

Efficiënt schatten van viscositeit en diffusiviteit met beperkte rekencapaciteit

door

Roemer Rouwhorst

aan de Technische Universiteit Delft,
in het openbaar te verdedigen op vrijdag 25 oktober om 10:30 uur.

Studentnummer: 5594863
Beoordelingscommissie: Dr. J. Söhl, TU Delft, supervisor
Drs. I. A. M. Goddijn, TU Delft

Een elektronische versie van deze scriptie is beschikbaar: <http://repository.tudelft.nl/>.



Inhoudsopgave

Introductie	3
1 Regressie	4
1.1 Regressie met constante variantie	4
1.2 Gewogen regressie	9
1.3 Probleemstelling	10
1.3.1 Versimpelde weergave	11
1.4 Kostenbeperking	14
1.4.1 Twee meetpunten	14
1.4.2 Meer dan twee meetpunten	16
1.5 Intercept	19
2 Simultaan schatten	21
Conclusie/Discussie	24

Samenvatting voor de geïnteresseerde

In deze scriptie wordt behandeld hoe we aan de hand van simulaties op een computer en met zo min mogelijk rekenwerk kunnen meten hoe stroperig (viscositeit) en hoe beweeglijk (zelfdiffusiviteit) verschillende stoffen zijn. De simulaties worden uitgevoerd op een computer in een kubusvormige simulatieruimte. Het doel van het onderzoek was om de optimale grootte van de kubussen te vinden, zodat we de viscositeit en zelfdiffusiviteit met zo min mogelijk rekenwerk kunnen vinden.

Uit het onderzoek bleek dat de optimale ribbelengtes van de kubus afhangen van de relatie tussen de nauwkeurigheid van de metingen en de bijbehorende kosten. Dankzij een methode genaamd "gewogen regressie" werden aan kubussen met grotere ribbelengtes zwaardere gewichten toegekend omdat deze preciezere resultaten opleverden, terwijl kleinere kubussen, ondanks hun lagere meetnauwkeurigheid, een waardevolle bijdrage leverden door minder rekenkracht te vereisen. Deze balans resulteerde in een vermindering van de schattingsfout in zowel de viscositeit als de zelfdiffusiviteit.

Daarnaast werd vastgesteld dat de simultane schatting van zowel viscositeit als zelfdiffusiviteit, waarbij gebruik werd gemaakt van de covariantie (samenhang) tussen beide parameters, leidde tot een 95%-betrouwbaarheidsinterval dat zo'n 85% van de onzekerheid wegnam ten opzichte van schattingen die deze covariantie niet meenamen.

Samenvatting

In deze scriptie wordt een methode ontwikkeld om de viscositeit en zelfdiffusiviteit van moleculen te schatten door middel van moleculaire simulaties. De simulaties worden uitgevoerd in kubusvormige volumes met verschillende ribbelengtes L , waarbij de computationele kosten en de nauwkeurigheid van de metingen afhankelijk zijn van de kubusgrootte. Metingen in kleinere kubussen vereisen minder rekenkracht maar leveren onnauwkeurige resultaten op, terwijl metingen in grotere kubussen preciezere schattingen geven tegen hogere computationele kosten.

De relatie tussen de werkelijke zelfdiffusiviteit D_{self}^{∞} en de gemeten zelfdiffusiviteit $D_{\text{self}}^{\text{MS}}$ wordt beschreven met de formule:

$$D_{\text{self}}^{\infty} = D_{\text{self}}^{\text{MS}} + \frac{\xi k_B T}{6\pi\eta L}$$

waarbij L de ribbelengte is en η de viscositeit. Dit leidt tot een lineair regressiemodel waarbij $D_{\text{self}}^{\text{MS}}$ de afhankelijke variabele is en $\frac{1}{L}$ de onafhankelijke variabele is. De schatting van de intercept en de helling moet nauwkeurig worden bepaald.

Er is gebruik gemaakt van gewogen regressie om heteroscedasticiteit in de fouten van de simulaties aan te pakken. Metingen in grotere kubussen tellen zwaarder mee voor de regressie omdat ze nauwkeuriger zijn, terwijl metingen in kleinere kubussen minder meetellen. De variantiefunctie $\sigma(x)$ en de kostenfunctie $f(x)$ spelen een cruciale rol bij het bepalen van de optimale ribbelengtes en gewichten. In het bijzonder is aangetoond dat voor hogere ordes van de variantiefunctie (bijv. $\sigma(x) = x^3$) de voorkeur wordt gegeven aan grotere ribbelengtes.

Daarnaast zijn de simultane schattingen van zowel de viscositeit als de zelfdiffusiviteit verbeterd door gebruik te maken van de covariantiematrix van de regressie. De afhankelijkheid tussen de intercept en helling wordt gebruikt voor een verkleining van het betrouwbaarheidsgebied met 85% ten opzichte van een traditionele benadering zonder covariantie. Dit is geïllustreerd met behulp van betrouwbaarheidsgebieden, waaruit bleek dat het meenemen van de covariantie bijdraagt aan het reduceren van de onzekerheid in de schattingen.

Introductie

In moleculaire dynamica is men geïnteresseerd in de **viscositeit**, een maat van stroperigheid van een stof, van verschillende moleculen. Deze viscositeit kan worden berekend aan de hand van computersimulaties. Bij deze simulaties wordt van een vast aantal moleculen in een kubusvormig, afgesloten systeem de beweging geobserveerd. Na een tijd wordt de simulatie stopgezet, en wordt van ieder molecuul de gemiddelde verplaatsing vastgelegd. Deze gemiddelde verplaatsing is een stofeigenschap en wordt de **zelfdiffusiviteit** genoemd. Via de zelfdiffusiviteit kan de viscositeit worden berekend.

Om de zelfdiffusiviteit te schatten maakt men gebruik van kubusvormige systemen van verschillende groottes. Een meting in een kleinere 'kubus' vereist minder rekenkracht, maar gaat ten koste van de nauwkeurigheid van de meting. Een meting in een grotere kubus is nauwkeuriger dan in een kleinere kubus, maar vereist meer rekenkracht. Het doel is een balans te vinden tussen de computationele kosten, en een nauwkeurige schatting van de zelfdiffusiviteit en viscositeit.

1 Regressie

Het schatten van de viscositeit van stoffen door middel van molecuulsimulaties gaat met een omweg. Eerst wordt de zelfdiffusiviteit bepaald. De zelfdiffusiviteit, ofwel de mate van snelheid waarin een stof zich verspreidt, kan benaderd worden aan de hand van computersimulaties. Yeh en Hummer [6] hebben een verband gevonden tussen de werkelijke zelfdiffusiviteit en de zelfdiffusiviteit die gevonden wordt door de molecuulsimulaties. De relatie bevat een foutterm D^{YH} die door **Y**eh en **H**ummer is gevonden: [2]

$$D_{\text{self}}^{\infty} = D_{\text{self}}^{\text{MS}} + D^{\text{YH}} = D_{\text{self}}^{\text{MS}} + \frac{\xi k_b T}{6\pi\eta L}$$

Waarbij L de lengte van de ribben van de kubus is, k_b de Boltzmannconstante, η de schuifviscositeit, T de temperatuur, en ξ een dimensieloze constante. De zelfdiffusiviteit die wordt bepaald aan de hand van de molecuulsimulaties wordt weergegeven als $D_{\text{self}}^{\text{MS}}$, waarbij 'MS' staat voor 'molecular simulations'. De werkelijke zelfdiffusiviteit van een stof wordt weergegeven als D_{self}^{∞} . Deze heeft een oneindigheidsteken (∞) omdat de werkelijke zelfdiffusiviteit equivalent is aan de zelfdiffusiviteit in een oneindig grote kubus.

Als we dit vervolgens herschrijven:

$$\underbrace{D_{\text{self}}^{\text{MS}}}_Y = \underbrace{D_{\text{self}}^{\infty}}_A + \underbrace{\frac{1}{\eta}}_B \cdot \underbrace{-\frac{\xi k_b T}{6\pi L}}_X \quad (1)$$

herkennen we een vorm van lineaire regressie. De simulaties worden uitgevoerd in kubussen met verschillende groottes, met verschillende lengten van de ribben. Deze lengte kan theoretisch gezien oneindig groot worden ($X \rightarrow 0$ als $L \rightarrow \infty$). De lengte kan echter niet oneindig klein worden, aangezien de moleculen in het doosje moeten passen. In het vervolg beschouwen we het deel van de X -term $-\frac{\xi k_b T}{6\pi}$ als constant, omdat alleen de ribbelengte L van de kubus zal variëren. Dit heeft als gevolg dat $D_{\text{self}}^{\text{MS}}$ recht evenredig is met $\frac{1}{L}$.

Voor het gemak nemen we aan dat de minimale ribbelengte van L correspondeert met een waarde van $X = 1$. Mocht dit nodig zijn, kan X geschaald worden, zodat zij een andere waarde aanneemt bij de minimale lengte van L . X neemt dus waarden aan op $(0, 1]$.

1.1 Regressie met constante variantie

We passen lineaire regressie (met één onafhankelijke variabele) toe van de vorm

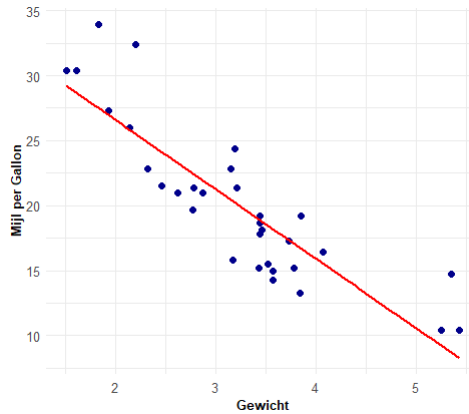
$$Y = A + BX$$

waarbij Y de afhankelijke variabele is, X de onafhankelijke variabele, A de intercept en B de helling. A geeft de voorspelde waarde van Y weer wanneer de onafhankelijke variabele 0 is. B geeft aan hoeveel Y verandert bij een toename van 1 eenheid in X .

In figuur 1 is een voorbeeld geïllustreerd van data van hoeveel mijl bepaalde auto's rijden op hun benzine uitgezet tegen hun gewicht. Met lineaire regressie wordt geprobeerd data zo goed mogelijk te voorspellen door middel van een linear verband tussen de afhankelijke variabele en de onafhankelijke variabele.

Het is duidelijk dat er geen rechte lijn door al deze punten bestaat. Hierom bevat het model een residuterm E : Deze geeft het verschil weer tussen de voorspelde waarde door de regressievergelijking en de werkelijke waarde. Voor de i de observatie ontstaat zo de volgende vergelijking:

$$Y_i = A + BX_i + E_i$$



Figuur 1: Data van mijl per gallon uitgezet tegen gewicht van verschillende auto's. De kleinste-kwadraten regressielijn $Y = 37.285 - 5.344X$ is weergegeven in het rood.

Het streven is dat de regressielijn de werkelijke waarden zo goed mogelijk voorspelt. Hiervoor moet eerst vastgesteld worden wat 'zo goed mogelijk' inhoudt. 'Zo goed mogelijk' is gedefinieerd als: De som over alle fouten moet minimaal zijn. De volgende twee zijn gebruikelijke manieren om dit te doen:

$$\text{LSS regressie: } \sum_{i=1}^n E_i^2 \quad \text{of} \quad \text{LAV regressie: } \sum_{i=1}^n |E_i|$$

Bij LSS regressie wordt de som van de kwadraten van de fouten geminimaliseerd, bij LAV regressie wordt de absolute waarde van de fouten geminimaliseerd. Beide opties hebben hun voor- en nadelen. Bij LSS regressie wordt er aangenomen dat alle fouten een verdeling hebben met constante variantie in tegenstelling tot LAV regressie. Ook is LSS wat minder goed bestand tegen uitschieters in de data dan LAV. Daarintegen heeft het wel fijne wiskundige eigenschappen die bijvoorbeeld helpen bij het opstellen van betrouwbaarheidsintervallen. LSS wordt daarom vaak als criterium gebruikt.

Het doel is om LSS te minimaliseren. Laat

$$S(A, B) = \sum_{i=1}^n E_i^2 = \sum_{i=1}^n (Y_i - A - BX_i)^2$$

Om de LSS te minimaliseren stellen we de partiële afgeleiden gelijk aan nul. Eerst nemen we de afgeleide naar A :

$$\frac{\partial S}{\partial A} = -2 \sum_{i=1}^n (Y_i - A - BX_i)$$

Omdat we zoeken naar een minimum volgt hieruit:

$$\sum_{i=1}^n (Y_i - A - BX_i) = 0$$

$$nA = \sum_{i=1}^n (Y_i - BX_i) = n\bar{Y} - nB\bar{X}$$

$$\hat{A} = \bar{Y} - B\bar{X} \quad (2)$$

Waarbij $\bar{Y} = \frac{1}{n} \sum_{i=1}^n Y_i$, $\bar{X} = \frac{1}{n} \sum_{i=1}^n X_i$. De uitdrukking voor A die S minimaliseert heet de 'kleinste kwadraten schatter'. We zitten in een minimum, want

$$\frac{\partial^2 S}{\partial A^2} = -2 \cdot \sum_{i=1}^n -1 = 2N > 0$$

Dan de afgeleide naar B :

$$\frac{\partial S}{\partial B} = -2 \sum_{i=1}^n X_i (Y_i - A + BX_i)$$

$$\sum_{i=1}^n X_i (Y_i - A + BX_i) = 0$$

$$\sum_{i=1}^n (X_i Y_i - AX_i - BX_i^2) = 0$$

Hierin vullen we (2) in:

$$\begin{aligned} \sum_{i=1}^n (X_i Y_i - (\bar{Y} - B\bar{X})X_i - BX_i^2) &= 0 \\ \sum_{i=1}^n X_i Y_i - \bar{Y} \sum_{i=1}^n X_i + B\bar{X} \sum_{i=1}^n X_i - \sum_{i=1}^n BX_i^2 &= 0 \\ B(\sum_{i=1}^n X_i^2 - \bar{X} \sum_{i=1}^n X_i) &= \sum_{i=1}^n X_i Y_i - \bar{Y} \sum_{i=1}^n X_i \end{aligned}$$

Vermenigvuldigen met n geeft dan:

$$\begin{aligned} B(n \sum_{i=1}^n X_i^2 - (\sum_{i=1}^n X_i)^2) &= n \sum_{i=1}^n X_i Y_i - \sum_{i=1}^n X_i \sum_{i=1}^n Y_i \\ \hat{B} &= \frac{n \sum_{i=1}^n X_i Y_i - \sum_{i=1}^n X_i \sum_{i=1}^n Y_i}{n \sum_{i=1}^n X_i^2 - (\sum_{i=1}^n X_i)^2} \stackrel{(*)}{=} \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sum_{i=1}^n (X_i - \bar{X})^2} \end{aligned} \quad (3)$$

En ook hier zitten we in een minimum, want

$$\frac{\partial^2 S}{\partial B^2} = -2 \cdot \sum_{i=1}^n -X_i^2 = 2 \sum_{i=1}^n X_i^2 > 0$$

(*) staat weergegeven in appendix A1. $\hat{A}$ en $\hat{B}$ zijn de schatters voor parameters A en B die S , de som van de gekwadrateerde residu's minimaliseren. In het specifiek zijn we geïnteresseerd in de parameter B , omdat deze in verband staat met de viscositeit.

Omdat $\sum (X_i - \bar{X}) = 0$, krijgen we voor de verwachtingswaarde van $\hat{B}$ de volgende uitdrukking:

$$\begin{aligned}
E(\hat{B}) &= E\left(\frac{\sum(X_i - \bar{X})Y_i - \bar{Y} \sum(X_i - \bar{X})}{\sum(X_i - \bar{X})^2}\right) \\
&= E\left(\frac{\sum(X_i - \bar{X})Y_i}{\sum(X_i - \bar{X})^2}\right) \\
&= \frac{\sum(X_i - \bar{X}) E(Y_i)}{\sum(X_i - \bar{X})^2} \\
&= \frac{\sum(X_i - \bar{X})(A + BX_i)}{\sum(X_i - \bar{X})^2} \\
&= \underbrace{\frac{\sum(X_i - \bar{X})A}{\sum(X_i - \bar{X})^2}}_{0, \text{ want } \sum(X_i - \bar{X}) = 0} + \frac{\sum(X_i - \bar{X})BX_i}{\sum(X_i - \bar{X})^2} \\
&= 0 + B \frac{\sum(X_i - \bar{X})X_i}{\sum(X_i - \bar{X})^2} \quad (*) \\
&= B
\end{aligned}$$

We zien dus dat $\hat{B}$ een zuivere schatter is van B . (*) is weergegeven in appendix A2. Voor de variantie van $\hat{B}$ geldt:

$$\begin{aligned}
\text{Var}(\hat{B}) &= \text{Var}\left(\frac{\sum(X_i - \bar{X})Y_i}{\sum(X_i - \bar{X})^2}\right) \\
&= \text{Var}\left(\frac{\sum(X_i - \bar{X})(A + BX_i + E_i)}{\sum(X_i - \bar{X})^2}\right) \\
&= \underbrace{\text{Var}\left(\frac{\sum(X_i - \bar{X})(A + BX_i)}{\sum(X_i - \bar{X})^2}\right)}_{\text{constant, want de } X_i\text{'s zijn bekend}} + \underbrace{\text{Var}\left(\frac{\sum(X_i - \bar{X})E_i}{\sum(X_i - \bar{X})^2}\right)}_{\text{Var}(aX)=a^2 \text{Var}(X)} \\
&= 0 + \frac{1}{(\sum(X_i - \bar{X})^2)^2} \left(\sum(X_i - \bar{X}) \text{Var}(E_i)\right)
\end{aligned}$$

De variantie van de residu's zijn van belang kijkend naar de variantie van onze schatter $\hat{B}$. Deze variantie moet dus benaderd worden. Hiervoor hebben we de volgende uitdrukking:

$$\widehat{\text{Var}(E_i)} = S_E^2 = \frac{\sum E_i^2}{n-2}$$

De term $n-2$ in de noemer verwijst naar de graden van vrijheid en komt voort uit het feit dat we n observaties hebben en twee parameters (A en B) schatten in ons model.

Gebruikmakend hiervan kunnen we een betrouwbaarheidsinterval voor parameter B opstellen met een gekozen significantieniveau α . Merk wel op dat in het benaderen van de variantie van B , er een schatter is gebruikt voor de variantie van de residu's. Dit introduceert extra onzekerheid in parameter B . De t -verdeling is geschikt om de extra onzekerheid te accommoderen. We construeren het betrouwbaarheidsinterval met significantieniveau α van B als volgt:

$$\hat{B} \pm t_{n-2, \alpha/2} \sqrt{\widehat{\text{Var}(\hat{B})}}$$

Waarbij $t_{n,\alpha/2}$ de kritieke waarde is van een t -distributie met n graden van vrijheid op significantieniveau $\alpha/2$.

Het doel is om de schattingsfout in de parameter B te minimaliseren. Dit houdt in een zo klein mogelijk betrouwbaarheidsinterval te vinden. De lengte van het betrouwbaarheidsinterval is gelijk aan $2 \cdot t_{n-2,\alpha/2} \sqrt{\widehat{\text{Var}}(B)}$. Er is geen invloed uit te oefenen op de kritieke waarde van de t -distributie (mits het aantal observaties gelijk blijft). Om de lengte van het interval te minimaliseren moet dus de variantie van B geminimaliseerd worden.

Voor het vervolg is het handiger om gebruik te maken van matrix-vector notatie. Het model verandert dan in

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \quad (4)$$

uitgeschreven:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 & X_1 \\ 1 & X_2 \\ \vdots & \vdots \\ 1 & X_n \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix} \quad (5)$$

Op een soortgelijke manier kan hiervoor de kleinste-kwadraten oplossing gevonden worden. We minimaliseren net als hiervoor $\sum E_i^2 = \boldsymbol{\epsilon}'\boldsymbol{\epsilon}$, met als resultaat [1]

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}$$

Uitgaande van homoskedastische fouten ($\text{Var } \boldsymbol{\epsilon} = \sigma^2 \mathbf{I}$) vinden we dan voor de verwachtingswaarde en variantie van B :

$$\begin{aligned} E(\hat{\boldsymbol{\beta}}) &= E((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y}) = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'E(\mathbf{y}) \\ &= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'E(\mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}) \\ &= (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{X}\boldsymbol{\beta} \\ &= \boldsymbol{\beta} \end{aligned}$$

$$\begin{aligned} \text{Var}(\hat{\boldsymbol{\beta}}) &= ((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}') \underbrace{\text{Var}(\mathbf{y})}_{\sigma^2 \mathbf{I}} ((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}')' \\ &= ((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}')\sigma^2 \mathbf{I} ((\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}')' \\ &= \sigma^2 (\mathbf{X}'\mathbf{X})^{-1} \end{aligned}$$

In alle berekeningen zijn we uitgegaan van een voorwaarde van lineaire regressie; constante variantie in de fouten (homoskedasticiteit). Bij de metingen van de molecuulsimulaties is dit niet het geval. Dit zou namelijk betekenen dat een meting in een kleine kubus een even goed resultaat zou geven als een meting in een grotere kubus. We maken juist gebruik van metingen in een grotere kubus omdat deze een nauwkeurigere schatting geven van de zelfdiffusiviteit. Een manier om deze voorwaarde te omzeilen is gebruik maken van gewogen regressie.

1.2 Gewogen regressie

Gewogen regressie is een uitbreiding van de gewone lineaire regressie. Gewogen regressie houdt rekening met heteroscedasticiteit (ongelijke variantie van de fouten) in de gegevens. In plaats van iedere meting even zwaar te laten meetellen bij het schatten van de parameters A en B , wordt aan iedere meting een gewicht toegekend dat aangeeft hoe belangrijk het is in de schatting. In ons geval betekent dit dat metingen die gedaan worden in een grotere kubus zwaarder meetellen voor de schatting, omdat de metingen daar nauwkeuriger zijn dan in een kleine kubus.

Bij gewogen regressie wordt de gewogen kwadratische fout (**W**eighted **S**um of **S**quares) geminimaliseerd om tot de beste fit te komen:

$$WSS = \sum_{i=1}^n w_i (\mathbf{Y}_i - \mathbf{X}_i \boldsymbol{\beta})^2 \quad (6)$$

waarbij $\mathbf{X}_i$ de i -de rij is van $\mathbf{X}$. Laat $\mathbf{W}$ de matrix zijn met gewichten w_i op de diagonaal. Dan minimaliseert de volgende $\hat{\boldsymbol{\beta}}$ de WSS in (6): [5]

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}' \mathbf{W} \mathbf{X})^{-1} \mathbf{X}' \mathbf{W} \mathbf{y}$$

De variantie van $\hat{\boldsymbol{\beta}}$ kan worden afgeleid: [5]

$$\begin{aligned} \text{Var}(\hat{\boldsymbol{\beta}}) &= (\mathbf{X}' \mathbf{W} \mathbf{X})^{-1} \mathbf{X}' \mathbf{W} \text{Var}(\mathbf{y}) \mathbf{W} \mathbf{X} (\mathbf{X}' \mathbf{W} \mathbf{X})^{-1} \\ &= (\mathbf{X}' \mathbf{W} \mathbf{X})^{-1} \mathbf{X}' \mathbf{W} \text{Var}(\boldsymbol{\epsilon}' \boldsymbol{\epsilon}) \mathbf{W} \mathbf{X} (\mathbf{X}' \mathbf{W} \mathbf{X})^{-1} \\ &= (\mathbf{X}' \mathbf{W} \mathbf{X})^{-1} \mathbf{X}' \mathbf{W} \boldsymbol{\Sigma} \mathbf{W} \mathbf{X} (\mathbf{X}' \mathbf{W} \mathbf{X})^{-1} \end{aligned}$$

Met

$$\boldsymbol{\Sigma} = \text{Var}(\boldsymbol{\epsilon}) = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \sigma_{n-1,n} \\ \sigma_{n1} & \cdots & \cdots & \sigma_{n,n-1} & \sigma_n^2 \end{bmatrix}$$

σ_i^2 is de variantie van meting i , σ_{ij} is de covariantie van meting i en j . We nemen aan dat de fouten in verschillende metingen geen verband met elkaar hebben, dus dat de covariantie tussen twee verschillende metingen i en j nul is ($\sigma_{i,j} = 0$). Dit heeft als gevolg dat $\boldsymbol{\Sigma}$ een diagonaalmatrix is.

Uit het gegeneraliseerde Gauss-Markov theorema volgt dat $\hat{\boldsymbol{\beta}}$ met $\mathbf{W} = \boldsymbol{\Sigma}^{-1}$ onder alle mogelijke lineaire, zuivere schatters van $\boldsymbol{\beta}$ de kleinste variantie heeft [5]. Dit is de matrix

$$\mathbf{W} = \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 & \cdots & \cdots & 0 \\ 0 & \frac{1}{\sigma_2^2} & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & \cdots & \cdots & 0 & \frac{1}{\sigma_n^2} \end{bmatrix}$$

En dus zijn de gewichten $w_i = 1/\sigma_i^2$ optimaal. Daaruit volgt voor $\text{Var}(\hat{\boldsymbol{\beta}})$

$$\text{Var}(\hat{\boldsymbol{\beta}}) = (\mathbf{X}' \mathbf{W} \mathbf{X})^{-1}$$

De waarden van A en B die hieruit volgen zijn [3]:

$$A = \frac{s_{xx}s_y - s_x s_{xy}}{\Delta}$$

$$B = \frac{s_{xy}s - s_x s_y}{\Delta}$$

Waarbij

$$s = \sum \frac{1}{\sigma_i^2}$$

$$s_x = \sum \frac{x_i}{\sigma_i^2}$$

$$s_{xx} = \sum \frac{x_i^2}{\sigma_i^2}$$

$$s_y = \sum \frac{y_i}{\sigma_i^2}$$

$$s_{yy} = \sum \frac{y_i^2}{\sigma_i^2}$$

$$\Delta = s_{xx}s - s_x^2$$

De variantie van de parameters A en B zijn niet meer zoals afgeleid in sectie 1.2, vanwege het gebruik van gewogen regressie. Het uitwerken van $\text{Var } \hat{\beta}$ geeft de covariantiematrix. Hierbij corresponderen de diagonale elementen van de matrix met de variantie van parameters, de non-diagonale elementen zijn de covariantie tussen de parameters.

$$V = \text{Var } \hat{\beta} = \frac{1}{\Delta} \begin{bmatrix} s_{xx} & -s_x \\ -s_x & s \end{bmatrix} := \begin{bmatrix} \text{Var}(A) & \text{Cov}(A, B) \\ \text{Cov}(A, B) & \text{Var}(B) \end{bmatrix}$$

We vinden voor de variantie van parameters A en B :

$$\sigma_A^2 = \frac{s_{xx}}{\Delta}, \quad \sigma_B^2 = \frac{s}{\Delta}$$

1.3 Probleemstelling

Op het interval $(0, 1]$ moeten punten $x_1, x_2, \dots, x_n$ waarop metingen worden uitgevoerd gekozen worden met bijbehorende gewichten $N_1, N_2, \dots, N_n$. De weging N_i geeft aan hoeveel metingen worden uitgevoerd punt x_i . Verder moet er aan de kostenvoorwaarde worden voldaan: De som van de computationele kosten van alle metingen mogen de maximale computationele kosten E niet overschrijden:

$$\sum_{i=1}^n N_i f(x_i) \leq E$$

$f(x)$ zijn de computationele kosten van het meten op punt x . De variantie in de helling B moet geminimaliseerd worden onder bovenstaande voorwaarde. Voor het gemak laten we nu de punten x_i gesorteerd zijn van klein naar groot, dus $x_1 < x_2 < x_3 < \dots$ etc.

Alle metingen die op een punt x_i worden uitgevoerd kunnen vervangen worden door één meting met een geschaalde variantie. Dit verandert niets aan de schatting van de parameters A en B . Dit is equivalent aan het uitvoeren van één geschaalde meting met variantie $S_i = \frac{\sigma_i}{\sqrt{N_i}}$.

1.3.1 Versimpelde weergave

Door eerst het probleem te bekijken zonder een van de voorwaarden, de kostenvoorwaarde, kan op een analytische manier gewichten bepaald worden die σ_B^2 minimaliseren.

We nemen eerst als variantiefunctie $\sigma(x) = x$ en kijken naar een situatie met twee punten, x_1 en x_2 . De variantie van de fout van het meten op het punt x_1 is in dit geval dus $\sigma(x_1)$. Voor ieder punt x_i waarop gemeten wordt geldt dat de variantie $\sigma_i = \frac{\sigma(x_i)}{\sqrt{N_i}} = \frac{x_i}{\sqrt{N_i}}$ is. En

$$\sigma_B^2 = \frac{s}{\Delta}$$

waarbij

$$\Delta = s_{xx}s - s_x^2$$

$$s = \sum \frac{1}{\sigma_i^2} = \sum \frac{N_i}{x_i^2}$$

$$s_x = \sum \frac{x_i}{\sigma_i^2} = \sum \frac{N_i}{x_i}$$

$$s_{xx} = \sum \frac{x_i^2}{\sigma_i^2} = \sum N_i = N$$

willen we minimaliseren.

Een standaard methode om dit te doen is om de partiële afgeleiden van σ_B^2 te nemen en deze gelijk te stellen aan 0. Als er een punt bestaat in ons domein zodat alle partiële afgeleiden gelijk zijn aan 0, heeft de functie een extremum. Dat hoeft er alleen nog maar gecheckt te worden of het een minimum is.

De functie is symmetrisch in zijn variabelen, dus we zullen hetzelfde antwoord vinden voor iedere variabele waar we een partiële afgeleide naar nemen. Voor nu nemen we aan dat de punten x_i gefixeerd zijn en $\sigma_B^2(N_1, N_2)$ alleen een functie is van de gewichten. Zonder verlies van algemeenheid nemen we de partiële afgeleide naar N_1 om een extremum te vinden:

$$\begin{aligned} \frac{d\sigma_B^2}{dN_1} &= \frac{(ss_{xx} - s_x^2)s' - s((s's_{xx} + ss'_{xx}) - 2s_x s'_x)}{(ss_{xx} - s_x^2)^2} \\ &= \frac{2ss_x s'_x - s_x^2 s' - s^2 s'_{xx}}{(ss_{xx} - s_x^2)^2} := 0 \\ &\iff 2ss_x s'_x - s_x^2 s' - s^2 s'_{xx} = 0 \end{aligned}$$

met

$$s' = \frac{1}{x_1^2} - \frac{1}{x_2^2}$$

$$s'_x = \frac{1}{x_1} - \frac{1}{x_2}$$

$$s'_{xx} = 0$$

$N_1 = \frac{x_1}{x_1+x_2}N$ en $N_2 = \frac{x_2}{x_1+x_2}N$ zijn oplossingen van bovenstaande vergelijking. We kijken nu naar een soortgelijke situatie met een andere variantiefunctie: $\sigma(x) = x^2$. Hierbij geldt

$$s = \sum \frac{1}{\sigma_i^2} = \sum \frac{N_i}{x_i^4}$$

$$s_x = \sum \frac{x_i}{\sigma_i^2} = \sum \frac{N_i}{x_i^3}$$

$$s_{xx} = \sum \frac{x_i^2}{\sigma_i^2} = \sum \frac{N_i}{x_i^2}$$

$$s' = \frac{1}{x_1^4} - \frac{1}{x_2^4}$$

$$s'_x = \frac{1}{x_1^3} - \frac{1}{x_2^3}$$

$$s'_{xx} = \frac{1}{x_1^2} - \frac{1}{x_2^2}$$

$N_1 = \frac{x_1^2}{x_1^2+x_2^2}N$ en $N_2 = \frac{x_2^2}{x_1^2+x_2^2}N$ zijn een oplossing van $\frac{d\sigma_B^2}{dN_1} = 0$. Het blijkt dat voor alle variantiefuncties $\sigma(x) = x^n$, de oplossingen voor twee punten $N_1^* = \frac{x_1^n}{x_1^n+x_2^n}N$ en $N_2^* = \frac{x_2^n}{x_1^n+x_2^n}N$ zijn. We substitueren N_1^*, N_2^* in s, s_x en s_{xx} en vinden

$$s^* = \frac{1}{(x_1^n + x_2^n)}(x_1^{-n} + x_2^{-n})$$

$$s_x^* = \frac{1}{(x_1^n + x_2^n)}(x_1^{1-n} + x_2^{1-n})$$

$$s_{xx}^* = \frac{1}{(x_1^n + x_2^n)}(x_1^{2-n} + x_2^{2-n})$$

En het criterium was $2ss_x s'_x - s_x^2 s' - s^2 s'_{xx} = 0$. Dan volgt

$$\begin{aligned}
2s^* s_x^* s'_x &= \frac{2}{(x_1^n + x_2^n)^2} & (x_1^{-n} + x_2^{-n})(x_1^{1-n} + x_2^{1-n})(x_1^{1-2n} + x_2^{1-2n}) \\
&= \frac{2}{(x_1^n + x_2^n)^2} & (x_1^{2-4n} + x_1^{1-3n} x_2^{1-n} + x_1^{2-3n} x_2^{-n} + x_1^{1-2n} x_2^{1-2n} \\
& & - x_1^{1-2n} x_2^{1-2n} - x_1^{-n} x_2^{2-3n} - x_1^{1-n} x_2^{1-3n} - x_2^{2-4n}) \\
(s_x^*)^2 s' &= \frac{1}{(x_1^n + x_2^n)^2} & (x_1^{1-n} + x_2^{1-n})^2 (x_1^{-2n} + x_2^{-2n}) \\
&= \frac{1}{(x_1^n + x_2^n)^2} & (x_1^{2-4n} + 2x_1^{1-3n} x_2^{1-n} + x_1^{-2n} x_2^{2-2n} \\
& & - x_1^{2-2n} x_2^{-2n} - 2x_1^{1-n} x_2^{1-3n} - x_2^{2-4n}) \\
(s^*)^2 s'_{xx} &= \frac{1}{(x_1^n + x_2^n)^2} & (x_1^{-n} + x_2^{-n})^2 (x_1^{2-2n} - x_2^{2-2n}) \\
&= \frac{1}{(x_1^n + x_2^n)^2} & (x_1^{2-4n} + 2x_1^{2-3n} x_2^{-n} + x_1^{2-2n} x_2^{-2n} \\
& & - x_1^{-2n} x_2^{2-2n} + 2x_1^{-n} x_2^{2-3n} - x_2^{2-4n})
\end{aligned}$$

En dus inderdaad geldt $2ss_x s'_x - s_x^2 s' - s^2 s'_{xx} = 0$. Omdat $\sigma(x) = x^n$ logische natuurlijke keuzes zijn voor variantiefuncties, en dan in het specifiek $\sigma(x) = x$, $\sigma(x) = x^2$ en $\sigma(x) = x^3$ beperken we ons hiertoe in het vervolg.

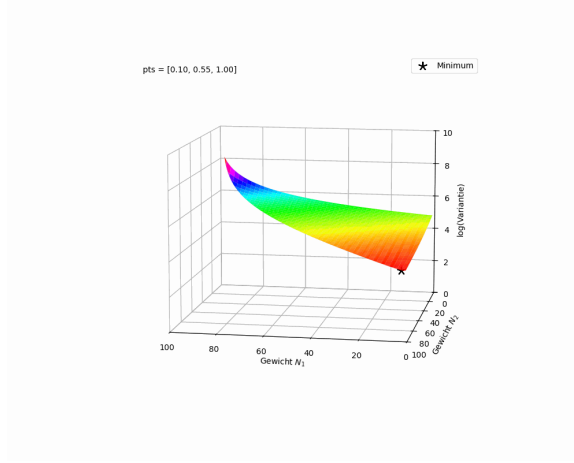
We hebben gevonden voor welke gewichten de eerste afgeleide gelijk is aan nul. Het andere criterium voor een minimum is dat de tweede afgeleide positief is ($\frac{d^2 \sigma_B^2}{dN_1^2} > 0$) in het punt waar de eerste afgeleide gelijk is aan nul. Een analytische benadering levert het volgende op:

$$\frac{d^2 \sigma_B^2}{dN_1^2} = \frac{s'' \Delta^2 - s \Delta \Delta'' - 2s' \Delta \Delta' + 2s(\Delta')^2}{\Delta^3}$$

Dit is een vrij complexe uitdrukking en het lijkt niet mogelijk op een analytische manier, noch met een computer algebra systeem als Maple of Mathematica te bewijzen dat de tweede afgeleide groter is dan 0 in het extremum. Numerieke methoden kunnen hier een oplossing voor bieden. Met numerieke methoden kan bekeken worden of de functie convex is.

Een functie is convex op een domein D als $f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$ voor alle $x, y \in D$ en voor alle $\lambda \in [0, 1]$. Dit heeft als visuele interpretatie dat als er een rechte lijn wordt getrokken tussen $f(x)$ en $f(y)$, dat de grafiek van de functie onder deze lijn ligt. De functie heeft een 'holle' vorm. Als de functie convex is is een stationair punt van de functie altijd een minimum. In figuur 2, waarvoor de code in appendix B staat, ziet de functie er inderdaad 'hol' uit. Dat suggereert convexiteit.

Er kan op een numerieke manier geverifieerd worden dat de functie inderdaad convex is. De code voor deze verificatie is weergegeven in appendix C. In deze implementatie wordt geverifieerd dat voor drie punten de variantiefunctie convex is. De gewichten $N_1^* = \frac{x_1^n}{x_1^n + x_2^n}$ en $N_2^* = \frac{x_2^n}{x_1^n + x_2^n}$ minimaliseren inderdaad de variantie van B . Uiteraard kan het programma aangepast worden naar een situatie met een andere variantiefunctie en een ander aantal metingen.



Figuur 2: Animatie van hoe de variantie van B verandert ten opzichte van een verandering van het tweede punt. Op de x-as en y-as het gewicht van meetpunten x_1 en x_2 , met 100 totale metingen, dus $N_1 + N_2 + N_3 = 100$. Hier is gekozen voor variantie $\sigma(x) = x^2$.

1.4 Kostenbeperking

In de vorige sectie is een versimpeld probleem bekeken en hiervoor de optimale gewichten bepaald. Hierbij is de beperking door meetkosten buiten beschouwing gelaten. Nu introduceren we deze restrictie. De beperking is van de volgende vorm:

$$\sum_{i=1}^n N_i f(x_i) \leq E$$

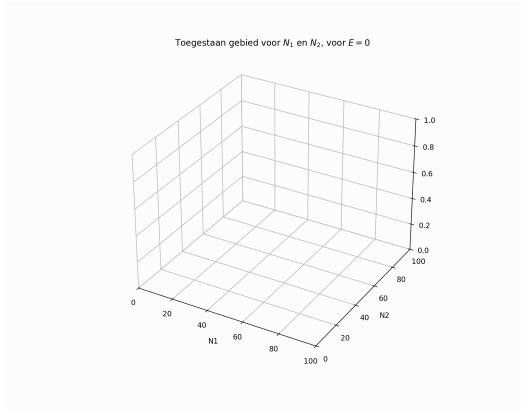
$f(x_i)$ is de kostenfunctie geëvalueerd in het punt x_i en representeert hoeveel het kost om bij x_i te meten. In de moleculaire dynamica komen kostenfuncties als $f(x) = \frac{1}{x}$ of $f(x) = \frac{1}{x^2}$ voor.

Deze voorwaarde beperkt de waardes die de gewichten aan kunnen nemen. Dit is gevisualiseerd in figuur 3a. Hoe kleiner de totale kosten E , des te minder waardes mogen N_1, N_2 (en N_3) aannemen.

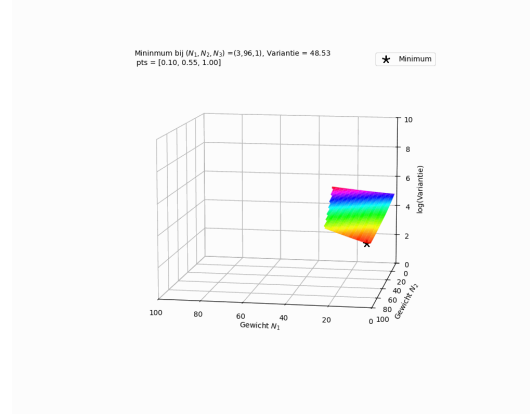
Door deze plot en die uit de vorige sectie te combineren kan een voorstelling gemaakt worden voor een minimum voor de variantie van B dat binnen de beperkingen ligt. Dit is weergegeven in figuur 3b, en de code hiervoor staat in appendix D. Hierin is het toegestane gebied voor N_1, N_2 verwerkt en binnen dit toegestane gebied is de variantie geplotted.

1.4.1 Twee meetpunten

De theoretische optimale gewichten voor een situatie met twee punten zijn bepaald. Nu kan er naar de optimale punten gekeken worden die bij deze gewichten horen. Eerst beschouwen we een situatie met twee punten. Figuur 4 doet vermoeden dat een situatie waarbij de meetpunten gelijke variantie hebben en zo ver mogelijk uit elkaar liggen de kleinste variantie heeft.



(a) Animatie van hoe het toegestane gebied van de metingen N_1 en N_2 verandert naarmate de meetkosten E veranderen, met kostenfunctie $f(x) = \frac{1}{x}$. Er worden in totaal 100 metingen uitgevoerd.



(b) Animatie van de variantie van B , en hoe het toegestane gebied van aantal metingen N_1, N_2 en N_3 varieert met de meetpunten, 100 totale metingen, kostenfunctie $f(x) = \frac{1}{x}$ en $E = 400$.

Als we dit analytisch bekijken voor een constante σ :

$$\begin{aligned}
 \frac{s}{s_{xx}s - s_x^2} &= \frac{\sum \frac{1}{\sigma_i^2}}{\sum \frac{1}{\sigma_i^2} \sum \frac{x_i^2}{\sigma_i^2} - \left(\sum \frac{x_i}{\sigma_i}\right)^2} = \frac{\frac{2}{\sigma^2}}{\frac{2}{\sigma^2} \frac{x_1^2 + x_2^2}{\sigma^2} - \left(\frac{x_1 + x_2}{\sigma^2}\right)^2} \\
 &= \frac{2\sigma^2}{2(x_1^2 + x_2^2) - (x_1 + x_2)^2} \\
 &= \frac{2\sigma^2}{x_1^2 - 2x_1x_2 + x_2^2} \\
 &= \frac{2\sigma^2}{(x_1 - x_2)^2}
 \end{aligned}$$

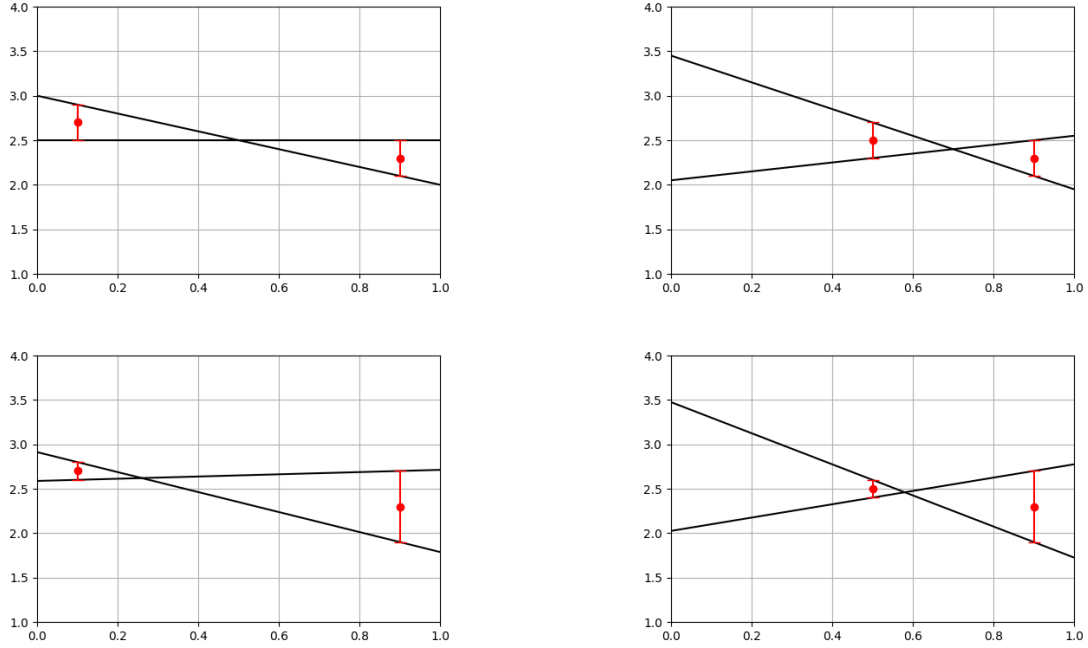
Deze variantie is zo klein mogelijk als de noemer zo groot mogelijk is. Hiervoor moet het verschil tussen x_1 en x_2 zo groot mogelijk zijn, en dus moeten deze punten zo ver mogelijk uit elkaar worden gekozen op $(0, 1]$. Figuur 4 suggereert ook dat een opstelling van twee punten minimale variantie heeft als de metingen zo verdeeld worden, dat de punten gelijke variantie hebben. Om dit te verifiëren definiëren we een optimaliseringsprobleem dat met behulp van Python opgelost kan worden:

Minimaliseer:

$$\frac{s}{s_{xx}s - s_x^2} = \frac{\sum \frac{1}{\sigma_i^2}}{\sum \frac{1}{\sigma_i^2} \sum \frac{x_i^2}{\sigma_i^2} - \left(\sum \frac{x_i}{\sigma_i}\right)^2}$$

S.t.:

$$\begin{aligned}
 \sum_{i=1}^n N_i f(x_i) &\leq E, \\
 x_i &\in (0, 1], & i = 1, \dots, n \\
 N_i &\in \mathbb{N}, & i = 1, \dots, n
 \end{aligned}$$



Figuur 4: Verschil in onzekerheid in de helling bij een lijn door twee punten bij twee verschillende waarden van x . In de subfiguur linksboven zien we de kleinste variantie in de helling, dit is een situatie met twee meetpunten die ver van elkaar af liggen en gelijke variantie hebben. In de subfiguur rechtsboven is een situatie weergegeven waarbij de punten gelijke variantie hebben maar dicht bij elkaar liggen. In de subfiguren onderin zijn situaties weergegeven waarbij de punten ongelijke variantie hebben.

In appendix E is de code toegevoegd voor het oplossen van het bovenstaande optimaliseringsprobleem met $n = 2$. Uit het resultaat van de optimalisatie blijkt dat de gewichten zo gekozen worden, dat de variantie van ieder punt gelijk is. Bijvoorbeeld: Voor $x_1 = 0.5$ en $x_2 = 1$ worden bij variantiefunctie $\sigma(x) = x^2$ de gewichten $N_1 = 20$ en $N_2 = 80$ gekozen, zodat $\sigma_1^2 = \frac{\sigma^2(x_1)}{N_1} = \frac{0.25}{20} = 0.0125$ en $\sigma_1^2 = \frac{\sigma^2(x_2)}{N_2} = \frac{1}{80} = 0.0125$.

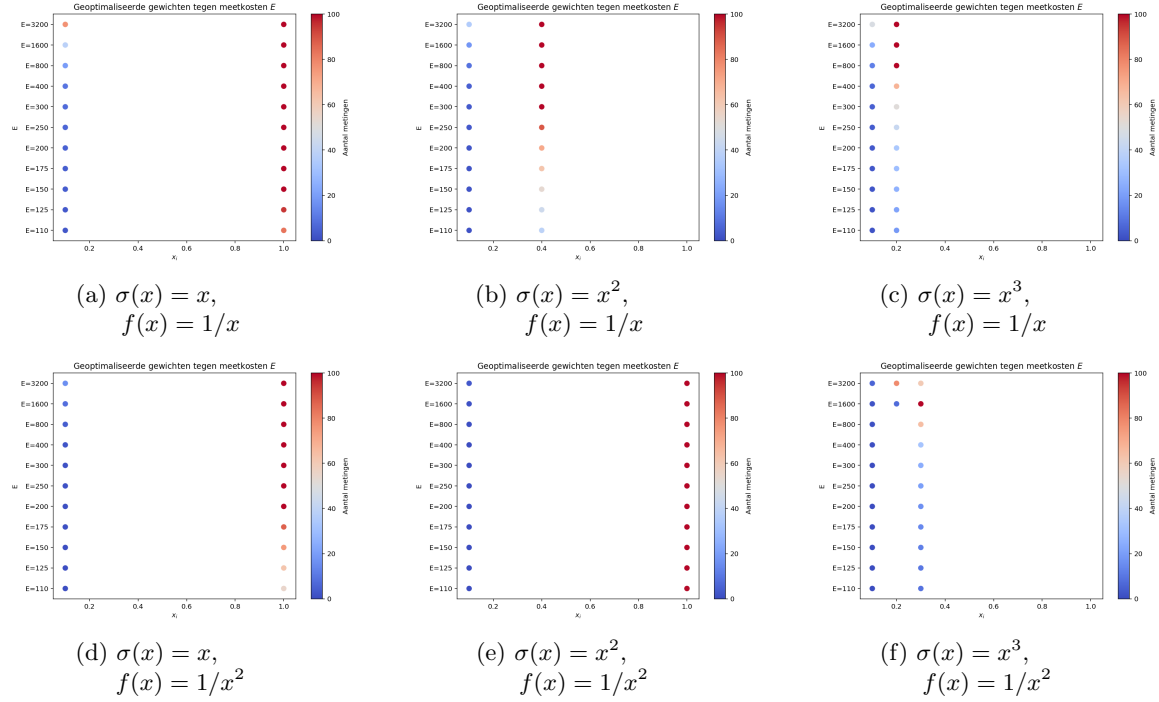
Het lijkt erop dat het minimum slechts door twee meetpunten wordt gevormd. Figuur 2 ondersteunt dit, waar zichtbaar is dat het minimum altijd op de contour ligt. Een minimum op de contour impliceert dat minstens één punt een gewicht van 0 heeft. Dit wordt verder uitgezocht in het volgende hoofdstuk.

1.4.2 Meer dan twee meetpunten

Wanneer de meetkosten geen beperking vormen, is een opstelling met twee meetpunten optimaal. Wanneer de meetkosten wel een beperking vormen, verandert de optimale opstelling afhankelijk van deze beperking. In appendix F is de code voor de figuren in figuur 5. In de figuur is weergegeven welke punten uit de reeks $(0.1, 0.2, \dots, 1)$ met welke bijbehorende gewichten optimaal zijn, afhankelijk van de variantiefunctie $\sigma(x)$ en kostenfunctie $f(x)$, uitgezet tegen bepaalde waarden voor meetkosten

E . De meetkosten gaan van boven naar beneden omlaag, dus bovenaan zijn de parameters het minst begrensd, onderaan het meest.

In het programma wordt de variantie van B omhoog geschaald met een factor 10^6 , omdat er anders problemen optreden met te kleine getallen bij de 'scipy.optimize' functie. Er is gekozen voor 10 verschillende punten omdat dit nog binnen de mogelijkheden van de optimalisatie past. Meer dan 10 parameters zorgt voor problemen bij de optimalisatie, en leidt in sommige gevallen niet tot de optimale oplossing.



Figuur 5: Optimale punten en bijbehorende gewichten, waarbij gekozen kon worden uit $x_i = 0.1, 0.2, \dots, 0.9, 1$, voor drie verschillende variantiefuncties $\sigma(x)$ en twee verschillende kostfuncties $f(x)$, uitgezet tegen 10 verschillende meetkosten E . Des te meer metingen op een punt worden uitgevoerd, hoe roder het is. Blauw geeft aan dat er weinig metingen worden uitgevoerd. In totaal zijn er 100 metingen uitgevoerd.

Er zijn zekere patronen te herkennen afhankelijk van de kostenfuncties en variantiefuncties. $\frac{1}{x} < \frac{1}{x^2}$ voor $x \in (0, 1]$, dus in de bovenste drie figuren 5a, 5b en 5c met kostenfunctie $f(x) = \frac{1}{x}$ is de kostenbeperking minder streng dan in de onderste drie figuren 5d, 5e en 5f, waar $f(x) = \frac{1}{x^2}$. In de meeste gevallen kiest de optimalisatie voor een oplossing met twee punten. Welke twee punten dan gekozen worden, wordt bepaald door de variantiefunctie. Bij $\sigma(x) = x$ worden twee punten gekozen die zo ver mogelijk uit elkaar liggen, bij een variantiefunctie met hogere macht gebeurt dit niet. Daar wordt gekozen voor een tweede punt dat dichterbij 0 ligt. De reden hiervoor is dat bij een variantiefunctie met een hogere macht, de kleinere variantie die meekomt met een meetpunt dat dichterbij nul ligt zwaarder meetelt dan het verschil van de twee meetpunten.

Voor de twee linker subfiguren 5a en 5d is de variantiefunctie $\sigma(x) = x$. In deze twee situaties wordt er gekozen voor een groot gewicht bij 1, en een klein gewicht bij 0.1. Dit komt overeen met

wat is besproken over $\sigma(x) = x$.

In subfiguur 5b wordt er als tweede punt een punt dicht bij 0 gekozen. Dit is te verklaren door de hogere macht van $\sigma(x)$; De variantie van een punt dat dicht bij 0 ligt is kleiner dan bij $\sigma(x) = x$. Subfiguur 5e lijkt meer op de linker twee subfiguren. Doordat de kostenfunctie nu van een hogere orde is, verschuift het tweede punt meer naar rechts.

In de rechter twee subfiguren 5c en 5f is te zien dat twee punten die dicht bij nul liggen gunstig is, vanwege hetzelfde argument als bij subfiguur 5b. Hier is de macht van σ nog hoger dan bij $\sigma(x) = x^2$, waardoor een meting dichtbij 0 een nog kleinere variantie heeft. Dit verschil tussen de middelste en de rechter figuren is ook zichtbaar; de middelste figuren prioriseren een groter gewicht met een punt verder weg van 0 boven een kleiner gewicht met een punt dicht bij 0. De rechter figuren prioriseren een kleiner gewicht met een punt dicht bij 0 boven een groter gewicht met een punt verder weg van 0.

1.5 Intercept

Kijkend naar vergelijking (1) kan ook de zelfdiffusiviteit van een stof bepaald worden door het schatten van parameter A . In sectie 1.2 is bepaald wat de variantie van deze parameter A is. De parameter A heeft een soortgelijke constructie voor het betrouwbaarheidsinterval als parameter B en deze was bepaald in sectie 1.2:

$$\hat{A} \pm t_{n-2, \alpha/2} \sqrt{\widehat{\text{Var}}(A)}$$

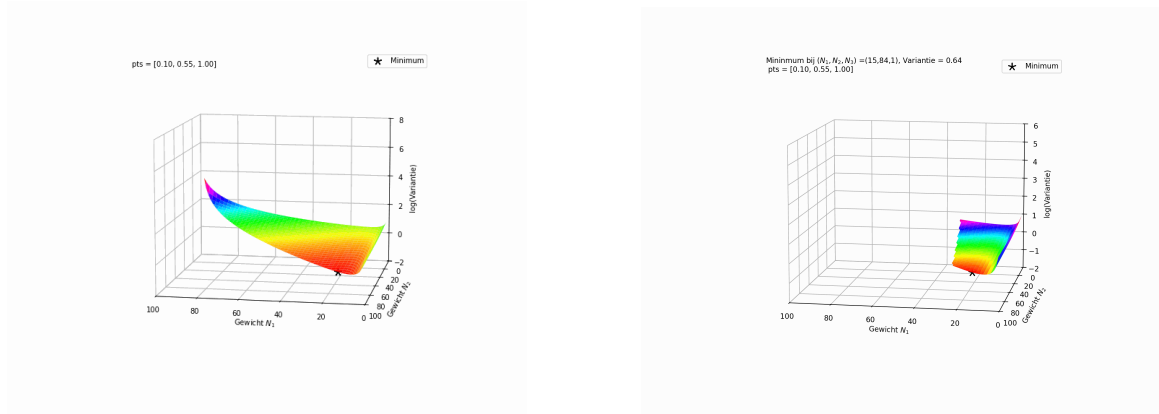
Het betrouwbaarheidsinterval heeft lengte $2 \cdot t_{n-2, \alpha/2} \sqrt{\widehat{\text{Var}}(A)}$ en het doel is deze te minimaliseren. Dit komt, vergelijkbaar als bij parameter B , neer op het minimaliseren van σ_A^2 :

$$\sigma_A^2 = \frac{s_{xx}}{\Delta}$$

De formules voor de varianties van A en B lijken op elkaar. De analyses die voor de variantie van B zijn uitgevoerd kunnen op een analoge manier voor die van A worden uitgevoerd. Omdat $x_i \leq 1$ is de variantie van A is altijd kleiner dan die van B , immers:

$$\sigma_A^2 = \frac{s_{xx}}{\Delta} = \frac{\sum \frac{x_i^2}{\sigma_i^2}}{\Delta} \leq \frac{\sum \frac{1}{\sigma_i^2}}{\Delta} = \sigma_B^2$$

Dit is ook te zien in figuren 6a en 6b:



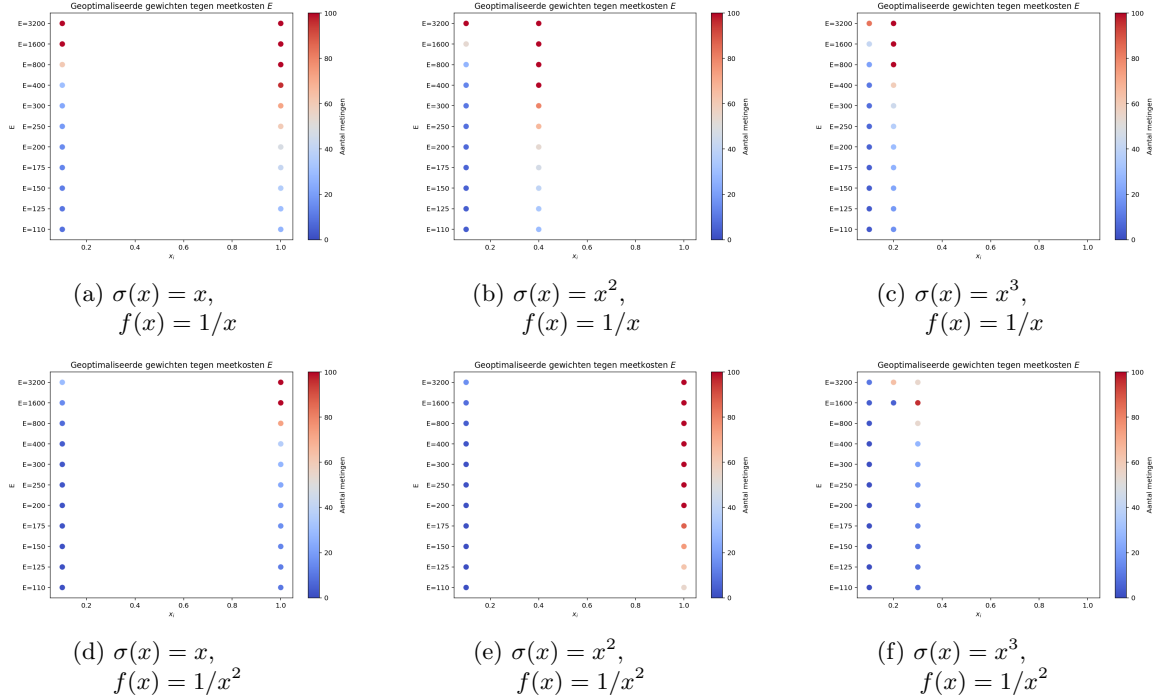
(a) Animatie van hoe de variantie van A verandert ten opzichte van een verandering van het tweede punt. Op de x-as en y-as het gewicht van meetpunten x_1 en x_2 , met 100 totale metingen, dus $N_1 + N_2 + N_3 = 100$ en $\sigma(x) = x^2$.

(b) Animatie van de variantie van A , en hoe het toegestane gebied van aantal metingen N_1, N_2 en N_3 varieert met de meetpunten, 100 totale metingen, kostenfunctie $f(x) = \frac{1}{x}$ en $E = 400$.

Figuur 6

In figuur 6a en 6b zijn de figuren 2 en 3b nagebootst voor de variantie van A . De figuren vertonen dezelfde eigenschappen als de figuren voor de variantie van B , met uitzondering van dat de variantie van A ongeveer twee ordes van grootte kleiner is.

Figuur 7 is een aangepaste versie van figuur 5, waarin de variantie van A wordt weergegeven. In beide figuren worden dezelfde meetpunten gebruikt, maar de gewichten die aan de punten worden



Figuur 7: Optimale punten en bijbehorende gewichten, waarbij gekozen kon worden uit $x_i = 0.1, 0.2, \dots, 0.9, 1$, voor drie verschillende variantiefuncties $\sigma(x)$ en twee verschillende kostfuncties $f(x)$, uitgezet tegen 10 verschillende meetkosten E . Des te meer metingen op een punt worden uitgevoerd, hoe roder het is. Blauw geeft aan dat er weinig metingen worden uitgevoerd.

toegekend verschillen. Voor de variantie van A krijgt het kleinste meetpunt een groter gewicht en het grootste meetpunt juist een kleiner gewicht in vergelijking met de variantie van B . Dit verschil kan worden verklaard door te kijken naar s en s_{xx} :

$$s = \sum \frac{1}{\sigma_i^2} = \sum \frac{N_i}{\sigma(x_i)^2}, \quad s_{xx} = \sum \frac{x_i^2}{\sigma_i^2} = \sum \frac{x_i^2 N_i}{\sigma(x_i)^2}$$

Bij s_{xx} wordt aan het kleinste meetpunt een groter gewicht toegewezen doordat x_i^2 (dat kleiner is dan 1 voor het kleinste punt) compenseert voor de kleine waarde van x_i . Hierdoor kunnen grotere gewichten worden toegekend aan kleinere meetpunten.

2 Simultaan schatten

Eerder zijn de onzekerheid in de parameters A en B apart geschat. Maar, de meetpunten en gewichten die de onzekerheid in B minimaliseren, minimaliseren die van A niet altijd en vice versa. Het zou beter zijn om beide parameters te schatten aan de hand van dezelfde metingen, in plaats van aparte metingen te moeten doen voor beide variabelen.

Als we voor onze twee parameters apart een betrouwbaarheidsinterval van het niveau 0.95 hebben, is de kans dat beide parameters simultaan in deze intervallen liggen in het slechtste geval $1 - 0.05 - 0.05 = 0.90$. Er kan een beter simultaan betrouwbaarheidsinterval verkregen worden door gebruik te maken van Bonferronibetrouwbaarheidsintervallen. Hierbij wordt er gebruik gemaakt van aparte betrouwbaarheidsintervallen voor beide parameters en berust op de Bonferroniongelijkheid:

$$P\left(\bigcup_{i=1}^k A_i\right) \leq \sum_{i=1}^k P(A_i)$$

Laat $\mathcal{A}_A$ een betrouwbaarheidsinterval van A zijn met niveau α_A , dus $P(A \in \mathcal{A}_A) = 1 - \alpha_A$ en laat $\mathcal{B}_B$ een betrouwbaarheidsinterval van B zijn met niveau α_B , dus $P(B \in \mathcal{B}_B) = 1 - \alpha_B$. Dan is $(A \in \mathcal{A}_A) \cap (B \in \mathcal{B}_B)$ het evenement dat zowel A en B in hun betrouwbaarheidsinterval van niveau α_A respectievelijk α_B liggen. Dan:

$$\begin{aligned} P((A \in \mathcal{A}_A) \cap (B \in \mathcal{B}_B)) &= 1 - P((A \notin \mathcal{A}_A) \cup (B \notin \mathcal{B}_B)) \\ &\geq 1 - (P(A \notin \mathcal{A}_A) + P(B \notin \mathcal{B}_B)) \\ &= 1 - (\alpha_A + \alpha_B) \end{aligned}$$

Stel dat we een gezamenlijk betrouwbaarheidsinterval zoeken van niveau 0.95 voor A en B , moeten α_A, α_B voldoen aan $\alpha_A + \alpha_B = 0.05$.

Bonferroni wordt vaak gebruikt voor het schatten van betrouwbaarheidsintervallen waarbij er niets of weinig bekend is over de gezamenlijke verdeling (covariantie) van de verschillende parameters. De covariantiematrix van A en B is: [3]

$$V = \frac{1}{\Delta} \begin{bmatrix} s_{xx} & -s_x \\ -s_x & s \end{bmatrix}$$

Hierin kan de covariantie tussen A en B helpen om de schattingen te verfijnen. Door deze covariantiematrix te gebruiken, kan men de afhankelijkheid tussen de schattingen van A en B meenemen, wat leidt tot een nauwer betrouwbaarheidsinterval, waar bij Bonferroni geen rekening mee gehouden wordt. Om de significantie van een regressiemodel te beoordelen, wordt vaak de F-statistiek gebruikt. De F-statistiek kan gezien worden als een multidimensionale extensie van de t-test, om meerdere parameters gezamenlijk te toetsen. We stellen hypotheses op die toetsen of de gewogen kleinste kwadraten parameters in β gelijk zijn aan de gevonden waarden β_0 :

$$H_0 : \beta = \beta_0 \quad \text{tegen} \quad H_1 : \beta \neq \beta_0$$

Om de hypotheses te toetsen wordt er gebruik gemaakt van toetsingsgrootheid F_0 . [1]

$$F_0 = \frac{(\beta - \beta_0)' V^{-1} (\beta - \beta_0)}{q S_E^2}$$

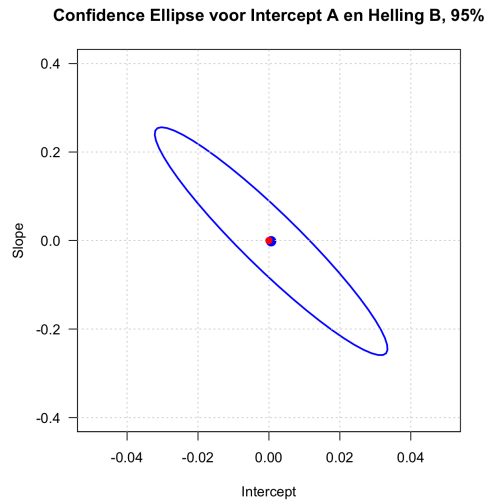
Onder H_0 heeft F_0 een $F_{q,n-k}$ verdeling waarbij q het aantal regressievariabelen is dat getest wordt, n het aantal observaties en k het totale aantal parameters in het model is. We testen in dit geval zowel A als B , en deze zijn de enige twee parameters in het model, dus $q = 2$ en $k = 2$.

F_0 kan gebruikt worden om een gebied te omschrijven waarin β ligt met een gekozen betrouwbaarheidsniveau. Omdat F_0 een $F_{2,n-2}$ verdeling heeft, volgt

$$P(F_0 \leq F_{2,n-2,\alpha}) = 1 - \alpha$$

waarbij $F_{2,n-2,\alpha}$ de kritieke waarde is van de F -verdeling van niveau α . Hieruit kan worden opge-
maakt dat iedere β die voldoet aan $(\beta - \beta_0)'V^{-1}(\beta - \beta_0) \leq qS_E^2 F_{2,n-2,\alpha}$ in dit gebied ligt. Deze ongelijkheid beschrijft een ellipsoïde die gecentreerd is rond β_0 .

Een zogenaamde 'confidence ellips' van niveau $1 - \alpha$ omschrijft een gebied waarin, als er eenzelfde steekproef gehouden zou worden met evenveel observaties, β_0 met een zekerheid van $1 - \alpha$ ook in zou liggen. In figuur 8 is een voorbeeld van zo een ellips weergegeven.



Figuur 8: Ellipsoïde van niveau 0.95 voor parameters A en B . $(0,0)$ is weergegeven in het rood.

Om deze ellips te construeren is gebruik gemaakt van de optimale punten en gewichten die horen bij de optimalisatie van de variantie van B met $\sigma(x) = x^3$ en $f(x) = \frac{1}{x^2}$. Er is gekozen voor de variantie van B boven die van A , de motivatie voor deze keuze is dat de variantie van A in alle gevallen toch al kleiner is dan de variantie van B . We hebben dan liever een kleinere variantie van B . De ellips is schuin afgebogen, van linksboven naar rechtsonder. Dit impliceert dat de parameters A en B een correlatie hebben. Bij een negatieve correlatie buigt een ellips van linksboven naar rechtsonder, bij positieve correlatie buigt deze van linksonder naar rechtsboven, en bij een correlatie dichtbij 0 lijkt de ellips op een cirkel. We zien dus dat de parameters A en B een negatieve correlatie met elkaar hebben, wat eerder bevestigd was in de covariantiematrix.

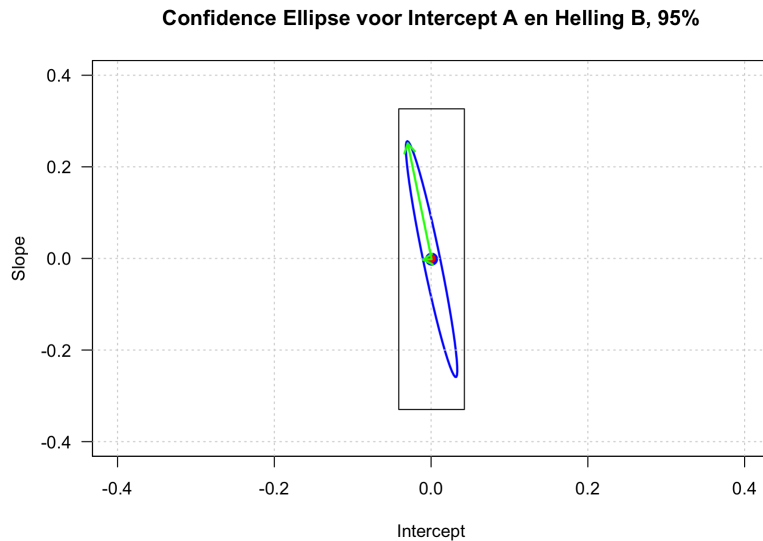
De oppervlakte van een betrouwbaarheidsgebied is een maat voor de mate van onzekerheid van de parameters. In het geval van een Bonferronibetrouwbaarheidsinterval kan deze onzekerheid worden gevisualiseerd als een rechthoekig gebied in twee dimensies. We overwegen hierbij een betrouwbaarheidsniveau van $\alpha = 0.95$.

De oppervlakte van het Bonferronibetrouwbaarheidsgebied kan eenvoudig worden berekend door de lengte van de afzonderlijke betrouwbaarheidsintervallen voor elke parameter te vermenigvuldigen. Aangezien het een simultaan betrouwbaarheidsinterval betreft, moeten de afzonderlijke betrouwbaarheidsintervallen een niveau kiezen dat overeenkomt met het totale betrouwbaarheidsniveau van 0.95. Een logische keuze is om de afzonderlijke intervallen op een betrouwbaarheidsniveau van $1 - \alpha/2 = 0.975$ te plaatsen.

De oppervlakte van de ellipsoïde is ook een maat voor de mate van onzekerheid. Deze oppervlakte wordt berekend door $\pi \cdot a \cdot b$, waarbij a en b de lengtes van de halve lange- en korte assen zijn die de ellips definiëren.

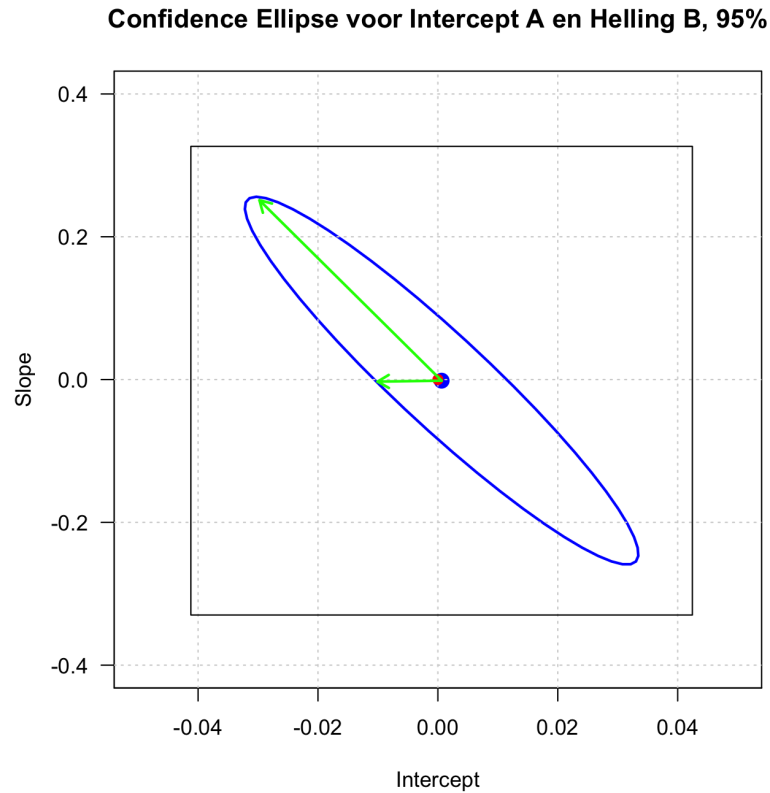
Deze lengtes zijn verbonden aan de eigenwaarden en eigenvectoren van de covariantiematrix. De orthonormale eigenvectoren zijn de richtingen van de assen van de ellips, de eigenwaarden staan in verband met de lengte van de assen. Omdat het betrouwbaarheidsniveau een rol speelt bij de grootte van de ellips (een groter betrouwbaarheidsniveau betekent een grotere ellips), zijn de lengte van de halve lange- en korte as van de ellipsoïde verbonden aan het betrouwbaarheidsniveau. De factor die hiervoor compenseert is de kritieke waarde van de χ^2 verdeling: $\chi^2_{0.95, n-2}$. Deze compenseert voor het onzekerheidsniveau bij het simultaan beoordelen van meerdere variabelen. De lengtes van de assen schalen met $\sqrt{\lambda_i \cdot \chi^2_{0.95, n-2}}$, waarbij λ_i de eigenwaarden van de covariantiematrix zijn [4]. Logischerwijs hoort de grootste eigenwaarde bij de langste as, de kleinste eigenwaarde bij de kortste as.

Het gezamenlijke Bonferroni betrouwbaarheidsgebied, de betrouwbaarheidsellips, en de assen van de ellips zijn in figuur 9 weergegeven. De R-code voor deze figuur is te vinden in appendix G.



Figuur 9: Bonferroni betrouwbaarheidsgebied van niveau 0.95 (zwarte rechthoek), betrouwbaarheidsellips van niveau 0.95, met de geschaalde eigenvectoren in het groen. (0,0) is in het rood weergegeven.

In figuur 10 is figuur 9 geschaald voor visuele interpretatie. Dit heeft als gevolg dat het lijkt alsof de eigenvectoren niet orthogonaal zijn, maar dit wordt veroorzaakt door de geschaalde assen.



Figuur 10: Figuur 9 geschaald voor visuele doeleinden.

De lengten van de korte en lange as zijn 0.461 en 0.019, wat een oppervlak geeft van $\pi \cdot 0.461 \cdot 0.019 = 0.0275$. De oppervlakte van het gebied omschreven door de Bonferroni betrouwbaarheidsintervallen is 0.1797. $\frac{0.1797 - 0.0275}{0.1797} \approx 0.847$, dus het toevoegen van de covariantie tussen A en B vermindert in dit geval de onzekerheid met zo'n 85%.

Conclusie/Discussie

In dit verslag zijn de ribbelengtes en bijbehorende gewichten van kubussen bepaald om de viscositeit zo nauwkeurig mogelijk te schatten, rekening houdend met computationele kosten. De resultaten laten zien dat de optimale ribbelengtes afhangen van zowel de variantiefunctie, die de nauwkeurigheid van de metingen bepaalt, als de kostenfunctie, die de bijbehorende meetkosten beschrijft. Naarmate de orde van de variantiefunctie toeneemt, wordt de voorkeur gegeven aan grotere ribbelengtes, vanwege de nauwkeurigere schatting. Een probleem dat hierbij optrad is dat optimalisatie maar een beperkt aantal parameters kan optimaliseren zonder een suboptimaal resultaat te geven.

Verder zijn de ribbelengtes en bijbehorende gewichten bepaald die de zelfdiffusiviteit zo nauwkeurig mogelijk schatten. De gevonden optimale ribbelengtes zijn hetzelfde als bij de viscositeit. De gewichten zijn echter anders vanwege de uitdrukking van de variantiefunctie.

Tot slot zijn de viscositeit en zelfdiffusiviteit simultaan geschat met behulp van de covariantiematrix. Het toevoegen van de covariantie tussen de parameters gaf een 95% betrouwbaarheidsgebied dat zo'n 85% van de onzekerheid wegneemt, ten opzichte van een 95% betrouwbaarheidsgebied dat deze covariantie niet gebruikt.

Voor vervolgonderzoek kunnen verschillende richtingen worden verkend. Er zou bijvoorbeeld gebruik gemaakt kunnen worden van data, om de resultaten te vergelijken met experimentele waarden. Daarnaast zou een optimalisatiemethode die beter geschikt is voor het optimaliseren van grote hoeveelheden parameters kunnen worden uitgezocht, en er kunnen extra kosten/variantiefuncties worden overwogen. Tenslotte zou het introduceren van extra variabelen, zoals de temperatuur, in de vorm van multi-pele regressie, een mogelijkheid zijn voor vervolgonderzoek.

Referenties

- [1] John Fox. *Applied Regression Analysis and Generalized Linear Models*. SAGE Publications, 2015.
- [2] Seyed Hossein Jamali e.a. „Shear Viscosity Computed from the Finite-Size Effects of Self-Diffusivity in Equilibrium Molecular Dynamics”. In: *Journal of Chemical Theory and Computation* 14.11 (2018). PMID: 30296092, p. 5959–5968. DOI: 10.1021/acs.jctc.8b00625. eprint: <https://doi.org/10.1021/acs.jctc.8b00625>. URL: <https://doi.org/10.1021/acs.jctc.8b00625>.
- [3] W. H. Press e.a. „Modeling of Data”. In: *Numerical Recipes: The Art of Scientific Computing*. 3rd. Cambridge University Press, 2007, p. 773–839.
- [4] W. H. Press e.a. „Modeling of Data”. In: *Numerical Recipes: The Art of Scientific Computing*. 3rd. Cambridge University Press, 2007, p. 814–817.
- [5] Larry Wasserman. *Lecture 24: Statistical Methods*. Accessed: 2024-05-17. 2024. URL: <https://www.stat.cmu.edu/~larry/=stat401/lecture-24.pdf>.
- [6] In-Chul Yeh en Gerhard Hummer. „System-size dependence of diffusion coefficients and viscosities from molecular dynamics simulations with periodic boundary conditions”. In: *The Journal of Physical Chemistry B* 108.40 (2004), p. 15873–15879.

Appendix A1

In deze appendix laten we zien dat

$$\frac{n \sum X_i Y_i - \sum X_i \sum Y_i}{n \sum X_i^2 - (\sum X_i)^2} \stackrel{(*)}{=} \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sum (X_i - \bar{X})^2}$$

Alle sommeringen moeten van $i = 1$ tot n worden genomen. Er wordt doorgaans gebruikt dat $n\bar{X} = \sum X_i$. Eerst de noemer:

$$\begin{aligned} n \sum X_i^2 - (\sum X_i)^2 &= n \sum X_i^2 - 2(\sum X_i)^2 + (\sum X_i)^2 \\ &= n \sum X_i^2 - 2n\bar{X}(\sum X_i) + (n\bar{X})^2 \\ &= n(\sum X_i^2 - 2\bar{X} \sum X_i + n(\bar{X})^2) \\ &= n \sum [X_i^2 - 2\bar{X}X_i + \bar{X}^2] \\ &= n \sum (X_i - \bar{X})^2 \end{aligned}$$

Dan de teller:

$$\begin{aligned} n \sum X_i Y_i - \sum X_i \sum Y_i &= n \sum X_i Y_i - 2 \sum X_i \sum Y_i + \sum X_i \sum Y_i \\ &= n \sum X_i Y_i - \sum X_i \sum Y_i - \sum X_i \sum Y_i + n\bar{X}n\bar{Y} \\ &= n \sum (X_i Y_i) - n\bar{X} \sum Y_i - n\bar{Y} \sum X_i + n \sum \bar{X}\bar{Y} \\ &= n \sum (X_i Y_i - \bar{X}Y_i - X_i\bar{Y} + \bar{X}\bar{Y}) \\ &= n \sum (X_i - \bar{X})(Y_i - \bar{Y}) \end{aligned}$$

En dus

$$\frac{n \sum X_i Y_i - \sum X_i \sum Y_i}{n \sum X_i^2 - (\sum X_i)^2} = \frac{n \sum (X_i - \bar{X})(Y_i - \bar{Y})}{n \sum (X_i - \bar{X})^2} = \frac{\sum (X_i - \bar{X})(Y_i - \bar{Y})}{\sum (X_i - \bar{X})^2}$$

Appendix A2

In deze appendix laten we zien dat

$$\sum (X_i - \bar{X})X_i = \sum (X_i - \bar{X})^2$$

$$\begin{aligned}\sum (X_i - \bar{X})X_i &= \sum X_i^2 - \bar{X}X_i \\ &= \sum (X_i^2 - 2\bar{X}X_i + \bar{X}X_i) \\ &= \sum (X_i^2 - 2\bar{X}X_i) + \bar{X} \sum X_i \\ &= \sum (X_i^2 - 2\bar{X}X_i) + n\bar{X}^2 \\ &= \sum (X_i^2 - 2\bar{X}X_i + \bar{X}^2) \\ &= \sum (X_i - \bar{X})^2\end{aligned}$$

Appendix B

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.animation import FuncAnimation

N = 100
N1_vals = np.linspace(1, 100, 100)
N2_vals = np.linspace(1, 100, 100)
t_vals = np.linspace(0, 2*np.pi, 100)

N1, N2 = np.meshgrid(N1_vals, N2_vals)
N3 = N - N1 - N2

def sigma(x):
    return x**2

def var_B(x, weights):
    sd = sigma(x) / np.sqrt(weights)
    s = np.sum(1 / (sd ** 2))
    s_x = np.sum(x / (sd ** 2))
    s_xx = np.sum((x ** 2) / (sd ** 2))
    delta = s * s_xx - s_x**2
    return 10000 * s / delta

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
text = ax.text2D(0.05, 0.95, "", transform=ax.transAxes)
ax.view_init(elev=10, azim=100)

def update_plot(frame):
    x2 = 0.55 - 0.44 * np.sin(frame)
    pts = np.array([0.1, x2, 1])

    valid_mask = N3 >= 0
    variances = np.full(N1.shape, np.nan)
    weights = np.array([N1[valid_mask], N2[valid_mask], N3[valid_mask]]).T
    var_B_values = np.array([var_B(pts, weight) for weight in weights])

    variances[valid_mask] = var_B_values

    ax.clear()
    ax.plot_surface(N1, N2, np.log(variances), cmap='hsv')
    text.set_text(f"x = [{pts[0]:.2f}, {pts[1]:.2f}, {pts[2]:.2f}]")
```

```

ax.set_xlim([0, N])
ax.set_ylim([0, N])
ax.set_zlim([0, 10])
ax.set_xlabel('Gewicht $N_1$')
ax.set_ylabel('Gewicht $N_2$')
ax.set_zlabel('log(Variantie)')

ax.text2D(0.05, 0.95, f"pts = [{pts[0]:.2f}, {pts[1]:.2f}, {pts[2]:.2f}]",
transform=ax.transAxes)

return text

ani = FuncAnimation(fig, update_plot, frames=t_vals, interval=50)
# ani.save('animation.gif', writer='imagemagick')
plt.show()

```

Appendix C

```
import numpy as np

def sigma(x):
    return x**2

def var_B(x, weights):
    sd = sigma(x) / np.sqrt(weights)
    s = np.sum(1 / (sd ** 2))
    s_x = np.sum(x / (sd ** 2))
    s_xx = np.sum((x ** 2) / (sd ** 2))
    delta = s * s_xx - s_x**2
    return 10000 * s / delta

N = 50
pts = np.array((0.1, 0.5, 0.9)) # Gekozen punten, aanpasbaar
t_vals = np.linspace(0, 1, 26)

weight_vals, var_vals = [], []
for i in range(N-2):
    for j in range((N-2)-i):
        k = (N-2) - i - j
        weights = np.array([i+1, j+1, k])
        var = var_B(pts, weights)
        var_vals.append(var)
        weight_vals.append(weights)

for idx, w1 in enumerate(weight_vals):
    var_w1 = var_vals[idx]
    for w2_idx, w2 in enumerate(weight_vals[idx+1:], start=idx+1):
        var_w2 = var_vals[w2_idx]
        for t in t_vals:
            w_itp = t * w1 + (1-t) * w2
            var_itp = var_B(pts, w_itp)
            if var_itp > t * var_w1 + (1-t) * var_w2:
                print(f"NIET CONVEX bij gewichtigen: {w1} en {w2} voor t={t}")
                break

# Als er nergens wordt geprint dat de functie niet convex is,
# betekent dit dat de functie convex is.
```

Appendix D

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation

N = 100
E = 400
N1_vals = np.linspace(1, N-2, N-2)
N2_vals = np.linspace(1, N-2, N-2)
t_vals = np.linspace(0, 2*np.pi, 100)

N1, N2 = np.meshgrid(N1_vals, N2_vals)
N3 = N - N1 - N2

def sigma(x):
    return x**2

def cost_constraint(x, weights, E):
    def cost_fun(x):
        return 1/x

    left_side = np.sum(np.array([weights[_[0]]*cost_fun(_[1]) for _ in enumerate(x)]),
        axis=0)
    return left_side <= E

def var_B(x, weights):
    sd = sigma(x) / np.sqrt(weights)
    s = np.sum(1 / (sd ** 2))
    s_x = np.sum(x / (sd ** 2))
    s_xx = np.sum((x ** 2) / (sd ** 2))
    delta = s * s_xx - s_x**2
    return 10000 * s / delta

fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111, projection='3d')
text = ax.text2D(0.05, 0.95, "", transform=ax.transAxes)
ax.view_init(elev=10, azim=100)

def update_plot(frame):
    x2 = 0.55 - 0.4 * np.sin(frame)
    pts = np.array([0.1, x2, 1])
    weights = np.array([N1, N2, N3])

    inequality = cost_constraint(pts, weights, E)
    valid_weights = inequality & (N1 >= 1) & (N2 >= 1) & (N3 >= 1)

    variances = np.full(N1.shape, np.nan)
```

```

weights = np.array([N1[valid_weights], N2[valid_weights], N3[valid_weights]]).T
var_B_values = np.array([var_B(pts, weight) for weight in weights])
variances[valid_weights] = var_B_values

min_index = np.nanargmin(variances)
min_coords = np.unravel_index(min_index, variances.shape)
N1_min = N1[min_coords]
N2_min = N2[min_coords]
min_var = variances[min_coords]

ax.clear()
ax.plot_surface(N1, N2, np.log(variances), cmap='hsv')
ax.scatter(N1_min, N2_min, np.log(min_var), marker = (5, 2, 0), color='black',
s=100, label='Minimum')

ax.text2D(0.05, 0.95, f"Mininum bij $(N_1, N_2, N_3)$ = ({N1_min:.0f},{N2_min:.0f},
{100-N1_min- N2_min:.0f}),
Variantie = {min_var:.2f} \n pts = [{pts[0]:.2f}, {pts[1]:.2f}, {pts[2]:.2f}]",
transform=ax.transAxes)

ax.set_xlim([0, N])
ax.set_ylim([0, N])
ax.set_zlim([0, 10])
ax.set_xlabel('Gewicht $N_1$')
ax.set_ylabel('Gewicht $N_2$')
ax.set_zlabel('log(Variantie)')
plt.legend()

plt.show()
return text

ani = FuncAnimation(fig, update_plot, frames=t_vals, interval=50)
ani.save('CombinedAnimation.gif', writer='imagemagick')
plt.show()

```

Appendix E

```
import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt

N = 100
E = 10000
pts = np.array([0.5, 1])
n = len(pts)

def var_B(x, weights):
    sd = sigma(x) / np.sqrt(weights)
    s = np.sum(1 / (sd ** 2))
    s_x = np.sum(x / (sd ** 2))
    s_xx = np.sum((x ** 2) / (sd ** 2))
    delta = s * s_xx - s_x**2
    return 1e4 * s / delta

def sigma(x):
    return x**2

def cost_constr(params):
    def cost_fun(params):
        return 1/params

    left_side = np.sum(np.array([params[_[0]]*cost_fun(_[1]) for _ in enumerate(pts)]),
        axis=0)
    return E - left_side

def weights_constr(params):
    return params.sum() - N

def objective_fun(params):
    return var_B(pts, params)

weights_constraint = {'type': 'eq', 'fun': weights_constr}
cost_constraint = {'type': 'ineq', 'fun': cost_constr}

constraints = [weights_constraint,
               cost_constraint,
               ]
```



```
init_guess = np.array([50, 25])

result = minimize(objective_fun,
                  init_guess,
                  constraints=constraints,
                  bounds=[(1, N)],
                  options={'disp': True})

print(result.x)
```

Appendix F

```
import numpy as np
from scipy.optimize import minimize
import matplotlib.pyplot as plt

N = 100
pts = np.linspace(0.1, 1, 9)
n = len(pts)

def var_B(x, weights):
    sd = sigma(x) / np.sqrt(weights)
    s = np.sum(1 / (sd ** 2))
    s_x = np.sum(x / (sd ** 2))
    s_xx = np.sum((x ** 2) / (sd ** 2))
    delta = s * s_xx - s_x**2
    return 1e4 * s / delta

def sigma(x):
    return x**1.5

def cost_constr(params):
    def cost_fun(params):
        return 1/params**2
    left_side = np.sum(np.array([params[_[0]]*cost_fun(_[1]) for _ in enumerate(pts)]),
        axis=0)
    return E - left_side

def weights_constr(params):
    return params.sum() - N

def objective_fun(params):
    return var_B(pts, params)

weights_constraint = {'type': 'eq', 'fun': weights_constr}
cost_constraint = {'type': 'ineq', 'fun': cost_constr}

constraints = [weights_constraint,
               cost_constraint]

init_guess = np.array([50]*n)
E_values = np.array([110, 125, 150, 175, 200, 250, 300, 400, 800, 1600])
plt.figure(figsize=(8, 6))

for i, E in enumerate(E_values):
    result = minimize(objective_fun,
                      init_guess,
                      constraints=constraints,
```

```

        bounds=[(0.1, N)],
        options={'disp': False})

    if result.success:
        weights = result.x
        filtered_pts = pts[weights > 0.4]
        filtered_weights = weights[weights > 0.4]

        y_vals = [i] * len(filtered_pts)

        plt.scatter(filtered_pts, y_vals, c=filtered_weights, cmap='coolwarm',
                    s=50, label=f'E={E}')

        print(f"E = {E}, Variance: {result.fun:.2f}")
        print("Weights:", [f'{weight:.2f}' for weight in weights], int(result.fun))
    else:
        print(f"E = {E} is te klein")

plt.xlim(0.05,1.05)

plt.title("Geoptimaliseerde gewichten tegen meetkosten $E$")
plt.xlabel("$x_i$")
plt.ylabel("E")

plt.yticks(range(len(E_values)), [f'E={e}' for e in E_values])
plt.colorbar(label='Aantal metingen')

plt.show()

```

Appendix G

```
library(reticulate)
library(car)
source_python("WeightOptimisationGrafieken.py")
set.seed(0)

x <- py$filtered_pts
weights <- py$filtered_weights
SE = sigma(x)/sqrt(weights)

y = c()
for (i in 1:length(x)){
  y <- append(y, rnorm(1, sd = SE[i]))
}

lm_fit <- lm(y ~ x, weights = 1 / SE^2)

plot(y ~ x, xlab = "x", ylab = "y")
abline(lm_fit, col = "blue")

plot(0, 0, type = "n", xlim = c(-0.05, 0.05), ylim = c(-0.4, 0.4),
      xlab = "Intercept", ylab = "Slope",
      main = "Confidence-Ellipse voor Intercept-A en Helling-B, 95%",
      xaxt = 'n', yaxt = 'n')

axis(1, at = seq(-1, 1, by = 0.02))
axis(2, at = seq(-1, 1, by = 0.2), las = 1)

confidenceEllipse(lm_fit, which.coef = c("(Intercept)", "x"),
                  vcov. = vcov(lm_fit), levels = 0.95, col = "blue",
                  add = TRUE)

grid()
points(0, 0, pch = 19, col = "red")

# Vierkant confidence interval maken
ci = confint(lm_fit, level = 0.975)
rect(ci[1,1], ci[2,1], ci[1,2], ci[2,2])

# Vectoren confidence ellipse opstellen
ev = eigen(vcov(lm_fit))
chi2_val = qchisq(0.95, df = 1)
scaling_factors = 10*sqrt(ev$values* chi2_val)

#Pijlen
arrows(lm_fit$coef[1], lm_fit$coef[2],
       lm_fit$coef[1] + ev$vectors[1,1] * scaling_factors[1],
```

```

lm_fit$coef[2] + ev$vectors[2,1] * scaling_factors[1],
col = "green", lwd = 2, length = 0.1)

arrows(lm_fit$coef[1], lm_fit$coef[2],
lm_fit$coef[1] + ev$vectors[1,2] * scaling_factors[2],
lm_fit$coef[2] + ev$vectors[2,2] * scaling_factors[2],
col = "green", lwd = 2, length = 0.1)

# opp berekenen
opp_ellips = scaling_factors[1] * scaling_factors[2]
opp_bonferroni = (ci[2,2] - ci[2,1]) * (ci[1,2] - ci[1,1])

opp_ellips/opp_bonferroni

```