onbekende parameters met de gegenereerde volatiliteit en met onze schattingen van de volatiliteit opgesteld met ons model met bekende parameters. We zien dat vooral in het begin onze schattingen gemaakt met het model met onbekende parameters afwijken van de echte waarden van de volatiliteit, maar dat ze er al gauw op gaan lijken. De schattingen gemaakt met het model met bekende parameters schatten de volatiliteit al eerder beter. Op den duur gaan de schattingen gemaakt met het model met onbekende parameters en met het model met bekende parameters sterk op elkaar lijken. We zien wel dat de gemiddelde RMSE van de schattingen gemaakt met het model met onbekende parameters groter is dan de gemiddelde RMSE van de schattingen gemaakt met het model met bekende parameters, maar dit zal dus vooral komen doordat het model met onbekende parameters meer tijd nodig heeft om in te stellen dan het model met bekende parameters.

- *De kwaliteit van de schatting van de onbekende parameters is sterk variabel*

In paragraaf 6.2.1 zien we in de figuren dat bij die simulatie de schatting van de parameters μ, θ, κ en ξ best redelijk is op den duur. De schatting van ρ zit echter naast de echte waarde, maar stelt zich wel in op een ander evenwicht. Uit paragraaf 6.2.3 weten we dat de schatting van de parameters ook afhankelijk is van het aantal gebruikte particles. We kunnen uit dit onderzoek niet de conclusie trekken dat de parameters altijd goed geschat worden, maar wel opmerkelijk is dat ook bij verkeerde schatting van de parameters, de volatiliteit nog wel goed geschat wordt. Dit is misschien te verklaren doordat de parameters veel in vermenigvuldigingen gebruikt worden in dit model, zo kunnen ze afzonderlijk misschien wel verkeerd geschat worden, maar vermenigvuldigd toch nog de correcte waarde opleveren. Verder onderzoek hiernaar is dan ook sterk aan te bevelen.

- *Met ons model met onbekende parameters kunnen we de volatiliteit van echte aandelen goed schatten*

We hebben in paragraaf 6.2.3 onze resultaten vergeleken met die van Shin Ichi Aihara et al. [1], aangezien we bij het schatten van de volatiliteit van een echt aandeel onze resultaten niet meer kunnen vergelijken met de 'echte' waarden van de volatiliteit. We zien dat we bij het schatten van de volatiliteit van de AEX-index resultaten verkrijgen die sterk overeen komen met die van Shin Ichi Aihara et al. [1]. Shin Ichi Aihara et al [1] merken in hun onderzoek op dat ze hun geschatte waarden ook niet kunnen vergelijken met de 'echte' waarden, maar dat hun numerieke experimenten er op wijzen dat hun model accuraat werkt en dat de schattingen dus goed zijn. Ook wij hebben met onze numerieke resultaten gezien dat we met ons model met onbekende parameters de volatiliteit goed kunnen schatten en onze resultaten betreffende de schatting van de volatiliteit komen sterk overeen met die van Shin Ichi Aihara et al. [1]. Hieruit trekken wij dan ook de conclusie dat we met behulp van ons model met onbekende parameters de volatiliteit van echte aandelen goed kunnen schatten.

- *We kunnen niet concluderen of onze schattingen van de parameters van echte aandelen goed of slecht zijn*

We hebben in paragraaf 6.2.1 gezien dat onze schattingen van de parameters sterk verschillen in kwaliteit. Ook als we onze geschatte parameters van de AEX-index vergelijken met de geschatte parameters van Shin Ichi Aihara et al. [1] zien we dat deze met zowel 500 als 2000 particles sterk verschillen. Ook verschillen de resultaten van onze schattingen met 500 en 2000 particles onderling sterk. We durven aan de hand van deze resultaten niet te zeggen dat onze schattingen van de parameters van echte aandelen goed of slecht zijn. Hier is meer onderzoek naar nodig.

7.3 Aanbevelingen

We hebben met ons onderzoek nu een redelijke benadering van de volatiliteit kunnen geven. De volatiliteit helemaal juist schatten is een utopie, maar we denken wel dat er verbeteringen in de schattingen mogelijk zijn. In deze paragraaf zullen we enkele aanbevelingen voor vervolgonderzoek doen.

- *Maak de resultaten op ieder tijdstip zichtbaar voor aandeel- en optiehandelaars*

Wij hebben in ons onderzoek een manier gevonden om de volatiliteit van een aandeel goed te schatten. We hebben echter nog niet de koppeling gemaakt met programma's die gebruikt worden door aandeel- en optiehandelaars, zodat deze op ieder tijdstip de volatiliteit op hun beeldscherm te zien krijgen. Dit is volgens ons met een extentie van onze methoden wel mogelijk en wij veronderstellen het dan ook zinvol om hier meer onderzoek naar te doen.

- *Breid het onderzoek naar het optimale aantal te gebruiken particles uit*

Bij onze simulaties hebben we slechts tweemaal 2000 particles gebruikt en verder hebben we voornamelijk gewerkt met 500 particles. Theoretisch gezien leveren meer particles betere resultaten op, maar het gebruik van meer particles kost ook heel veel rekentijd. Bij gebrek aan zeer krachtige computers en tijd hebben wij onze simulaties gedraaid met relatief weinig particles. We zien wel in ons onderzoek dat onze schattingen van de volatiliteit van de AEX-index met 500 particles redelijk overeen komen met de schattingen van Shin Ichi Aihara et al. [1] die heel veel meer particles gebruikt hebben en dat onze schatting met 2000 particles niet significant beter is. Ook zien we in paragraaf 6.1.6 dat onze schatting met 500 particles over het algemeen beter is dan die met 2000 particles, wat opmerkelijk is. We veronderstellen het nuttig om dieper onderzoek te doen naar het toenemen van de kwaliteit van de schatting bij gebruik van meer particles afgewogen tegen de tijd die het programma nodig heeft om te draaien.

- *Gebruik andere methoden dan 'sequential importance sampling' om de volatiliteit te schatten*

In ons onderzoek hebben we de volatiliteit geschat met de sequentiële importance sampling methode. Er zijn echter nog meer vormen van particle filters, die wellicht betere resultaten op zullen leveren. Ook bestaan er nog geavanceerdere methoden dan particle filters om schattingen te maken van onobserveerbare toestanden, wij achten het vruchtbaar om ook hier meer onderzoek naar te doen.

- *Breid het onderzoek naar het schatten van de onbekende parameters van het model uit*

Uit ons onderzoek hebben we de conclusie getrokken dat we niet kunnen zeggen dat onze schattingen van de onbekende parameters van het model goed of slecht zijn. Hoewel de volatiliteit goed benaderd wordt, is ook een goede benadering van de parameters interessant, een slechte schatting van θ en ξ kan immers grote gevolgen hebben op de kwaliteit van de schatting van de volatiliteit, dit hebben we gezien in paragraaf 6.1.5. Wij raden dan ook aan om meer onderzoek te doen naar het schatten van deze parameters en hierbij er ook op te letten dat het schatten van afzonderlijke parameters misschien wel niet de beste manier is, omdat ze ook in vermenigvuldigingen voorkomen.

- *Neem een ander model als basis*

We zijn in ons onderzoek uitgegaan van een aangepaste versie van het Heston model. Het Heston model stamt uit 1993 en is gebaseerd op het Black-Scholes model uit 1973. Inmiddels is het 2011 en zijn er in ieder geval extenties op het Heston model, zoals het Chen model uit 1996 van Lin Chen [4]. We raden dan ook aan om meer onderzoek te doen naar het schatten van de volatiliteit met extenties van het Heston model en wellicht hele andere modellen die de samenhang van de aandeelprijs en de onderliggende volatiliteit beschrijven. Wellicht levert dat ook nog betere resultaten op.

Appendices

A Matlabcode

A.1 marktsimulatie en volatiliteit schatten met bekende parameters

Hieronder is de MATLAB-code voor het simuleren van marktdata en het schatten van de volatiliteit met bekende parameters. De importance function is de normaal verdeelde optimale benadering, en de resample methode is multinomial.

```
clear all




%set randomstream
s = RandStream('mt19937ar','Seed',e);
RandStream.setDefaultStream(s);




%preliminaries
kappa = 3;              %snelheid waarmee de volatiliteit naar theta gaat
theta = 0.1;            %gemiddelde volatiliteit in de simulatie
mu = 0.1;               %gemiddelde stijging van de logaritme van de stock price
rho = -0.2;             %correlatie tussen de volatiliteit en de stock price
xi = 0.5;               %afwijking van de volatiliteit


t = 5;
dt = 0.01;
T = t*(1/dt)+1;


lambda2 = (4*kappa*exp(-kappa*dt))/(xi*xi*(1-exp(-kappa*dt)));
d = (4*theta*kappa)/(xi*xi);


%beginwaarden
v1(1) = 0.1;
y(1) = 1;

%contrueren van v_t
for i = 2:T
    lambda = lambda2*v1(i-1,1);
    v1(i,1) = (xi^2*(1-exp(-kappa*dt))) / (4*kappa) * ncx2rnd(d,lambda);
    %v1(i) = v1(i-1) + kappa * ( theta - v1(i-1) ) * (i - (i-1)) + xi * sqrt(v1(i-1))* ...
        normrnd(0,i*dt-(i-1)*dt)
    sigma2 = 0.5*(v1(i,1)+v1(i-1,1))*dt;
    sigma = sqrt(sigma2);
    y(i,1) = y(i-1) + mu*dt - 0.25*(v1(i,1)+v1(i-1,1))*dt + rho/xi * (v1(i,1)-v1(i-1,1) - ...
        kappa*theta*dt + 0.5*kappa*(v1(i,1)+v1(i-1,1))*dt) + sqrt(1-rho^2)*normrnd(0,sigma);
end

subplot(3,1,1); plot(0:dt:t,exp(y))
title('stock price')
subplot(3,1,2); plot(0:dt:t,v1)
title('real volatility')


% subplot(2,1,1); plot(0:dt:t,v1)
% title('volatility')
% subplot(2,1,2); plot(0:dt:t,exp(y))
% title('stock price')
%
%
% savefile = 'y_and_preliminaries.mat';
% save(savefile, 'kappa', 'theta', 'mu', 'rho', 'xi', 't', 'dt', 'T', 'y', 'v1')


xis = xi*sqrt(1-rho^2);
kappas = -0.5*xi*rho+kappa;
```

```
thetas = (kappa*theta -xi*rho*mu)/(-0.5*xi*rho+kappa);
lambdas2 = (4*kappas*exp(-kappa*dt))/(xis^2*(1-exp(-kappas*dt)));
ds = (4*thetas*kappas)/(xis^2);
cs = (xis^2*(1-exp(-kappa*dt)))/(4*kappas);









tic



N = 500;
N_thr = N/3;
P=1;
q=0;

%lambda = (4*kappa*exp(-kappa*dt)) / (xi^2*(1-rho^2)*dt*(1-exp(-kappa*dt)));
%d2 = (4*theta*kappa) / (xi^2*(1-rho^2)*dt);

v2 = zeros(2,N);
v_min = 0.01;
w = ones(1,N) * 1/N;
w_squared = zeros(1,N);
w_tilde = zeros(1,N);
w_output = zeros(T,N);
%v2_0 = theta;
%y_0 = y(1);

% T                                     %Deze output is niet echt nodig, maar is tijdens ...
    het wachten fijn om te weten.

for i = 1:N
    v2(2,i) = normrnd(theta,xi);        %in deze for-loop worden de eerste particles ...
        getrokken. Deze zijn onderling onafhankelijk.
    if (v2(2,i)<0)
        v2(2,i)=0.001;
    end
    %v2(1,i) = normrnd(v2_0+kappa*(theta-v2_0)*dt+xi*rho*(y(1)-y_0) - ...
        xi*rho*(mu-0.5*v2_0)*dt,sqrt(xi^2*(1-rho^2)*v2_0*dt));
%v2(i,2) = v2(i-1,2) + kappa*(theta-v2(i-1,2))*dt + xi*rho*(y(i,1)-y(i-1,1)) - ...
    xi*rho*(mu-0.5*v2(i-1,2))*dt + xi*sqrt(1-rho^2)*sqrt(v2(i-1,2))*normrnd(0,dt);
end
V(1)=0;
sumv_squared = 0;
for i =1:N
    V(1)=V(1)+w(i)*v2(2,i);             %V staat voor de uiteindelijke geschatte ...
        volatiliteit. Hier wordt V op tijdstip 1 berekend.

end
sumv_squared = sumv_squared + (v1(1)-V(1))^2;

for i = 2:T
    v2(1,:) = v2(2,:);                  %De tijd is met dt vooruit gegaan, dus de v2 ...
        matrix verschuift alles omhoog.
    for j = 1:N
        v2(2,j) = normrnd(v2(1,j)+kappa*(theta-v2(1,j))*dt + ...
            xi*rho*(y(i)-y(i-1))-xi*rho*(mu-0.5*v2(1,j))*dt , ...
            sqrt(xi^2*(1-rho^2)*v2(1,j)*dt));            %De tweede rij van v2 wordt hier ...
            gesimuleerd.
          if (v2(2,j)<0)
             v2(2,j)=v_min;              %volatiliteit kan niet kleiner zijn dan nul.
         end
                %wat hierboven vermeld staat gebruiken we alleen bij de
                %normale benadering



%        lambdas = lambdas2*v2(1,j);
%        v2(2,j) = (cs)* ncx2rnd(ds,lambdas);
                %wat hierboven vermeld staat gebruiken we alleen bij de
```

```
128                 %chi-kwadraat benadering
129
130
131
132 %        v2(2,j) = ((xi^2*(1-exp(-kappa*dt))) / (4*kappa))* ncx2rnd(d,lambda);
133                 %wat hierboven vermeld staat gebruiken we alleen bij de
134                 %suboptimale benadering
135
136
137
138
139         mu_a = y(i-1)+mu*dt - 1/4*(v2(2,j)+v2(1,j))*dt + rho/xi * (v2(2,j)-v2(1,j) - ...
                kappa*theta*dt+ kappa*1/2*(v2(2,j)+v2(1,j))*dt);
140         sigma_a = sqrt(1/2*(v2(2,j)+v2(1,j))*dt*(1-rho^2));
141                   %deze gebruiken we in alle gevallen
142
143
144
145
146          mu_c = v2(1,j)+kappa*(theta-v2(1,j))*dt + xi*rho * (y(i)-y(i-1)) - xi*rho * ...
                (mu-1/2*v2(1,j))*dt;
147         sigma_c = sqrt(xi^2*(1-rho^2)*v2(1,j)*dt); %!!!!!!!!!!!!!!!!!
148         constante_b = (xi^2*(1-exp(-kappa*dt)))/(4*kappa); %!!!!!!!!!!!!!!!!!!
149         d_b = (4*theta*kappa)/(xi^2);
150         lambda_b = (4*kappa*exp(-kappa*dt)) / (xi^2*(1-exp(-kappa*dt)))*v2(1,j);
151                   %deze gebruiken we bij normaal en chi-kwadraat en niet
152                   %bij suboptimaal
153
154
155         a = normpdf(y(i),mu_a,sigma_a);
156         if (a<=0)
157             a=v_min;
158         end
159                   %deze gebruiken we altijd
160
161
162        b = (1/constante_b) * ncx2pdf(v2(2,j) / constante_b,d_b,lambda_b);
163                    %deze gebruiken we bij normaal en chi-kwadraat en niet
164                   %bij suboptimaal
165
166
167
168         c = normpdf(v2(2,j),mu_c,sigma_c);
169         if (c<=0)
170             c=v_min;
171         end
172                   %deze gebruiken we alleen bij normaal
173
174
175 %         c = (1/cs) * ncx2pdf(v2(2,j)/cs,ds,lambdas);
176                   %deze gebruiken we alleen bij chi-kwadraat
177
178
179         kip = a*b/c;                    %kip staat hier voor de breuk van kansdichtheden ...
                die we gebruiken om de gewichten te berekenen van de nieuwe tijd.
180                   %deze gebruiken we bij normaal en chi-kwadraat
181
182 %         kip=a;     % Suboptimal choice for importance function
183                   %deze gebruiken we bij suboptimaal.
184
185            w_tilde(j) = w(j) * kip;
186     end
187     for j = 1:N
188         w(j) = w_tilde(j)/sum(w_tilde); %De gewichten worden hier genormaliseerd.
189     end
190     for j = 1:N
191         w_squared(j)=w(j)^2;
192     end
193     N_eff = 1/sum(w_squared);           %Hier begint het resampling proces.
194     if (N_eff<N_thr)
195         q=q+1;
196         display('er wordt geresampled')
197         v2(1,:) = randsample(v2(2,:),N,true,w);
198         v2(2,:)=v2(1,:);
199         w = ones(1,N) * 1/N;
```

```
200     end
201
202
203
204     V(i) = 0;
205     for j = 1:N
206         V(i) = V(i) + w(j) * v2(2,j);   %De uiteindelijke geschatte volatiliteit wordt ...
                hier berekend.
207     end
208     sumv_squared = sumv_squared + (v1(i)-V(i))^2;
209
210 end
211 RMSE = sqrt(sumv_squared/T);
212
213
214
215
216  subplot(3,1,3);
217  plot(0:dt:t,v1,'b',0:dt:t,V,'r')
218  title('estimated volatility in red')
219
220
221
222 t1=toc;
```

A.2 marktsimulatie en volatiliteit schatten met onbekende parameters

Hieronder is de MATLAB-code voor het simuleren van marktdata en het schatten van de volatiliteit met onbekende parameters. De importance function is de normaal verdeelde optimale benadering, en de resample methode is multinomial.

```
1  clear all
2  %tic
3
4
5
6  %set randomstream
7  s = RandStream('mt19937ar','Seed',e);
8  RandStream.setDefaultStream(s);
9
10 %preliminaries
11 kappa = 3;              %snelheid waarmee de volatiliteit naar theta gaat
12 theta = 0.1;            %gemiddelde volatiliteit in de simulatie
13 mu = 0.1;               %gemiddelde stijging van de logaritme van de stock price
14 rho = -0.2;             %correlatie tussen de volatiliteit en de stock price
15 xi = 0.5;               %afwijking van de volatiliteit
16
17
18 t = 5;
19 dt = 0.01;
20 T = t*(1/dt)+1;
21
22
23 lambda2 = (4*kappa*exp(-kappa*dt))/(xi*xi*(1-exp(-kappa*dt)));
24 d = (4*theta*kappa)/(xi*xi);
25
26
27 %beginwaarden
28 v1(1) = 0.1;
29 y(1) = 1;
30
31 %contrueren van v_t
32 for i = 2:T
33     lambda = lambda2*v1(i-1,1);
34     v1(i,1) = (xi^2*(1-exp(-kappa*dt))) / (4*kappa) * ncx2rnd(d,lambda);
35     %v1(i) = v1(i-1) + kappa * ( theta - v1(i-1) ) * (i - (i-1)) + xi * sqrt(v1(i-1))* ...
           normrnd(0,i*dt-(i-1)*dt)
36     sigma2 = 0.5*(v1(i,1)+v1(i-1,1))*dt;
37     sigma = sqrt(sigma2);
```

```
    y(i,1) = y(i-1) + mu*dt - 0.25*(v1(i,1)+v1(i-1,1))*dt + rho/xi * (v1(i,1)-v1(i-1,1) - ...
        kappa*theta*dt + 0.5*kappa*(v1(i,1)+v1(i-1,1))*dt) + sqrt(1-rho^2)*normrnd(0,sigma);
end

subplot(3,1,1); plot(0:dt:t,exp(y))
title('stock price')
subplot(3,1,2); plot(0:dt:t,v1)
title('real volatility')









xis = xi*sqrt(1-rho^2);
kappas = -0.5*xi*rho+kappa;
thetas = (kappa*theta -xi*rho*mu)/(-0.5*xi*rho+kappa);
lambdas2 = (4*kappas*exp(-kappa*dt))/(xis^2*(1-exp(-kappas*dt)));
ds = (4*thetas*kappas)/(xis^2);
cs = (xis^2*(1-exp(-kappa*dt)))/(4*kappas);



sigma_theta = 0.01;
sigma_rho = 0.02;
sigma_xi = 0.05;
sigma_kappa = 0.3;
sigma_mu = 0.01;



restr_theta = 0.0001;
restr_rho = 0.001;
restr_xi = 0.001;
restr_kappa = 0.001;
restr_mu = 0.001;



teller1=0;
teller2_1=0;
teller2_2=0;
teller3=0;
teller4=0;
teller5=0;
teller6=0;
resampnumber = zeros(1,T);


tic;


N = 500;
N_thr = N/3;
resampteller=0;
alpha = zeros(5,N);
ALPHA=zeros(5,T);
%lambda = (4*kappa*exp(-kappa*dt)) / (xi^2*(1-rho^2)*dt*(1-exp(-kappa*dt)));
%d2 = (4*theta*kappa) / (xi^2*(1-rho^2)*dt);

v2 = zeros(2,N);
v_min = 0.01;
w = ones(1,N) * 1/N;
w_squared = zeros(1,N);
w_tilde = zeros(1,N);
w_output = zeros(T,N);
%v2_0 = theta;
%y_0 = y(1);
```

```
112
113 % T                                    %Deze output is niet echt nodig, maar is tijdens ...
        het wachten fijn om te weten.
114
115
116 % tic
117
118
119
120
121
122 % alpha_1=0.05 + 0.1*rand;
123 % alpha_2=-0.1-0.2*rand;
124 % alpha_3=0.25 + 0.5*rand;
125 % alpha_4=1.5 + 3*rand;
126 % alpha_5=0.05 + 0.1*rand;
127
128
129
130
131
132
133
134
135
136
137 for i = 1:N
138     alpha(1,i) = 0.01 + 2*rand;      %theta tussen 0.05 en 0.5
139     alpha(2,i) = -0.5 + 0.5*rand;             %rho tussen -0.8 en 0
140     alpha(3,i) = 0.01 + 0.9*rand;       %xi tussen 0.01 en 0.91
141     alpha(4,i) = 1 + 8*rand;            %kappa tussen 1 en 10
142     alpha(5,i) = 0.05 + 0.45*rand;      %mu tussen 0.05 en 0.3
143      v2(2,i) = normrnd(0.27,0.02);        %in deze for-loop worden de eerste particles ...
             getrokken. Deze zijn onderling onafhankelijk.
144     if (v2(2,i)<=0)
145         v2(2,i)=0.001;
146     end
147
148     %v2(1,i) = normrnd(v2_0+kappa*(theta-v2_0)*dt+xi*rho*(y(1)-y_0) - ...
            xi*rho*(mu-0.5*v2_0)*dt,sqrt(xi^2*(1-rho^2)*v2_0*dt));
149 %v2(i,2) = v2(i-1,2) + kappa*(theta-v2(i-1,2))*dt + xi*rho*(y(i,1)-y(i-1,1)) - ...
        xi*rho*(mu-0.5*v2(i-1,2))*dt + xi*sqrt(1-rho^2)*sqrt(v2(i-1,2))*normrnd(0,dt);
150 end
151 V(1)=0;
152 sumv_squared = 0;
153 alpha_1=0;
154 alpha_2=0;
155 alpha_3=0;
156 alpha_4=0;
157 alpha_5=0;
158 for i =1:N
159     V(1)=V(1)+w(i)*v2(2,i);             %V staat voor de uiteindelijke geschatte ...
            volatiliteit. Hier wordt V op tijdstip 1 berekend.
160     alpha_1=alpha_1+w(i)*alpha(1,i);
161     alpha_2=alpha_2+w(i)*alpha(2,i);
162     alpha_3=alpha_3+w(i)*alpha(3,i);
163     alpha_4=alpha_4+w(i)*alpha(4,i);
164     alpha_5=alpha_5+w(i)*alpha(5,i);
165 end
166 sumv_squared = sumv_squared + (v1(1)-V(1))^2;
167 ALPHA(1,1)=alpha_1;
168 ALPHA(2,1)=alpha_2;
169 ALPHA(3,1)=alpha_3;
170 ALPHA(4,1)=alpha_4;
171 ALPHA(5,1)=alpha_5;
172
173
174 for i = 2:T
175     v2(1,:) = v2(2,:);                  %De tijd is met dt vooruit gegaan, dus de v2 ...
            matrix verschuift alles omhoog.
176     for j = 1:N
177
178 %         lambdas = lambdas2*v2(1,j);
179         alpha(1,j) = alpha(1,j)+normrnd(0,sigma_theta/sqrt(i-1));
180         alpha(2,j) = alpha(2,j)+normrnd(0,sigma_rho/sqrt(i-1));
```

```
181         alpha(3,j) = alpha(3,j)+normrnd(0,sigma_xi/sqrt(i-1));
182         alpha(4,j) = alpha(4,j)+normrnd(0,sigma_kappa/sqrt(i-1));
183         alpha(5,j) = alpha(5,j)+normrnd(0,sigma_mu/sqrt(i-1));
184         if alpha(1,j)<=restr_theta
185             alpha(1,j)=restr_theta;
186             teller1=teller1+1
187         end
188
189         if alpha(2,j)>=1-restr_rho
190             alpha(2,j)=1-restr_rho;
191             teller2_1=teller2_1+1
192         end
193
194
195         if alpha(2,j)<=-1+restr_rho
196             alpha(2,j)=-1+restr_rho;
197             teller2_2=teller2_2+1
198         end
199
200
201         if alpha(3,j)<=restr_xi
202             alpha(3,j)=restr_xi;
203             teller3=teller3+1
204         end
205         if alpha(4,j)<=restr_kappa
206             alpha(4,j)=restr_kappa;
207             teller4=teller4+1
208         end
209         %v2(2,j) = (cs)* ncx2rnd(ds,lambdas);
210         v2(2,j) = normrnd(v2(1,j)+alpha(4,j)*(alpha(1,j)-v2(1,j))*dt + ...
              alpha(3,j)*alpha(2,j)*(y(i)-y(i-1)) - ...
              alpha(3,j)*alpha(2,j)*(alpha(5,j)-0.5*v2(1,j))*dt  ,  ...
              sqrt(alpha(3,j)^2*(1-alpha(2,j)^2)*v2(1,j)*dt));            %De tweede rij van ...
              v2 wordt hier gesimuleerd.
211
212         if (v2(2,j)<=0)
213             v2(2,j)=v_min;              %volatiliteit kan niet kleiner zijn dan nul.
214         end
215         mu_a = y(i-1)+alpha(5,j)*dt - 1/4*(v2(2,j)+v2(1,j))*dt + alpha(2,j)/alpha(3,j) * ...
              (v2(2,j)-v2(1,j)-alpha(4,j)*alpha(1,j)*dt + alpha(4,j)*1/2*(v2(2,j)+v2(1,j))*dt);
216         sigma_a = sqrt(1/2*(v2(2,j)+v2(1,j))*dt*(1-alpha(2,j)^2));
217                    %deze gebruiken we in alle gevallen
218
219
220
221
222         mu_c = v2(1,j)+alpha(4,j)*(alpha(1,j)-v2(1,j))*dt + alpha(3,j)*alpha(2,j) * ...
              (y(i)-y(i-1)) - alpha(3,j)*alpha(2,j) * (alpha(5,j)-1/2*v2(1,j))*dt;
223         sigma_c = sqrt(alpha(3,j)^2*(1-alpha(2,j)^2)*v2(1,j)*dt);
224         constante_b = (alpha(3,j)^2*(1-exp(-alpha(4,j)*dt)))/(4*alpha(4,j));
225         d_b = (4*alpha(1,j)*alpha(4,j))/(alpha(3,j)^2);
226         lambda_b = (4*alpha(4,j)*exp(-alpha(4,j)*dt)) / ...
              (alpha(3,j)^2*(1-exp(-alpha(4,j)*dt)))*v2(1,j);
227
228         a = normpdf(y(i),mu_a,sigma_a);
229
230         b = (1/constante_b) * ncx2pdf(v2(2,j)/constante_b,d_b,lambda_b);
231         c = normpdf(v2(2,j),mu_c,sigma_c);
232         if a==0
233             ln_a=-100000;
234         else
235             ln_a=log(a);
236         end
237
238
239         if b==0
240             ln_b=-100000;
241         else
242             ln_b=log(b);
243         end
244
245         if c==0
246             ln_c=-100000;
247         else
248             ln_c=log(c);
```

```
249         end
250
251         kip=exp(ln_a+ln_b-ln_c);
252
253         if kip≥1000
254             kip=0;
255         end
256
257
258
259
260
261
262
263 %          kip = a*b/c;                    %kip staat hier voor de breuk van ...
        kansdichtheden die we gebruiken om de gewichten te berekenen van de nieuwe tijd.
264 %         if kip > 1000
265 %            kip=0;
266 %         end
267
268
269
270 %         kip = a*b/c;                    %kip staat hier voor de breuk van kansdichtheden ...
        die we gebruiken om de gewichten te berekenen van de nieuwe tijd.
271 %         if kip > 1000
272 %            kip=0;
273 %         end
274
275         w_tilde(j) = w(j) * kip;
276     end
277     for j = 1:N
278         w(j) = w_tilde(j)/sum(w_tilde); %De gewichten worden hier genormaliseerd.
279     end
280     for j = 1:N
281         w_squared(j)=w(j)^2;
282     end
283
284     V(i) = 0;
285     for j = 1:N
286         V(i) = V(i) + w(j) * v2(2,j);   %De uiteindelijke geschatte volatiliteit wordt ...
              hier berekend.
287         ALPHA(:,i) = ALPHA(:,i) + w(j)*alpha(:,j);
288     end
289
290     sumv_squared = sumv_squared + (v1(i)-V(i))^2;
291
292
293     N_eff = 1/sum(w_squared);           %Hier begint het resampling proces.
294     if (N_eff<N_thr)
295         display('er wordt geresampled')
296         [v_and_alpha] = resampleSystematic2([v2(2,:);alpha], w );
297         v2(2,:) = v_and_alpha(1,:);
298         v_and_alpha(1,:) = [];
299         alpha = v_and_alpha;
300         w = ones(1,N) * 1/N;
301         resampteller=resampteller+1;
302         resampnumber(i) = resampteller;
303     end
304
305
306
307 end
308 RMSE = sqrt(sumv_squared)*(1/sqrt(T));
309
310 %k/M
311
312 resampnumber;
313 resampteller;
314 toc;
315 t_end=toc;
316
317
318 end
319 %
320 subplot(3,1,3);
```

```
plot(0:dt:t,v1,'b',0:dt:t,V,'r')
title('estimated volatility in red')

thetaline = theta*ones(T,1);
rholine = rho*ones(T,1);
xiline = xi*ones(T,1);
kappaline = kappa*ones(T,1);
muline = mu*ones(T,1);


figure(2)
plot(0:dt:t,thetaline,'b',0:dt:t,ALPHA(1,:),'r')
title('estimated theta in red')
figure(3)
plot(0:dt:t,rholine,'b',0:dt:t,ALPHA(2,:),'r')
title('estimated rho in red')
figure(4)
plot(0:dt:t,xiline,'b',0:dt:t,ALPHA(3,:),'r')
title('estimated xi in red')
figure(5)
plot(0:dt:t,kappaline,'b',0:dt:t,ALPHA(4,:),'r')
title('estimated kappa in red')
figure(6)
plot(0:dt:t,muline,'b',0:dt:t,ALPHA(5,:),'r')
title('estimated mu in red')
```

A.3 echte marktdata en volatiliteit schatten

Hieronder is de MATLAB-code voor het downloaden van echte marktdata en het schatten van de volatiliteit. In deze code is de AEX index gedownload. Uiteraard zijn de parameters onbekend. De importance function is de normaal verdeelde optimale benadering, en de resample methode is multinomial.

```
clear all
tic
%set randomstream
e=1234
s = RandStream('mt19937ar','Seed',e);
RandStream.setDefaultStream(s);

% %preliminaries
% kappa = 3;              %snelheid waarmee de volatiliteit naar theta gaat
% theta = 0.1;            %gemiddelde volatiliteit in de simulatie
% mu = 0.1;               %gemiddelde stijging van de logaritme van de stock price
% rho = -0.2;             %correlatie tussen de volatiliteit en de stock price
% xi = 0.5;               %afwijking van de volatiliteit

Y='01011991';
Z='31052011';
CMP=hist_stock_data(Y,Z,'^AEX'); %Download stock data from from Yahoo! Finance
Edata=flipud(CMP.AdjClose);

size_Edata=size(Edata);
T=size_Edata(1,1);
dt=1/252;
t=(T-1)*dt;

% t = 504;
% dt = 1;
% T = t*(1/dt)+1;


% Edata2=Edata(1:505) %take the first two years


y=log(Edata);

% for i=1:T
%
%     y(i)=log(Edata2(i));
% end
```

```
39
40
41  %
42  % lambda2 = (4*kappa*exp(-kappa*dt))/(xi*xi*(1-exp(-kappa*dt)));
43  % d = (4*theta*kappa)/(xi*xi);
44
45
46  % %beginwaarden
47  % v1(1) = 0.1;
48  % y(1) = 1;
49
50  % %contrueren van v_t
51  % for i = 2:T
52  %     lambda = lambda2*v1(i-1,1);
53  %     v1(i,1) = (xi^2*(1-exp(-kappa*dt))) / (4*kappa) * ncx2rnd(d,lambda);
54  %     %v1(i) = v1(i-1) + kappa * ( theta - v1(i-1) ) * (i - (i-1)) + xi * sqrt(v1(i-1))* ...
        normrnd(0,i*dt-(i-1)*dt)
55  %     sigma2 = 0.5*(v1(i,1)+v1(i-1,1))*dt;
56  %     sigma = sqrt(sigma2);
57  %     y(i,1) = y(i-1) + mu*dt - 0.25*(v1(i,1)+v1(i-1,1))*dt + rho/xi * (v1(i,1)-v1(i-1,1) ...
        - kappa*theta*dt + 0.5*kappa*(v1(i,1)+v1(i-1,1))*dt) + sqrt(1-rho^2)*normrnd(0,sigma);
58  % end
59  %
60  % subplot(3,1,2); plot(0:dt:t,v1)
61  % title('real volatility')
62  % toc
63
64  % subplot(2,1,1); plot(0:dt:t,v1)
65  % title('volatility')
66  % subplot(2,1,2); plot(0:dt:t,exp(y))
67  % title('stock price')
68  %
69  %
70  % savefile = 'y_and_preliminaries.mat';
71  % save(savefile, 'kappa', 'theta', 'mu', 'rho', 'xi', 't', 'dt', 'T', 'y', 'v1')
72
73
74  % xis = xi*sqrt(1-rho^2);
75  % kappas = -0.5*xi*rho+kappa;
76  % thetas = (kappa*theta -xi*rho*mu)/(-0.5*xi*rho+kappa);
77  % lambdas2 = (4*kappas*exp(-kappa*dt))/(xis^2*(1-exp(-kappas*dt)));
78  % ds = (4*thetas*kappas)/(xis^2);
79  % cs = (xis^2*(1-exp(-kappa*dt)))/(4*kappas);
80
81
82
83  sigma_theta = 0.01;
84  sigma_rho = 0.02;
85  sigma_xi = 0.05;
86  sigma_kappa = 0.3;
87  sigma_mu = 0.01;
88
89
90
91  restr_theta = 0.0001;
92  restr_rho = 0.001;
93  restr_xi = 0.00001;
94  restr_kappa = 0.001;
95  restr_mu = 0.001;
96
97
98
99
100 teller1=0
101 teller2_1=0
102 teller2_2=0
103 teller3=0
104 teller4=0
105 teller5=0
106 teller6=0
107 resampnumber = zeros(1,T);
108
109
110
111
```

```
112
113 N = 500;
114 N_thr = 2*N/3;
115 resampteller=0;
116 alpha = zeros(5,N);
117 ALPHA=zeros(5,T);
118 %lambda = (4*kappa*exp(-kappa*dt)) / (xi^2*(1-rho^2)*dt*(1-exp(-kappa*dt)));
119 %d2 = (4*theta*kappa) / (xi^2*(1-rho^2)*dt);
120
121 v2 = zeros(2,N);
122 v_min = 0.01;
123 w = ones(1,N) * 1/N;
124 w_squared = zeros(1,N);
125 w_tilde = zeros(1,N);
126 % w_output = zeros(T,N);
127 %v2_0 = theta;
128 %y_0 = y(1);
129
130 % T                                       %Deze output is niet echt nodig, maar is tijdens ...
      het wachten fijn om te weten.
131
132
133 % alpha_1=0.05 + 1.15*rand;
134 % alpha_2=-0.8 + 0.7*rand;
135 % alpha_3=0.1 + 0.5*rand;
136 % alpha_4=1 + 9*rand;
137 % alpha_5=-0.2 + 0.5*rand;
138 % v2_start=normrnd(0.27,0.02);
139 %
140 %
141 % ALPHA(1,1)=alpha_1;
142 % ALPHA(2,1)=alpha_2;
143 % ALPHA(3,1)=alpha_3;
144 % ALPHA(4,1)=alpha_4;
145 % ALPHA(5,1)=alpha_5;
146
147
148
149
150
151
152 tic
153 for i = 1:N
154     alpha(1,i) = 0.01 + 2*rand;      %theta tussen 0.05 en 0.5
155     alpha(2,i) = -0.5 + 0.5*rand;             %rho tussen -0.8 en 0
156     alpha(3,i) = 0.01 + 0.9*rand;       %xi tussen 0.01 en 0.91
157     alpha(4,i) = 1 + 8*rand;            %kappa tussen 1 en 10
158     alpha(5,i) = 0.05 + 0.45*rand;      %mu tussen 0.05 en 0.3
159      v2(2,i) = normrnd(0.27,0.02);        %in deze for-loop worden de eerste particles ...
           getrokken. Deze zijn onderling onafhankelijk.
160
161     if (v2(2,i)<=0)
162         v2(2,i)=0.001;
163     end
164     %v2(1,i) = normrnd(v2_0+kappa*(theta-v2_0)*dt + xi*rho*(y(1)-y_0) - ...
          xi*rho*(mu-0.5*v2_0)*dt , sqrt(xi^2*(1-rho^2)*v2_0*dt));
165 %v2(i,2) = v2(i-1,2) + kappa*(theta-v2(i-1,2))*dt + xi*rho*(y(i,1)-y(i-1,1)) - ...
      xi*rho*(mu-0.5*v2(i-1,2))*dt + xi*sqrt(1-rho^2)*sqrt(v2(i-1,2))*normrnd(0,dt);
166 end
167 V(1)=0;
168 % sumv_squared = 0;
169 alpha_1=0;
170 alpha_2=0;
171 alpha_3=0;
172 alpha_4=0;
173 alpha_5=0;
174 for i =1:N
175     V(1)=V(1)+w(i)*v2(2,i);             %V staat voor de uiteindelijke geschatte ...
          volatiliteit. Hier wordt V op tijdstip 1 berekend.
176     alpha_1=alpha_1+w(i)*alpha(1,i);
177     alpha_2=alpha_2+w(i)*alpha(2,i);
178     alpha_3=alpha_3+w(i)*alpha(3,i);
179     alpha_4=alpha_4+w(i)*alpha(4,i);
180     alpha_5=alpha_5+w(i)*alpha(5,i);
181 end
```

```
% sumv_squared = sumv_squared + (v1(1)-V(1))^2;
ALPHA(1,1)=alpha_1;
ALPHA(2,1)=alpha_2;
ALPHA(3,1)=alpha_3;
ALPHA(4,1)=alpha_4;
ALPHA(5,1)=alpha_5;


for i = 2:T
    v2(1,:) = v2(2,:);                  %De tijd is met dt vooruit gegaan, dus de v2 ...
        matrix verschuift alles omhoog.
    for j = 1:N

%         lambdas = lambdas2*v2(1,j);
        alpha(1,j) = alpha(1,j)+normrnd(0,sigma_theta/sqrt(i-1));
        alpha(2,j) = alpha(2,j)+normrnd(0,sigma_rho/sqrt(i-1));
        alpha(3,j) = alpha(3,j)+normrnd(0,sigma_xi/sqrt(i-1));
        alpha(4,j) = alpha(4,j)+normrnd(0,sigma_kappa/sqrt(i-1));
        alpha(5,j) = alpha(5,j)+normrnd(0,sigma_mu/sqrt(i-1));
         if alpha(1,j)<=restr_theta
            alpha(1,j)=restr_theta;
            teller1=teller1+1
        end

        if alpha(2,j)>=1-restr_rho
            alpha(2,j)=1-restr_rho;
            teller2_1=teller2_1+1
        end


        if alpha(2,j)<=-1+restr_rho
            alpha(2,j)=-1+restr_rho;
            teller2_2=teller2_2+1
        end


        if alpha(3,j)<=restr_xi
            alpha(3,j)=restr_xi;
            teller3=teller3+1
        end
        if alpha(4,j)<=restr_kappa
            alpha(4,j)=restr_kappa;
            teller4=teller4+1
        end
        %v2(2,j) = (cs)* ncx2rnd(ds,lambdas);
        v2(2,j) = normrnd(v2(1,j)+alpha(4,j)*(alpha(1,j)-v2(1,j))*dt + ...
            alpha(3,j)*alpha(2,j)*(y(i)-y(i-1)) - ...
            alpha(3,j)*alpha(2,j)*(alpha(5,j)-0.5*v2(1,j))*dt  ,  ...
            sqrt(alpha(3,j)^2*(1-alpha(2,j)^2)*v2(1,j)*dt));            %De tweede rij van ...
            v2 wordt hier gesimuleerd.

        if (v2(2,j)<=0)
            v2(2,j)=v_min;              %volatiliteit kan niet kleiner zijn dan nul.
        end
        mu_a = y(i-1)+alpha(5,j)*dt - 1/4*(v2(2,j)+v2(1,j))*dt + alpha(2,j)/alpha(3,j) * ...
            (v2(2,j)-v2(1,j) - alpha(4,j)*alpha(1,j)*dt + ...
            alpha(4,j)*1/2*(v2(2,j)+v2(1,j))*dt);
        sigma_a = sqrt(1/2*(v2(2,j)+v2(1,j))*dt*(1-alpha(2,j)^2));
                  %deze gebruiken we in alle gevallen



        mu_c = v2(1,j)+alpha(4,j)*(alpha(1,j)-v2(1,j))*dt + alpha(3,j)*alpha(2,j) * ...
            (y(i)-y(i-1)) - alpha(3,j)*alpha(2,j) * (alpha(5,j)-1/2*v2(1,j))*dt;
        sigma_c = sqrt(alpha(3,j)^2*(1-alpha(2,j)^2)*v2(1,j)*dt);
        constante_b = (alpha(3,j)^2*(1-exp(-alpha(4,j)*dt)))/(4*alpha(4,j));
        d_b = (4*alpha(1,j)*alpha(4,j))/(alpha(3,j)^2);
        lambda_b = (4*alpha(4,j)*exp(-alpha(4,j)*dt)) / ...
            (alpha(3,j)^2*(1-exp(-alpha(4,j)*dt)))*v2(1,j);

        a = normpdf(y(i),mu_a,sigma_a);

        b = (1/constante_b) * ncx2pdf(v2(2,j)/constante_b,d_b,lambda_b);
        c = normpdf(v2(2,j),mu_c,sigma_c);
```

```
248         if a==0
249             ln_a=-100000;
250         else
251             ln_a=log(a);
252         end
253
254
255         if b==0
256             ln_b=-100000;
257         else
258             ln_b=log(b);
259         end
260
261         if c==0
262             ln_c=-100000;
263         else
264             ln_c=log(c);
265         end
266
267         kip=exp(ln_a+ln_b-ln_c);
268
269         if kip>1000
270             kip=0;
271         end
272
273
274
275
276
277
278
279 %          kip = a*b/c;                    %kip staat hier voor de breuk van ...
        kansdichtheden die we gebruiken om de gewichten te berekenen van de nieuwe tijd.
280 %         if kip > 1000
281 %            kip=0;
282 %         end
283
284
285
286 %         kip = a*b/c;                    %kip staat hier voor de breuk van kansdichtheden ...
        die we gebruiken om de gewichten te berekenen van de nieuwe tijd.
287 %         if kip > 1000
288 %            kip=0;
289 %         end
290
291         w_tilde(j) = w(j) * kip;
292     end
293     for j = 1:N
294         w(j) = w_tilde(j)/sum(w_tilde); %De gewichten worden hier genormaliseerd.
295     end
296     for j = 1:N
297         w_squared(j)=w(j)^2;
298     end
299
300     V(i) = 0;
301     for j = 1:N
302         V(i) = V(i) + w(j) * v2(2,j);   %De uiteindelijke geschatte volatiliteit wordt ...
            hier berekend.
303         ALPHA(:,i) = ALPHA(:,i) + w(j)*alpha(:,j);
304     end
305
306 %     sumv_squared = sumv_squared + (v1(i)-V(i))^2;
307
308
309     N_eff = 1/sum(w_squared);           %Hier begint het resampling proces.
310     if (N_eff<N_thr)
311         display('er wordt geresampled')
312         [v_and_alpha] = resampleSystematic2([v2(2,:);alpha], w );
313         v2(2,:) = v_and_alpha(1,:);
314         v_and_alpha(1,:) = [];
315         alpha = v_and_alpha;
316         w = ones(1,N) * 1/N;
317         resampteller=resampteller+1;
318         resampnumber(i) = resampteller;
319     end
```

```
320
321
322     %i/T                                %Deze output is niet echt nodig, maar is tijdens ...
            het wachten fijn om te weten.
323 end
324 % RMSE = sqrt(sumv_squared)*(1/T);
325
326
327 %k/M
328
329 %resampnumber
330 %resampteller
331 toc
332
333 %
334 figure(1)
335 subplot(2,1,1); plot(0:dt:t,exp(y))
336 title('stock price')
337
338
339 subplot(2,1,2);
340 plot(0:dt:t,V)
341 title('estimated volatility')
342
343 % thetaline = theta*ones(T,1);
344 % rholine = rho*ones(T,1);
345 % xiline = xi*ones(T,1);
346 % kappaline = kappa*ones(T,1);
347 % muline = mu*ones(T,1);
348
349
350 figure(2)
351 plot(0:dt:t,ALPHA(1,:),'r')
352 title('estimated theta in red')
353 figure(3)
354 plot(0:dt:t,ALPHA(2,:),'r')
355 title('estimated rho in red')
356 figure(4)
357 plot(0:dt:t,ALPHA(3,:),'r')
358 title('estimated xi in red')
359 figure(5)
360 plot(0:dt:t,ALPHA(4,:),'r')
361 title('estimated kappa in red')
362 figure(6)
363 plot(0:dt:t,ALPHA(5,:),'r')
364 title('estimated mu in red')
```

B Resultaten

B.1 De non-centraal verdeelde optimal importance function

De non-centraal verdeelde optimal importance function:								
seed		1	2	3	4	5	6	7
RMSE	30 particles	0.041347662	0.055384874	0.041428959	0.056240002	0.039452805	0.0460436	0.048042057
	100 particles	0.038191563	0.060966497	0.040399361	0.055310924	0.044146176	0.040307345	0.050938222
	500 particles	0.039112514	0.052484306	0.038966025	0.052324174	0.040115542	0.039953643	0.050590164
tijd in sec	30 particles	19.73458187	20.29521553	26.05413652	19.17270447	19.25697304	18.80636697	19.38333771
	100 particles	60.86050103	80.69605031	62.11465315	62.41002061	62.4357242	61.03651709	62.92833973
	500 particles	303.3166167	401.3660492	304.602212	309.5746549	306.2458828	300.9766386	311.8011046
number of resamples	30 particles	38	36	35	36	37	39	36
	100 particles	43	42	44	37	42	40	43
	500 particles	44	39	47	45	48	44	42
seed		8	9	10	11	12	13	14
RMSE	30 particles	0.058113995	0.046008576	0.038931835	0.042834996	0.03537132	0.052726797	0.039121989
	100 particles	0.053288374	0.03874969	0.03706688	0.048225431	0.036885471	0.042346598	0.03863627
	500 particles	0.055031979	0.038464952	0.038078425	0.049472118	0.033925094	0.03977349	0.038391433
tijd in sec	30 particles	19.18234115	19.216241	18.9164344	19.29983822	18.9102251	18.91756276	19.09255024
	100 particles	62.33829916	62.18914338	61.79007465	62.53238796	61.14583063	61.24757574	61.66670745
	500 particles	309.7542806	304.4635741	306.8524534	308.2539304	301.700464	303.171014	309.252763
number of resamples	30 particles	39	41	36	32	33	34	38
	100 particles	42	42	40	39	37	42	38
	500 particles	45	47	40	40	46	45	42
seed		15	16	17	18	19	20	**gemiddeldes**
RMSE	30 particles	0.043148289	0.037382135	0.04481929	0.066789677	0.041238375	0.054869936	0.046464858
	100 particles	0.048129177	0.036259438	0.046935792	0.057502126	0.045744679	0.052777556	0.045640378
	500 particles	0.046022326	0.036947267	0.045011678	0.057545308	0.045352019	0.050077852	0.044382015
tijd in sec	30 particles	19.02309619	18.63250088	18.84743531	19.07554345	18.89115867	19.20178822	19.49550158
	100 particles	61.41652802	60.94553622	61.1584763	62.28003967	61.02408413	61.80444612	62.70104678
	500 particles	305.7352709	303.043592	302.2291203	308.6439537	305.8170943	309.9809775	310.8390823
number of resamples	30 particles	36	39	35	38	39	33	36.5
	100 particles	39	43	38	40	38	44	40.65
	500 particles	42	46	46	43	44	42	43.85

B.2 De normaal verdeelde optimal importance function

De normaal verdeelde optimal importance function:								
seed		1	2	3	4	5	6	7
RMSE	30 particles	0.039974878	0.049896671	0.040132653	0.051642363	0.043035154	0.041723721	0.052887288
	100 particles	0.0359705	0.052225681	0.040497182	0.050957729	0.036521665	0.039919957	0.046904364
	500 particles	0.038104744	0.054140782	0.039946286	0.057200507	0.045295193	0.047406368	0.049997885
tijd in sec	30 particles	8.718327331	7.642084209	8.350267991	8.323247545	8.636375358	8.545632008	8.887776291
	100 particles	23.60238381	23.15326396	24.94648593	26.08573748	25.00328291	26.18371267	26.07017539
	500 particles	114.3278666	114.7754265	120.7832254	123.3657176	126.1236028	121.1620549	133.492198
number of resamples	30 particles	29	29	30	27	34	30	29
	100 particles	36	30	34	32	37	34	29
	500 particles	37	30	41	43	53	44	32
seed		8	9	10	11	12	13	14
RMSE	30 particles	0.063969051	0.036412495	0.031683842	0.047288724	0.039048836	0.042523437	0.04265242
	100 particles	0.054828852	0.034025993	0.031981906	0.049214153	0.037099836	0.038365771	0.038450445
	500 particles	0.05495933	0.037318959	0.032645653	0.050201305	0.037987297	0.039159797	0.041841617
tijd in sec	30 particles	8.753448901	8.336321279	8.380492095	8.351663278	8.360752765	8.52973485	8.517415836
	100 particles	26.91266521	25.5673805	25.66409796	25.37734182	25.71160425	25.77932577	25.88567761
	500 particles	124.6283748	118.9032189	121.8719206	120.8999356	122.3228451	122.3668377	122.4137593
number of resamples	30 particles	27	34	30	25	31	36	23
	100 particles	36	39	32	25	38	36	28
	500 particles	39	46	42	29	45	38	31
seed		15	16	17	18	19	20	**gemiddeldes**
RMSE	30 particles	0.042912146	0.037043557	0.042946186	0.061551717	0.041704601	0.057821471	0.045342561
	100 particles	0.047261329	0.033263728	0.046615144	0.054574365	0.04560474	0.057106272	0.043569481
	500 particles	0.051465712	0.037165224	0.044785657	0.059200389	0.049068675	0.068765208	0.046832829
tijd in sec	30 particles	8.542022247	8.603490846	8.215867511	8.245184555	8.226388009	8.167936236	8.416721457
	100 particles	25.53918856	25.81366182	25.15631651	25.14804415	25.10128766	24.92364642	25.38126402
	500 particles	121.3214334	121.7629289	121.2097185	121.3477443	121.2104502	120.4794429	121.7384351
number of resamples	30 particles	33	30	27	27	31	26	29.4
	100 particles	32	38	33	30	32	34	33.25
	500 particles	38	45	45	42	34	38	39.6

B.3 De suboptimal importance function

De suboptimal importance function:								
seed		1	2	3	4	5	6	7
RMSE	30 particles	0.073306685	0.071888832	0.057188191	0.088955582	0.05884973	0.056712091	0.118063536
	100 particles	0.073355063	0.071869051	0.057594784	0.088541787	0.058606952	0.056506346	0.118072955
	500 particles	0.073201407	0.072133759	0.057185393	0.088500701	0.058149956	0.05681356	0.117762358
tijd in sec	30 particles	5.559480582	6.02859928	5.357927288	5.41990054	6.177855175	6.379770225	4.302440603
	100 particles	16.37812405	17.99181018	17.08033413	16.60785122	18.23476137	19.15075219	13.8981465
	500 particles	534.4367507	83.53243837	77.54538071	77.29996292	85.44626194	88.01307168	62.29125942
number of resamples	30 particles	41	32	42	60	30	25	125
	100 particles	51	36	47	76	36	29	134
	500 particles	51	41	51	78	39	30	136
seed		8	9	10	11	12	13	14
RMSE	30 particles	0.071497105	0.071326191	0.06077349	0.080677206	0.077932388	0.067098376	0.076917261
	100 particles	0.071713652	0.070737267	0.060965788	0.080927189	0.077712927	0.066626954	0.07736612
	500 particles	0.071548286	0.070779043	0.060752095	0.081047485	0.077647521	0.06673571	0.077151173
tijd in sec	30 particles	6.727885335	7.108579456	6.83891414	5.797259193	7.126409813	5.765774485	7.120598717
	100 particles	19.68941557	21.13133643	20.44843207	17.51845717	21.04091296	16.879081	21.58292124
	500 particles	92.63357329	99.40810548	96.88486746	81.61763917	99.75040128	76.77887819	101.2411966
number of resamples	30 particles	25	13	14	46	12	61	14
	100 particles	27	15	17	54	14	74	18
	500 particles	31	17	17	56	16	79	19
seed		15	16	17	18	19	20	**gemiddeldes**
RMSE	30 particles	0.073457371	0.042860369	0.102045201	0.253871925	0.170042656	0.095141648	0.088430292
	100 particles	0.073307239	0.042806652	0.101912203	0.253534732	0.17004944	0.094528548	0.088336783
	500 particles	0.073387903	0.0429459	0.101689183	0.253504793	0.170265935	0.094533401	0.088286778
tijd in sec	30 particles	7.005894862	5.762110111	3.419330188	8.013613699	7.573701999	5.729331539	6.160768862
	100 particles	21.02046366	17.18156882	10.86122559	23.87261948	22.81574161	17.35925893	18.53716071
	500 particles	98.8368171	79.94293366	48.49519225	113.7856005	107.6686049	79.22299095	109.2415963
number of resamples	30 particles	15	33	128	8	10	60	39.7
	100 particles	17	37	135	10	9	67	45.15
	500 particles	17	40	137	12	11	67	47.25

B.4 Verschillende Resamplemethoden

verschillende resamplemethoden:								
seed		1	2	3	4	5	6	7
RMSE	Multinomial	0.053806135	0.039011843	0.054696159	0.045289925	0.039725305	0.052781811	0.058857754
	Residual	0.060618134	0.039016716	0.051702481	0.040883061	0.042298186	0.05100673	0.056084259
	Stratified	0.054305807	0.03956481	0.056686512	0.040742101	0.049642716	0.04782869	0.057026716
	Systematic	0.060720396	0.039408913	0.05393197	0.046425859	0.042942027	0.050299405	0.055054966
tijd in sec	Multinomial	118.9415087	123.3234727	118.8827973	118.361044	118.316482	118.3681377	121.1853833
	Residual	120.1016966	122.1565827	120.4189448	120.4763459	120.7569208	119.9361765	122.5791483
	Stratified	120.9165663	118.6359758	119.1503073	119.0091637	120.5936697	118.4083164	121.422527
	Systematic	118.1637295	122.1731087	118.7959941	119.148432	118.91213	119.0019434	121.2425659
number of resamples	Multinomial	36	40	38	51	38	41	40
	Residual	38	43	34	53	50	46	39
	Stratified	32	42	38	40	49	41	39
	Systematic	38	38	42	47	49	33	33
seed		8	9	10	11	12	13	14
RMSE	Multinomial	0.032622657	0.033231875	0.04926637	0.035596073	0.039362235	0.044453162	0.032365703
	Residual	0.035566973	0.032935581	0.050267445	0.03679518	0.03816393	0.03817861	0.039072129
	Stratified	0.039551074	0.03138434	0.052511682	0.039805771	0.041729845	0.039168012	0.047565508
	Systematic	0.033798611	0.033994452	0.054644636	0.038677991	0.042286334	0.039539322	0.049302314
tijd in sec	Multinomial	120.9786074	120.7946288	121.2037488	121.6940224	120.6778006	121.4437238	120.4917509
	Residual	122.2681661	122.646809	122.4855761	123.5878603	123.5264192	122.6712739	122.0723167
	Stratified	120.6053131	120.7876968	121.1610518	121.7140271	121.3957138	116.7261518	120.5030889
	Systematic	120.7152471	120.8356238	121.7479177	121.7812325	121.5599443	120.7094792	120.3824124
number of resamples	Multinomial	49	45	30	40	41	36	37
	Residual	42	47	33	41	42	32	39
	Stratified	44	41	31	43	42	31	40
	Systematic	45	51	36	44	44	28	38
seed		15	16	17	18	19	20	**gemiddeldes**
RMSE	Multinomial	0.036861797	0.044885856	0.06009337	0.045805437	0.061680341	0.044976177	0.045268499
	Residual	0.037151975	0.044985729	0.056827198	0.0486936	0.055600852	0.044780892	0.045031483
	Stratified	0.036239933	0.051398551	0.057027026	0.049445892	0.042263244	0.045702495	0.045979536
	Systematic	0.037796676	0.04684238	0.056750978	0.048366366	0.053900532	0.046156089	0.046542011
tijd in sec	Multinomial	120.8878142	121.1735469	120.6939008	121.187727	120.6153826	120.5005157	120.4860998
	Residual	122.6453312	123.0503352	121.9360791	123.0856658	122.0320822	121.998571	122.0216151
	Stratified	120.9335707	121.1932933	120.8727336	121.3009607	120.6856918	120.4421239	120.3228972
	Systematic	120.8424306	121.1905525	120.6534885	121.2483839	120.93226	120.4870942	120.5261985
number of resamples	Multinomial	44	40	42	44	41	40.65	40.6825
	Residual	38	42	34	44	34	40.7	40.585
	Stratified	43	49	37	46	35	40.2	40.16
	Systematic	45	50	37	42	40	40.95	41.0475

B.5 Verschillende waarden van N_{thr}

	Verschillende waarden voor N_{thr}							
seed	Nthr	1	2	3	4	5	6	7
RMSE	N/6	0.039424042	0.055208282	0.040411479	0.059427609	0.041767469	0.046298572	0.051634735
	N/3	0.045416258	0.061092839	0.039811023	0.052933927	0.043305917	0.040178942	0.050928403
	N/2	0.039604	0.05363881	0.039706076	0.053189232	0.041650239	0.038943726	0.045879678
	2N/3	0.038325556	0.05449893	0.04111042	0.051851401	0.040504575	0.039145521	0.050513124
tijd in sec	N/6	123.5170387	122.6153473	121.9379503	121.2883437	121.2515966	121.5286205	121.3508628
	N/3	120.8733643	122.2804443	121.5538134	121.3955709	121.2827417	121.4973912	121.3790186
	N/2	120.0136148	122.2269404	121.2005682	121.310657	121.2294788	121.3381665	121.8902787
	2N/3	119.1118368	121.8329942	121.2757896	124.8516686	121.0839084	121.2488011	121.2629267
number of resamples	N/6	24	23	30	25	36	30	25
	N/3	43	37	46	35	51	40	34
	N/2	57	47	60	50	64	58	46
	2N/3	83	73	93	82	90	87	67
seed	Nthr	8	9	10	11	12	13	14
RMSE	N/6	0.054861503	0.03236915	0.031243612	0.04925778	0.040587951	0.039452823	0.044467042
	N/3	0.055000891	0.037788167	0.033852817	0.057983345	0.040984509	0.04027865	0.040053577
	N/2	0.055849935	0.032604328	0.031169764	0.050111286	0.039841264	0.039964868	0.042967848
	2N/3	0.055572112	0.032367949	0.032080463	0.047627311	0.036881267	0.040705604	0.03882735
tijd in sec	N/6	121.1732434	121.1001695	121.2015073	120.8792077	121.2427285	120.8485179	121.1322538
	N/3	121.1390712	121.0553981	121.0179065	121.1126284	121.2961454	120.6279499	121.1757921
	N/2	121.2247616	121.1233475	121.1624073	121.0442516	121.2373055	120.7654497	121.3580071
	2N/3	121.3046074	121.1218016	121.3442658	121.0455044	121.1252052	120.8453886	121.2222651
number of resamples	N/6	23	29	27	17	27	27	20
	N/3	33	48	45	31	45	42	28
	N/2	51	58	58	50	59	55	48
	2N/3	80	93	89	66	85	89	73
seed	Nthr	15	16	17	18	19	20	**gemiddeldes**
RMSE	N/6	0.038245901	0.036666245	0.049899291	0.061930917	0.047683964	0.057659294	0.045924883
	N/3	0.048091031	0.036029257	0.044611517	0.057311021	0.048043924	0.054354965	0.046402549
	N/2	0.034306702	0.037209728	0.044972005	0.059225004	0.050907523	0.049502352	0.044062218
	2N/3	0.034154118	0.03696873	0.051011335	0.06013104	0.046635445	0.05366153	0.044128689
tijd in sec	N/6	121.1675736	121.3798616	120.2181052	120.9812776	120.2447368	121.0138985	121.3036421
	N/3	120.9031084	121.4834179	120.2581841	121.0736738	120.2848256	120.9060537	121.129825
	N/2	121.0502913	121.603458	120.1452789	120.8844312	120.4009052	120.7332419	121.0971421
	2N/3	121.0547703	121.3588283	120.0544418	120.9370092	120.4949665	120.9066569	121.1741818
number of resamples	N/6	33	29	29	30	29	28	27.05
	N/3	40	50	43	35	40	32	39.9
	N/2	59	66	60	48	60	45	54.95
	2N/3	84	91	102	82	89	72	83.5

B.6 Meerdere gegenereerde markten

gegenereerde markten							
seed	1	2	3	4	5	6	7
RMSE	0.056755776	0.064302625	0.058273735	0.06371552	0.061056247	0.056469374	0.078795155
tijd	209.7802461	230.1044067	274.4217585	241.2787397	253.4793009	214.2576343	278.1205062
number of resamples	68	43	70	101	55	67	42
θ begin	0.957396111	1.051077115	1.011264341	0.993091299	1.013286689	1.017518788	1.016109036
θ eind	0.079663918	0.126067776	0.049869311	0.070172033	0.096849415	0.109017715	0.18722152
ρ begin	-0.253184674	-0.259528074	-0.260441388	-0.249189127	-0.247924262	-0.255423357	-0.249177867
ρ eind	-0.371112055	-0.204705287	-0.319791982	-0.388855753	-0.020793217	-0.219509152	-0.170892346
ξ begin	0.447285698	0.46970432	0.458034601	0.469561883	0.470426054	0.472099137	0.458879688
ξ eind	0.734667128	0.604644717	0.734636977	1.013928183	0.761798356	0.814214026	0.837014214
κ begin	5.211759331	4.926007667	4.973410778	5.174841889	4.953962678	4.986334932	4.901423768
κ eind	4.108768758	2.550401865	7.619452542	2.201208403	6.458060644	6.985067529	5.990445784
μ begin	0.281520224	0.26781481	0.264886439	0.284599746	0.272953601	0.266749565	0.283122096
μ eind	0.287590115	0.437418492	0.358403251	0.117319723	0.099463844	0.278901784	0.212213155
aantal keer θ bijgesteld	112	129	324	35	1	5	235
aantal keer ρ naar boven bijgesteld	0	0	0	0	0	0	0
aantal keer ρ naar beneden bijgesteld	0	0	0	0	0	0	0
aantal keer ξ bijgesteld	55	73	125	77	120	74	165
aantal keer κ bijgesteld	2	0	0	0	9	16	309
seed	8	9	10	11	12	13	14
RMSE	0.07075746	0.05590261	0.049838718	0.061436397	0.054135593	0.053294123	0.056433796
tijd	255.0159028	235.9887129	1569.432679	357.8948069	267.3812452	297.3265176	239.513025
number of resamples	49	60	27	28	56	44	31
θ begin	1.016454629	1.021568237	1.07078001 9	1.007265216	1.010353658	1.008443428	1.001059235
θ eind	0.151319575	0.102759338	0.084693147	0.101535215	0.052896278	0.070949119	0.17182036
ρ begin	-0.248244836	-0.254206642	-0.237391816	-0.247373395	-0.253730953	-0.244830137	-0.239553099
ρ eind	-0.345532168	-0.205181965	-0.10105965	-0.22710392	-0.19990486	-0.052758908	-0.101507685
ξ begin	0.463586037	0.451965528	0.452139132	0.443195357	0.467843639	0.462103695	0.476827929
ξ eind	0.69711531	0.854454191	0.34692783	0.507985895	0.700461861	0.591229748	0.49414086
κ begin	4.925214405	4.783789424	5.092464431	5.013272174	5.011398759	4.942464535	4.948556662
κ eind	2.023324755	6.12086376	5.341747161	4.029090051	6.749427768	5.103560562	1.905697298
μ begin	0.269799644	0.277448539	0.272543688	0.267623152	0.271205763	0.278435196	0.271842612
μ eind	0.177914717	0.419594038	0.284274885	0.202306341	0.128374847	0.089792357	0.14146641
aantal keer θ bijgesteld	34	44	128	296	24	202	66
aantal keer ρ naar boven bijgesteld	0	0	0	0	0	0	0
aantal keer ρ naar beneden bijgesteld	0	0	0	0	0	0	0
aantal keer ξ bijgesteld	126	97	2655	301	166	214	99
aantal keer κ bijgesteld	136	3	86	20	0	10	5

	vervolg: gegenereerde markten						
seed	15	16	17	18	19	20	**gemiddeldes**
RMSE	0.05204877	0.050484462	0.057415624	0.064327716	0.070274445	0.055839083	0.059577876
tijd	293.0743696	258.8030128	204.0851191	260.6093463	504.3129516	271.356464	335.8118373
number of resamples	48	70	83	45	37	32	52.8
θ begin	1.017159686	0.968432348	1.00512028	1.013036507	1.002046658	1.036875406	1.011916934
θ eind	0.092582854	0.049601671	0.064903429	0.081231097	0.072225194	0.132411893	0.097389543
ρ begin	-0.253426956	-0.247387916	-0.25326122	-0.258984971	-0.253609602	-0.249759896	-0.250831509
ρ eind	-0.147983889	-0.1419669	-0.2432866	-0.367034333	-0.321598571	-0.078877306	-0.211472827
ξ begin	0.473392823	0.469244646	0.445895334	0.466178895	0.467874837	0.452746299	0.461949277
ξ eind	0.654766594	0.537019628	0.9492511	0.774005059	0.404770581	0.568984367	0.679100831
κ begin	5.041759333	5.149465698	5.07072116	4.968432069	5.057454408	5.038037584	5.008538584
κ eind	7.783233477	3.203465306	7.464425689	2.690507859	7.485413117	3.836413085	4.982528771
μ begin	0.278333211	0.264520261	0.273607367	0.269072143	0.267507169	0.264069139	0.272382718
μ eind	0.333330753	0.194213737	0.136701403	0.138003718	0.17824088	0.179439407	0.219748193
aantal keer θ bijgesteld	217	505	119	274	398	270	170.9
aantal keer ρ naar boven bijgesteld	0	0	0	0	0	0	0
aantal keer ρ naar beneden bijgesteld	0	0	0	0	0	0	0
aantal keer ξ bijgesteld	142	117	55	251	414	133	272.95
aantal keer κ bijgesteld	3	30	0	16	1	121	38.35

Bibliografie

[1] Shin Ichi Aihara, Arunabha Bagchi, and Saikat Saha. On parameter estimation of stochatic volatility models from stock data using particle filter -application to aex index-. *International Journal of Innovative Computing, Information and Control*, 5(1):17–27, January 2009.

[2] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *The Journal of Political Economy*, 81(3):637–654, 1973.

[3] Broadie and Kaya. Exact simulation of stochastic volatility and other affine jump diffusion processes. *Operations Research*, 54(2):217–231, 2006.

[4] Lin Chen. Stochastic mean and stochastic volatility - a three-factor model of the term structure of interest rates and its application to the pricing of interest rate derivatives. *Financial Markets, Institutions, and Instruments*, 5:1–88, 1996.

[5] Branko Ristic et al. *Beyond the Kalman Filter*. DTSO, Unknown, 2004.

[6] Petar M. Djurić et al. Particle filtering. *IEEE Signal Processing Magazine*, 20(5):19–38, 2003.

[7] Steven L. Heston. A closed-form solution for options with stochastic volatility with apllications to bond and currency options. *The review of financial studies*, 6(2):327–343, 1993.

[8] Jayesh H. Kotecha and Petar M. Djurić. Gaussian particle filtering. *IEEE Transactions on singal processing*, 51(10):2592–2601, October 2003.