CHALMERS

GÖTEBORGS UNIVERSITET

Matematiska modeller och datorberäkningar för ytdiffusion

Mathematical models and numerical calculations for surface diffusion

Examensarbete för kandidatexamen i matematik vid Göteborgs universitet

Jessica El-Jabaoui
Felicia Hägerström
Christian Jenei

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2023

Matematiska modeller och datorberäkningar för ytdiffusion

Examensarbete för kandidatexamen i matematik vid Göteborgs universitet

Jessica El-Jabaoui Felicia Hägerström Christian Jenei

Handledare: Tobias Gebäck

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2023

Förord

Först och främst tackar vi vår handledare Tobias Gebäck för hans stöd och hans insats. Vi vill även tacka våra opponenter och examinatorerna Ulla Dinger och Maria Roginskaya. Vi vill även tacka alla som bidragit med återkoppling under fackspråksmöten och bibliotektstillfällen.

Under arbetets gång har en loggbok förts över de veckovisa framsteg som gruppen har gjort och antal timmar projektet har tagit. Projektet har gjorts i grupp om tre personer, i denna bidragsrapport kommer det framgå hur gruppen har arbetat tillsammans för att genomföra arbetet. Viktigt att tilläga är att gruppen jobbat nära och haft kontinuerlig kontakt och alla beslut om innehåll har gemensamt diskuterats under processen.

Kapitel	Avsnittsrubrik	Huvudförfattare
	Populärvetenskaplig presentation	Jessica
	Sammandrag och Abstract	Felicia
1	Inledning	Felicia
1.1	Bakgrund	Felicia
1.2	Syfte	Jessica, Felicia
1.3	Problemformulering	Jessica, Felicia
1.4	Avgränsningar	Jessica, Felicia
2	Samhälleliga och etiska aspekter	Jessica
3	Teori	
3.1	Differentialgeometrin hos kurvor och ytor	
3.1.1	Parametrisering av kurvor	Christian
3.1.2	Båglängdsparametrisering	Felicia
3.1.3	Frenet-Serrets formler	Christian
3.2	Geometriska flöden	Jessica, Felicia
3.2.1	Krökningsflöde	Jessica
3.2.2	Ytdiffusion	Jessica
3.3	Utvecklingsmetoder för hyperytor	
3.3.1	Nivåmängdsmetoden	Christian
3.3.2	Fasfälts metoden	Felicia, Christian
3.3.3	Cahn-talet	Jessica
3.3.4	Diffusiv skalning	Felicia
4	Metod	Felicia
5	Resultat	
5.1	Beräkningar för geometriska flöden	Felicia
5.2	Numeriska beräkningar	Jessica, Christian
6	Diskussion	Jessica, Felicia
A	Appendix 1 – Teori	Felicia, Christian
B	Appendix 2 – Numerisk	Christian
C	Appendix 3 – Felkällor	Jessica
D	Appendix 3 – Källkod	Christian

Populärvetenskaplig presentation

Vad händer när vätskor tränger in i torra material? Det är en vanligt förekommande frågeställning i tillverkningsindustrin för att kunna producera bättre produkter, till exempel tabletter som tas upp snabbare av kroppen eller papperssugrör som håller längre. Med matematikens hjälp kan detta fenomen studeras teoretiskt och ge en djupare förståelse för hur vätskor flödar över ytor.

Inom matematiken beskriver geometriska flöden dynamiken hos geometriska ytor, exempel på ett geometriskt flöde är hur gränsen mellan luft och vatten beter sig. Ett exempel på geometriskt flöde är ytdiffusion, som kan beskrivas som en process där molekyler rör sig med en hastighet som beror på ytans geometri. Sådana typer av geometriska flöden har en kort historia inom matematiken och har blivit ett aktuellt forskningsområde de senaste 30 åren med hjälp av ny teknik och högprestandaberäkning [1].

Ytdiffusion beskrivs matematiskt som en differentialekvation, vilket är en ekvation som relaterar en eller flera okända funktioner med hur dessa funktioner ändras över tid. Eftersom funktionerna är okända, krävs ofta datorberäkningar för att lösa ekvationen. En differentialekvation som används för att studera ytdiffusioner är den modifierade Allen Cahn-ekvationen. Ordningen av differentialekvationerna är skillnaden mellan den modifierade Allen Cahn-ekvationen och den konventionella differentialekvationen för ytdiffusion. Den modifierade Allen Cahn-ekvationen är av andra ordningen, vilket är två ordningar lägre och som gör ekvationen mindre beräkningstung för en dator att lösa. Om den modifierade Allen Cahn-modellen är en god approximation av ytdiffusion, skulle den vara ett användbart verktyg för att studera ytdiffusion.

I arbetet undersöktes huruvida den modifierade Allen Cahn-modellen är en god approximation av ytdiffusion. Det finns nämligen inget bevis för att den modifierade Allen Cahn-ekvationen beter sig som en ytdiffusion. För att den modifierade Allen Cahn-ekvationen ska anses beskriva ytdiffusion, ska volymen av vätskan bevaras när den flödar över ett material. För att sedan undersöka sambandet utförde vi både analytiska och numeriska beräkningar för ytdiffusion.

Analytiskt studerades enkla geometriska former: cirklar och ellipser. Enligt våra studier bevaras både cirkelns och ellipsens area för ytdiffusion. Cirkeln är stationär under ytdiffusion och ändrar inte form medan ellipsens form växer långsamt på trubbiga sidorna och krymper snabbt på de spetsiga delarna.

För numeriska beräkningar användes ett datorprogram med en beräkningsmodell där materialstrukturer utformas för att få en kontrollerad diffusion. Med hjälp av datorprogrammet simulerades geometriska flöden för den modifierade Allen Cahn-ekvationen över ytor och volymförändring. Efter vissa val av parametrar och begynnelsevärden kan vi visa att de numeriska beräkningarna antyder att den modifierade Allen-Cahn beskriver ytdiffusion väl.

Detta är ett område som ännu inte är väl utforskat och vi har endast visat att numeriska simuleringar tyder på ett samband mellan ytdiffusion och den modifierade Allen Cahn-modellen. Ett analytiskt bevis att den modifierade Allen Cahn-ekvation beter sig som en fjärde ordningens differentialekvation skulle ytterligare visa på modellens användbarhet för att underlätta beräkningar för ytdiffusion.

Sammandrag

I detta arbete undersöks geometriska flöden och hur dessa kan implementeras numeriskt med fasfältsmodeller. Speciellt ligger fokus på ytdiffusion, som är ett geometriskt flöde där ändliga ytor i andra dimensionen konvergerar mot en cirkel och arean bevaras. Beräkning av ytdiffusion kräver lösning av fjärde ordningens differentialekvation, och när detta simuleras numeriskt krävs hög upplösning av initialdata, då derivator av hög ordning annars inte blir precisa. Datorprogram som har hög upplösning av initialdata samt gör många uträkningar under flera iterationer är datorkrävande, och det är därför intressant att hitta en enklare ekvation som numeriskt kan beskriva ytdiffusion, därför undersöks en modifierad Allen-Cahn ekvation i arbetet. Simuleringar med den modifierade Allen-Cahn ekvationen görs i ett datorprogram som implementerar Lattice-Boltzmann metoden, och dessa jämförs med teori samt analytiska beräkningar av ytdiffusion. Det visar sig att modifierade Allen-Cahn ekvationen beskriver ytdiffusion väl vid val av parametrar och initialdata då arean bevarades samt deformeringen skedde i enlighet med teorin.

Nyckelord: Ytdiffusion, Krökningsflöde, Modifierade Allen-Cahn ekvationen, Geometriska flöden, Fasfält.

Abstract

This study investigates geometric flow and how it can be implemented as phase field models. Surface diffusion is the main geometric flow of this study. Two dimensional finite surfaces under surface diffusion will converge to a circle and have no change in area. Calculations on surface diffusion are done by solving a differential equation of the fourth degree, and when implemented numerically the input-data needs to be of high resolution for the derivatives to be accurate. Computer programs with such requirements are computer demanding, thus it is of interest to find a more simple equation to use as substitute for the surface diffusion calculations. A modified Allen-Cahn equation is evaluated for this purpose in this study. Simulations of surfaces deforming according to the modified Allen-Cahn equation are made in a computer program implementing the Lattice-Boltzmann method. These simulations are then compared to theory and analytical calculations of surface diffusion, and it is shown that the modified Allen-Cahn equation is a suitable equation when simulating surface diffusion if parameters and input-data are chosen carefully.

Keywords: Surface diffusion, Mean curvature flow, Modified Allen-Cahn equation, Geometrical flow, Phase field.

Innehållsförteckning

1 Inledning

Under 1980-talet började fasfältsmetoden användas i datorprogram för att lösa problem relaterade till krökning av randen till kroppar eller ytor. Fasfältsmetoden används som ett verktyg för simulering av materials fasövergångar under geometriska flöden, exempel på ett geometriskt flöde som kan tillämpas med fasfältsmetoden är ytdiffusion. Trots att området är utforskat och välkänt är så kräver numeriska tillämpningar av metoden mycket datorkraft. Datorn utför datorkrävande beräkningar vid simulering och därav har numerisk tillämpning fortfarande utvecklingsmöjligheter [1].

1.1 Bakgrund

Forskningsgruppen CoSiMa på Matematiska vetenskaper vid Chalmers tekniska högskola och Göteborgs universitet har ett pågående projekt. De skapar ett digitalt verktyg för materialutveckling för svenska företag. Med verktyget ska företag som förpackar vätskor och lösta ämnen i material kunna designa materialstruktur och simulera dess funktion samt masstransport [2].

Tidigare har CoSiMa arbetat med projekt som behandlar materialstruktur och molekylär transport, exempelvis frisättning av läkemedel eller täta livsmedelsförpackningar. De har skapat verktyg för att ge företag bättre kunskap om material de använder för sina produkter [3].

I kandidatarbetet undersöks en fasfältstillämpning av geometriska flöden, speciellt ytdiffusion, vilket i två dimensioner matematiskt beskriver hur ytor ändrar form. Om området sedan vidareutvecklas för tillämpning i tre dimensioner kan vi simulera hur kroppar deformeras under ytdiffusion. Dessa simuleringar kan därför vara användbara för produkten som CoSiMa utvecklar i sitt pågående projekt.

1.2 Syfte

I detta arbete undersöks sambandet mellan ytdiffusion och den modifierade Allen-Cahn ekvationen, som är en fasfältsekvation. Anledningen till detta är att ytdiffusion kräver lösning av fjärde ordningens differentialekvation, och när detta simuleras numeriskt krävs hög upplösning av initialdata, då derivator av hög ordning annars inte blir precisa. Datorprogram som har hög upplösning av initialdata samt gör många uträkningar under flera iterationer är datorkrävande, och det är därför intressant att hitta en enklare ekvation som numeriskt kan beskriva ytdiffusion, därför undersöks en modifierad Allen-Cahn ekvation i arbetet.

1.3 Problemformulering

Arbetet går ut på att ta reda på hur olika parametrar påverkar fasfältsmodellen, och utreda om den modifierade Allen-Cahn ekvationen implementerar ytdiffusion.

Differentialgeometri hos kurvor och ytor, geometriska flöden som ytdiffusion, och utvecklingsmetoder för ytor studeras. Sedan undersöks det hur enkla former deformeras, samt om arean av dessa former bevaras. Slutligen utförs numeriska beräkningar där olika parametrar, former och upplösning av initialdata varieras.

1.4 Avgränsningar

Arbetet avgränsas genom att endast beräkningar och numeriska analyser sker i två dimensioner. Programmeringsverktyget som används vid de numeriska beräkningarna är skapad av CoSiMa för tre dimensioner, men är modifierat för två dimensioner. En ytterligare avgränsning som görs är att Lattice Boltzmann metoden, som används i beräkningsprogrammet Gesualdo, inte redovisas.

2 Samhälleliga och etiska aspekter

Ett antal etiska och samhälleliga aspekter beaktas i samband med projektet. Frågeställningen som besvaras i arbetet är teoretisk och har ingen koppling till något levande men på sikt kan arbetets idéer tillämpas i tre dimensioner för simuleringar inom materialutveckling i verkligt bruk.

Positiva följder av ämnesområdet som kandidatarbetet behandlar finns bland annat i läkemedelsindustrin men även i olika typer av förpackningar. I läkemedelsindustrin kan matematiken användas för att ta reda på hur snabbt medicin frigörs i kroppen. Matematiken skapar möjlighet för att dosera korrekt mängd läkemedel, som gör behandlingar för olika sjukdomar mer effektiva. Förutom de positiva aspekter inom medicin, kan liknande metod användas för att ta reda på hur olika typer av material reagerar i kontakt med vätska. Detta är särskilt viktigt inom matindustrin vid förpackning av livsmedel, till exempel mjölkpaket och papperssugrör [4].

De negativa etiska aspekterna är inte direkt kopplade till matematiska formler och beräkningar, utan till företag som använder forskningen och hur företagen tillämpar detta. Matematiken kan eventuellt användas för att veta hur snabbt medicin frigörs i kroppen, på samma sätt som det kan användas för att reglera korrekt mängd läkemedel i kroppen, kan det användas för att studera frisättning av toxiska substanser. All matematik kan eventuellt användas till något icke etiskt i fel händer. Däremot har vår undersökning inte någon direkt koppling till etiska aspekter eller negativ samhällelig påverkan.

3 Teori

Detta kapitel syftar till att ge den teoretiska bakgrund som krävs för att förstå metoden och resultatet. Kapitlet kommer behandla teori om differentialgeometri för kurvor och ytor, geometriska flöden och utvecklingsmetoder för ytor.

3.1 Differentialgeometrin hos kurvor och ytor

Kurvor kan definieras som oändligt deriverbara funktioner som avbildar ett intervall på en delmängd av ett rum, som exempelvis $\mathbb{R}^n$. I avsnittet behandlas $\mathbb{R}^3$, men sektionen om geometriska flöden behandlar $\mathbb{R}^2$, där samma begrepp är tillämpbara.

En kurva i $\mathbb{R}^3$ definieras som $f : I \to M \subset \mathbb{R}^3$. Det finns olika metoder för att definiera kurvor, till exempel genom parametriska ekvationer, implicita ekvationer och differentialekvationer.

3.1.1 Parametrisering av kurvor

Definition 3.1 (Parametrisering av kurvor)**.** *Betrakta kurvan*

$$\gamma(t) := (x(t), y(t), z(t)), \quad t \in I, \tag{3.1}$$

där γ är en kurva i $\mathbb{R}^3$. Kurvans parameter $t \in I$, $I = (a, b)$ där $a, b \in \mathbb{R}$.

Ett exempel på en kurvparametrisering finns i appendix A (se A.1).

Definition 3.2 (Tangentvektorn)**.** *Tangentvektorn $\boldsymbol{T}$ till en kurva $\gamma(t)$ i rummet är definierad som den första derivatan av $\gamma(t)$ med avseende på t, normaliserad till längden. Tangentvektorn visar riktningen i en punkt på kurvan vid tidpunkten t. Enhetstangentvektorn uttryckas som*

$$\mathbf{T} = \frac{\gamma'(t)}{|\gamma'(t)|}, \tag{3.2}$$

där $|\cdot|$ betecknar längden av en vektor [5].

En punkt t på kurvan γ där $\gamma'(t) = 0$ kallas en singularitet [6]. Det betyder att kurvan vid den punkten inte kan beskrivas som en slät kurva, eftersom det inte existerar en definierad riktning för kurvan vid punkten. Singulariteter kan uppstå om kurvan bildar en spets. En kurva som ej har några singulariteter benämns som reguljär.

Definition 3.3 (Reguljär kurva [6])**.** *En kurva $\gamma : I \to \mathbb{R}^3$ är reguljär om*

$$\gamma'(t) \neq 0, \quad \forall t \in I. \tag{3.3}$$

Definition 3.4 (Sluten kurva)**.** *En sluten kurva är en kurva sådan att om man rör sig längs kurvan återvänder man till startpunkten. Alltså är $\gamma : I \to \mathbb{R}^3$ sluten om:*

$$I = [a,b] \text{ och } \gamma(a) = \gamma(b). \tag{3.4}$$

Definition 3.5 (Periodisk kurva)**.** *Periodisk kurva är en kontinuerlig funktion $\gamma : \mathbb{R} \to \mathbb{R}^n$ sådan att det finns en positiv konstant T kallad kurvans period, så att $\gamma(t+T) = \gamma(t)$, $\forall t \in \mathbb{R}$. Detta innebär att kurvan har ett repeterande mönster, och dess värden vid två punkter som är åtskilda med en period T är samma [5].*

3.1.2 Båglängdsparametrisering

Definition 3.6 (Båglängdsparametrisering)**.** *Båglängdsparameteriseringen av en kurva*

$$\gamma = \gamma(t), \ t \in [\alpha, \beta]$$

är $\tilde{\gamma}(s)$ om

$$s(t) = \int_\alpha^t |\gamma'(\tau)| d\tau \tag{3.5}$$

och

$$\tilde{\gamma}(s(t)) = \gamma(t), \quad \forall t \in [\alpha, \beta] \tag{3.6}$$

där s är kurvans längd mellan α och t [5].

Båglängdsparametrisering används vid krökning-beräkningar då det resulterar i att tangentvektorn till kurvan alltid är en enhetsvektor, eftersom omskrivning av (3.5) enligt analysens fundamentalsats ger

$$\frac{\partial s}{\partial t} = |\gamma'(t)|$$

som medför

$$\frac{\partial t}{\partial s} = \frac{1}{|\gamma'(t)|}. \tag{3.7}$$

Kedjeregeln vid derivering av en kurva efter parameterbyte ger

$$\frac{\partial \gamma}{\partial s} = \frac{\partial \gamma}{\partial t} \cdot \frac{\partial t}{\partial s}$$

och därmed

$$\frac{\partial \gamma}{\partial s} = \frac{\gamma'(t)}{|\gamma'(t)|},$$

vilket visar att tangenten är av längd ett. Därför inför vi beteckningen $\mathbf{T}(s) = \frac{\partial \gamma}{\partial s}$ för enhetstangenten [5].

3.1.3 Frenet–Serrets formler

Innan Frenet–Serrets formler presenteras så inleds sektionen med grundläggande teori om krökning och torsion, samt en härledning av vektorbasen i formlerna.

Definition 3.7 (Krökning)**.** *Låt $\gamma : I \to \mathbb{R}^3$ vara en kurva parametriserad av båglängden $s \in I$. Då definieras krökningen hos kurvan γ i punkten s som $|\gamma''(s)|$ och betecknas som $\kappa(s)$* [6].

Några observationer som kan göras med hänsyn till definitionen (3.7) är att om $\kappa = |\gamma''(s)| = 0$ för alla $t \in I$, så är kurvan $\gamma(s)$ en rak linje, eftersom kurvan inte har någon krökning [6]. Vid punkter s på kurvan $\gamma(s)$ där $\kappa(s) \neq 0$ är normalvektorn väldefinierad. Då kan följande definition göras $\mathbf{T}'(s) = \gamma''(s) = \kappa(s)\mathbf{n}(s)$, där $\mathbf{n}(s)$ är enhetsnomalvektorn i punkten s. För att visa att $\mathbf{T}'(s) \perp \mathbf{T}(s)$ låt,

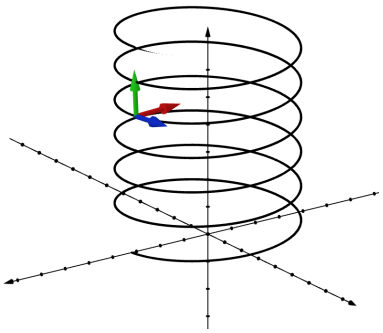
$$\mathbf{T}(s) = (x(s), y(s), z(s)).$$

Givet att $|\mathbf{T}(s)|^2 = 1$, har vi $x(s)^2 + y(s)^2 + z(s)^2 = 1$. Vid derivering med avsende på s fås följande enligt kedjeregeln,

$$\frac{\partial}{\partial s}|\mathbf{T}(s)|^2 = 2x(s)x'(s) + 2y(s)y'(s) + 2z(s)z'(s) = 0.$$

Det betyder att $\frac{\partial}{\partial s}|\mathbf{T}(s)|^2 = 2(\mathbf{T}(s) \cdot \mathbf{T}'(s)) \Rightarrow \mathbf{T}(s) \cdot \mathbf{T}'(s) = 0$ Således leder deriveringen av $|\mathbf{T}(s)|^2 = 1$ med avseende på s till att $\mathbf{T}(s)$ är ortogonal mot $\mathbf{T}'(s)$.

Vektorn $\mathbf{n}(s)$ kan skrivas som $\mathbf{n}(s) = \frac{\gamma''(s)}{|\gamma''(s)|} = \frac{\mathbf{T'(s)}}{\kappa(s)}$ och är alltid riktad mot till centern av kurvan (se figur 3.1). För att få sista basvektorn till basen innehållande $\mathbf{T}(s)$ och $\mathbf{n}(s)$ introduceras bi-normalsvektorn $\mathbf{B}(s) = \mathbf{T}(s) \times \mathbf{n}(s)$ [6]. Dessa tre vektorer bildar ett högerhänt koordinatsystem med tre axlar och kallas för Frenet-ramen. Den används för att beskriva orienteringen och rörelsen hos en kurva i rummet. Se exempel (A.1) samt figur 3.1 för visualisering av Frenet-ramen i rummet.



Figur 3.1: En tredimensionell kurva som bildar en spiralform, beskriven av parametriseringen (A.1). Samt dess vektorer från Frenet-ramen, tangent (Blå vektor), normal (Röd vektor), och bi-normal (Grön vektor).

Frenet–Serrets formlerna beskriver derivatorna hos $\mathbf{T}(s), \mathbf{n}(s)$ och $\mathbf{B}(s)$. Den första derivatan $\mathbf{T}'(s)$ är given från definitionen då,

$$\frac{d\mathbf{T}(s)}{ds} = \gamma''(s) = |\gamma''(s)|\frac{\gamma''(s)}{|\gamma''(s)|} = \kappa(s)\mathbf{n}(s).$$

Längden av derivatan till $\mathbf{B}(s)$ beskriver förändringshastigheten av lutningen av planet som går genom vektorerna $\mathbf{T}$(s) och $\mathbf{n}$(s) [6]. Denna egenskap kallas för torsionen hos en kurva. Vidare kan det observeras att $\mathbf{B}'(s)$ är vinkelrät med $\mathbf{B}(s)$ enligt samma härledning som för $\mathbf{T}'(s) \perp \mathbf{T}(s)$, samt att $\mathbf{B}(s)\cdot\mathbf{T}(s) = 0 \Rightarrow \mathbf{B}'(s)\cdot\mathbf{T}(s)+\mathbf{B}(s)\cdot\mathbf{T}'(s) = 0$ där $\mathbf{B}(s)\cdot\mathbf{T}'(s) = 0$ eftersom $\mathbf{T}'(s) = \kappa(s)\mathbf{n}(s)$ alltså att $\mathbf{B}(s) \perp a\mathbf{n}(s)$, $\forall a \in \mathbb{R}$. Således kan det observeras att $\mathbf{B}'(s) \perp \mathbf{B}(s)$ samt $\mathbf{B}'(s) \perp \mathbf{T}(s)$ vilket leder till slutsatsen att $\mathbf{B}'(s)$ är parallel med $\mathbf{n}(s)$ och är därmed en multipel av $\mathbf{n}(s)$.

Definition 3.8 (Torsionen)**.** *Låt $\gamma : I \to \mathbb{R}^3$ vara en kurva parametriserad av båglängden $s \in I$. Torsionen defineras som $\boldsymbol{B}'(s) = \tau(s)\boldsymbol{n}(s)$ och betecknas $\tau(s)$ [6].*

Vidare beräknas derivatan till normalen $\mathbf{n}$ som följande,

$$\begin{aligned}\mathbf{n}(s) =& \mathbf{B}(s)\times\mathbf{T}(s)\\ \Rightarrow \mathbf{n}'(s) =& \mathbf{B}'(s)\times\mathbf{T}(s) + \mathbf{B}(s)\times\mathbf{T}'(s)\\ =& \tau(s)\mathbf{n}(s)\times\mathbf{T}(s) + \mathbf{B}(s)\times\kappa\mathbf{n}(s)\\ =& \tau\mathbf{B}(s) - \mathbf{T}(s)\kappa.\end{aligned}$$

Frenet–Serrets formler definieras som följande,

$$\begin{cases}\mathbf{T}'(s) = \kappa(s)\mathbf{n}(s)\\ \mathbf{n}'(s) = -\kappa(s)\mathbf{T}(s) + \tau(s)\mathbf{B}(s)\\ \mathbf{B}'(s) = -\tau(s)\mathbf{n}(s)\end{cases}. \tag{3.8}$$

I resterande delar studeras endast planet, då är $\tau(s) = 0, \quad \forall s \in I$. Frenet–Serrets formler för planet blir då följande,

$$\begin{cases}\mathbf{T}'(s) = \kappa(s)\mathbf{n}(s)\\ \mathbf{n}'(s) = -\kappa(s)\mathbf{T}(s)\end{cases}. \tag{3.9}$$

För krökningen i två dimensioner så tar vi hänsyn till tecken beroende på orientering.

3.2 Geometriska flöden

Geometriska flöden beskriver en familj av kurvor, med andra ord en grupp av kurvor som delar en eller flera gemensamma egenskaper.

Definition 3.9 (Geometriskt flöde [1])**.** *Givet en familj av parametriserade kurvor $\gamma(p,t)$, $t \in [0,T)$, $p \in I$ definieras det geometriska flödet som*

$$\mathbf{v}(\gamma(p,t),t) = \frac{\partial \gamma}{\partial t} \quad . \tag{3.10}$$

Det geometriska flödet kan beskrivas med normalhastigheten

$$\mathbf{v_n}(\gamma(p,t),t) = \mathbf{v}(\gamma(p,t),t) \cdot \mathbf{n}(\gamma(p,t)), \tag{3.11}$$

där $\mathbf{n} \in \mathbb{R}^n$ är den inåtriktade normalvektorn för kurvan [1].

Exempel på geometriska flöden är krökningsflöde (*Curvature flow*), ytdiffusionsflöde, Willmore-flöde, Hele-Shaw-flöde och Stefan-problem [1]. Både ytdiffusion samt krökningflöde är en process där ytan på ett objekt utjämnas över tid.

3.2.1 Krökningsflöde

Definition 3.10 (Krökningsflöde (Curvature flow))**.** *Krökningsflöde är definierat som*

$$\frac{\partial \gamma}{\partial t} = \kappa \boldsymbol{n} \tag{3.12}$$

där $\boldsymbol{n}$ definieras som en inåt riktad normalvektor [7].

Ytor under krökningsflöde kommer att krympa till en punkt, exempelvis är tidsderivatan av arean A, för en cirkel med radie R och med nivåkurva $\gamma(s),\ s \in [0, 2\pi)$ och $\kappa(s) = \frac{1}{R}$ följande,

$$\frac{\partial A}{\partial t} = -\int_0^{2\pi} \mathbf{v} \cdot \mathbf{n} ds = -\int_0^{2\pi} \kappa(s)\mathbf{n} ds = -\int_0^{2\pi} \frac{1}{R} ds = -2\pi. \tag{3.13}$$

Eftersom normalen $\mathbf{n}$ är inåtriktad krymper ytan, alltså ger den upphov till minustecknet i uträkningen för tidsderivatan [8].

3.2.2 Ytdiffusion

Definition 3.11 (Ytdiffusion)**.** *Ytdiffusion definieras*

$$\frac{d\gamma}{dt} = -\Delta_\gamma \kappa(t) \boldsymbol{n}. \tag{3.14}$$

där Δ_γ är Laplace-operatorn på kurvan[1].

Ytdiffusion kräver att normalhastigheten för kurvan är lika med Laplace av krökningen vid varje punkt på ytan för tiden $t > 0$. Arean A innanför en sluten kurva $\gamma(s),\ s \in [a,b)$ under ytdiffusion är konstant då tidsderivatan av arean är

$$\frac{\partial A}{\partial t} = \int_a^b -\mathbf{v} \cdot \mathbf{n} ds = -\int_a^b \Delta_\gamma \kappa(s) ds = -\int_a^b \frac{\partial^2}{\partial s^2} \kappa(s) ds = -[\kappa'(s)]_a^b = \kappa'(a) - \kappa'(b) = 0, \tag{3.15}$$

eftersom kurvan är sluten är $\gamma(a) = \gamma(b) \rightarrow \kappa'(a) = \kappa'(b)$.

3.3 Utvecklingsmetoder för hyperytor

Hyperytan Γ representeras i det euklidiskt rummet $\mathbb{R}^n$. Utvecklingen av en hyperyta refererar till en hyperyta som styrs av någon geometrisk lag $\mathbf{v}$ som krökningsflöde (3.10), ytdiffusion (3.11) eller andra. För fallet $\mathbb{R}^2$ representeras Γ som en kurva och deformeras enligt dessa lagar i enlighet med tidsparametern t.

3.3.1 Nivåmängdsmetod

Konceptet med nivåmängsmetoden är att studera hyperytan $\Gamma_t \in \mathbb{R}^n$ som nollnivån för funktionen ϕ i $\mathbb{R}^{n+1}$. Fallet där $\Gamma_t \in \mathbb{R}^2$ kan följande definition ges,

$$\Gamma_t := \{\mathbf{x} \in \mathbb{R}^2; \quad \phi(\mathbf{x}, t) = 0\}. \tag{3.16}$$

Följande uttryck konstrueras för skapa en modell för hyperytan Γ_t under geometriskt flöde (3.10), detta görs genom derivering av ekvationen $\phi(\mathbf{x}, t) = 0$. Genom kedjeregeln fås,

$$\frac{\partial \phi}{\partial t} + \nabla \phi \cdot \frac{\partial \mathbf{x}}{\partial t} = 0. \tag{3.17}$$

Via hastighetsnormalen för geometriska flöden (3.10) härleds kurvans hastighet $\mathbf{v} = \dfrac{\partial \mathbf{x}}{\partial t}$, då skrivs (3.17),

$$\frac{\partial \phi}{\partial t} + \nabla \phi \cdot \mathbf{v} = 0. \tag{3.18}$$

Ekvationen (3.18) kallas för nivåmängdsekvationen och definieras av hastighetsvektorn $\mathbf{v}$ och begynnelsevillkoret ϕ_0 så att $\Gamma_0 := \{\mathbf{x} \in \mathbb{R}^2; \quad \phi_0(\mathbf{x}) = 0\}$. Hastighetsvektorns faktorer för krökningsflödet, enligt (3.10), blir då,

$$\mathbf{n} = -\frac{\nabla \phi}{|\nabla \phi|} \text{ och } \kappa = -\nabla \cdot \mathbf{n}, \tag{3.19}$$

där $\mathbf{n}$ är inåtriktade normalen för Γ_t [1]. Ekvationen (3.18) ger att,

$$0 = \frac{\partial \phi}{\partial t} + \nabla \phi \cdot \mathbf{v} = \frac{\partial \phi}{\partial t} - |\nabla \phi| \mathbf{v_n} = \frac{\partial \phi}{\partial t} - |\nabla \phi| \kappa = \frac{\partial \phi}{\partial t} - |\nabla \phi| \nabla \cdot (\frac{\nabla \phi}{|\nabla \phi|}) \tag{3.20}$$

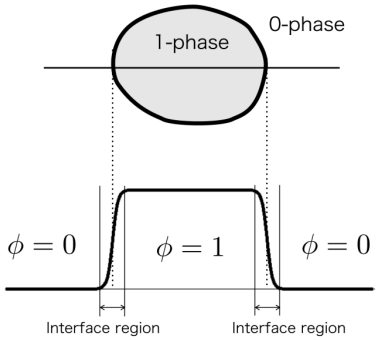
Ekvationen blir då,

$$\frac{\partial \phi}{\partial t} - |\nabla \phi| \nabla \cdot (\frac{\nabla \phi}{|\nabla \phi|}) = 0 \tag{3.21}$$

Ekvation 3.21 är nivåmängdsformuleringen för krökningsflödet[9][10][11].

3.3.2 Fasfältsmetod

Fasfältsmetoden gentemot nivåmängdsmetoden beskriver inte den explicita hyperytan, utan beskriver ett diffus gränsskikt mellan nivåytor som karaktäriseras av tangens hyperbolicus (tanh). Parametern δ bestämmer bredden av det diffusa gränsskiktet där Γ_t existerar [1]. Ett exempel på hur fasfältsfunktionen ϕ beskriver kurvan Γ ses i figur 3.2.



Figur 3.2: Ett exempel på hur fasfältsfunktionen ϕ kan beskriva två olika faser enlighet fasfältsmetoden. Nedersta figuren visar tvärsnittet på ϕ där kurvan Γ beskrivs som randen av ytan på den översta figuren. Reprinted from *A brief introduction to phase field method* written by R. Kobayashi, with the permission of AIP Publishing [12].

Fasfältsformuleringen för krökningsflöde är implementerad genom Allen-Cahn ekvationen, som formuleras

$$\frac{\partial \phi}{\partial t} = \Delta\phi - f'(\phi), \tag{3.22}$$

där $f'(\phi)$ bestämmer jämviktsprofilen ϕ_0. Funktionen $f(\phi)$ är en potential med två minimipunkter, motsvarande två stabila faser. Utseendet på jämviktsprofilen motiveras av det materialbeteende som ska modelleras.

Den modifierade Allen-Cahn ekvationen framställs via en sammanskrivning av (3.22) och (3.21), som resulterar i följande ekvation

$$\frac{\partial \phi}{\partial t} = \Delta\phi - f'(\phi) - |\nabla\phi|\nabla\left(\frac{\nabla\phi}{|\nabla\phi|}\right) = \nabla\left(\nabla\phi - \sqrt{2f(\phi)}\frac{\nabla\phi}{|\nabla\phi|}\right). \tag{3.23}$$

Genom att justera tanh-profilen så att $f(\phi) = \frac{8}{\delta^2}\phi^2(1-\phi)^2$. Med detta $f(\phi)$ blir jämviktsprofilen $\phi_0 = \frac{1}{2}\left(1 - \tanh\left(\frac{2d(x)}{\delta}\right)\right)$, där $d(x)$ betecknar avståndsfunktionen mellan punkten x och hyperytan Γ_t, samt introducera mobilitetsfaktorn M som bestämmer hastigheten för hur ytan utvecklas, får vi följande omskrivning av ekvation (3.23),

$$\frac{\partial \phi}{\partial t} = M\nabla \cdot \left(\nabla\phi - \frac{4\phi(1-\phi)}{\delta}\frac{\nabla\phi}{|\nabla\phi|}\right). \tag{3.24}$$

Den resulterade ekvationen (3.24) kallas för den modifierade Allen-Cahn ekvationen.

3.3.3 Cahn-talet

Definition 3.12 (Cahntalet)**.** *Cahntalet Cn definieras som förhållandet mellan bredden på det diffusa gränsskiktet δ och den karakteristiska längd L av ytan Γ enligt,*

$$Cn = \delta/L. \tag{3.25}$$

Bredden på det diffusa gränsskiktet kan med Cahntalet beskrivas som en dimensionslös storhet som tar hänsyn till ytan dimensioner. För ett droppformad yta kan L definieras som radien. Med ett konstant Cahntal kan ytan med olika former och storlekar jämföras. Litet Cn påvisar hög upplösning relativt δ, och stort Cn låg upplösning relativt δ.

3.3.4 Diffusiv skalning

Vid simulering av diffusion med Lattice-Boltzmann metoden gäller en relation i rumsled. Det visar sig att $\Delta t \backsim \Delta x^2$, det vill säga att en ökning av upplösning gör så att tiden går långsammare i simulationen. Diffusiva skalningen gör så det behövs fler iterationer för att simulera samma tidsintervall [13]. Exempelvis resulterar en dubblering av rumsupplösning i att antal iterationer fyrfaldigas.

4 Metod

Arbetet består av två huvudsektioner: analytiska och numeriska beräkningar. Avsnittet med numeriska beräkningar är vidare uppdelat i tre avsnitt: diffusiv skalning, areaförändring, och deformationsanalys.

4.1 Analytiska beräkningar

I avsnittet för de analytiska beräkningarna studerades ytors krökningar under krökningsflöde samt ytdiffusion analytiskt. Enkla ytor som cirkel och ellips användes vid beräkningar som gjordes för hand. Derivatan för initialdata som beräknades i detta avsnitt jämfördes med intitialderivatan från de numeriska beräkningarna.

4.2 Numeriska beräkningar

Numeriska beräkningar gjordes med beräkningsprogrammet Gesualdo, som simulerar geometriska flöden över ytor. Beräkningsprogrammet tillämpar Lattice-Boltzmann modellen. Initialdatan skapades i MATLAB, där ytor, upplösningar, samt parametrar bestämdes. Indata till programmet upprepas periodiskt i alla riktningar, efter val av randvillkor. Lämpliga värden av parametrar för de fortsatta simuleringarna valdes efter de första simuleringarna, där olika parametrar testades (se appendix B.3 för exakta värden). Värden på parametrar klassades som olämpliga om deformationen av figuren skedde oförutsägbart (se mer om detta i appendix C.2).

Den diffusiva skalningen undersöktes först för Allen-Cahn ekvationen, och sedan för den modifierade Allen-Cahn ekvationen. Detta gjordes för att data från samma tidspunkter ska kunna jämföras, då det kan krävas olika antal iterationer av programmet för att nå samma tid för olika simuleringar. När diffusiva skalningen beräknas krävs det att alla andra parametrar är samma, då målet är att jämföra lösningen av samma ekvation i olika upplösningar, därför användes Cahn-tal vid bestämning av bredden av den diffusa gränsskiktet (betecknas δ). En relation mellan iteration och upplösning togs

fram med regression över datapunkter där simuleringar hade utvecklats lika långt.

Om den modifierade Allen-Cahn ekvationen beskriver ytdiffusion i simuleringarna behöver arean bevaras. Arean hos initialdata bestående av fyra olika former testades med flera olika upplösningar, en gång med konstant δ för alla upplösningar och en gång med Cahn-tal anpassat efter upplösning. Därefter analyserades datan för areaförändringar.

Bilder från simuleringarna togs ut för att åskådliggöra deformationen hos initaldatan. Deformationen enligt simulering jämfördes sedan med vad som borde ske under ytdiffusion i teorin.

5 Resultat

5.1 Beräkningar för geometriska flöden

5.1.1 Krökningsflöde för en cirkel

Låt $\gamma(t) = (R\cos(t), R\sin(t))$, där $t \in [0, 2\pi)$, vara parametriseringen av en cirkel med radie $R > 0$. Båglängden av cirkelns kurva är då

$$s(t) = \int_0^t |\gamma(\tau)| d\tau = R \int_0^t d\tau = Rt.$$

Låt oss nu använda båglängden i parametriseringen genom följande omskrivning:

$$s = Rt \quad \rightarrow \quad t = \frac{s}{R},$$

som medför

$$\tilde{\gamma}(s) = \left(R\cos\left(\frac{s}{R}\right), R\sin\left(\frac{s}{R}\right)\right), \quad s \in [0,2\pi R). \tag{5.1}$$

Krökningen κ definieras enligt

$$\kappa = \left| \frac{\partial \mathbf{T}}{\partial s} \right|,$$

alltså derivatan av tangentvektorn. Tangenten $\mathbf{T}$ beräknas som

$$\mathbf{T}(s) = \gamma'(s) = \left(-\sin\left(\frac{s}{R}\right), \cos\left(\frac{s}{R}\right)\right),$$

och då får vi krökningen

$$\kappa = \left|\tilde{\gamma}''(s)\right| = \frac{1}{R}\left|\left(-\cos\left(\frac{s}{R}\right), -\sin\left(\frac{s}{R}\right)\right)\right| = \frac{1}{R}. \tag{5.2}$$

Krökningen av en cirkel är endast beroende av radien, och konstant över hela kurvan.

5.1.2 Ytdiffusion för en cirkel

Låt oss använda resultatet från krökningsflödet för en cirkel, så cirkeln γ har båglängdsparametriseringen (5.1). Hastigheten för ytdiffusion definieras enligt 3.10 som $v = -\Delta_\gamma \kappa n$, där

$$\Delta_\gamma \kappa(s) = \frac{\partial^2}{\partial s^2}\kappa, \tag{5.3}$$

och κ är krökningen, i cirkelns fall är $\kappa = \frac{1}{R}$. Eftersom κ är en konstant är $\Delta_\gamma \kappa(s) = 0$, och cirkeln är därför stationär under ytdiffusion.

5.1.3 Krökningsflöde för en ellips

Låt γ vara en ellips på formen

$$1 = \frac{x^2}{a^2} + \frac{y^2}{b^2},$$

där $a, b \in \mathbb{R}$ är konstanter. En parametrisering av γ är

$$\gamma(t) = (a\cos(t), b\sin(t)), \quad t \in [0, 2\pi). \tag{5.4}$$

Observera att denna parametrisering av en ellips inte är en båglängdsparametriseringsbar (Se appendix A.3), därför måste tangenten $\mathbf{T}$ beräknas som

$$\mathbf{T}(t) = \frac{\gamma'(t)}{|\gamma'(t)|} = \frac{(-a\sin(t), b\cos(t))}{\sqrt{a^2\sin^2(t) + b^2\cos^2(t)}}.$$

Krökningen κ är

$$\kappa = \left|\frac{\partial \mathbf{T}}{\partial s}\right|,$$

vilket med kedjereglen blir

$$\kappa = \left|\frac{\partial \mathbf{T}}{\partial s}\right| = \left|\frac{\partial \mathbf{T}}{\partial t}\frac{\partial t}{\partial s}\right|. \tag{5.5}$$

Vi har att

$$s(t) = \int_0^t |\gamma'(t)|d\tau \rightarrow \quad \frac{\partial t}{\partial s} = \frac{1}{|\gamma'(t)|}, \tag{5.6}$$

enligt (3.7), som insatt i (5.5) ger oss följande uttryck för krökningen

$$\kappa(t) = \frac{|\mathbf{T}'(t)|}{|\gamma'(t)|}. \tag{5.7}$$

Vid derivering av $\mathbf{T}$ fås uttrycket

$$\mathbf{T}'(t) = \left(\frac{-ab^2\cos(t)}{\sqrt{(a^2\sin^2(t) + b^2\cos^2(t))^3}}, \frac{-a^2b\sin(t)}{\sqrt{(a^2\sin^2(t) + b^2\cos^2(t))^3}}\right),$$

och insatt i (5.7) ger

$$\kappa(t) = \frac{ab}{\sqrt{(a^2\sin^2(t) + b^2\cos^2(t))^3}}, \tag{5.8}$$

se appendix A.5 för uträkningen.

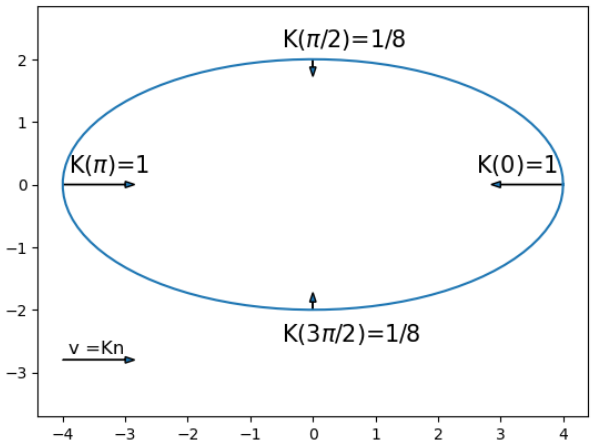
Enkla fall att studera krökningsflödet av en ellips är när vinkeln $t \in \{0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}\}$, då få vi

$$\begin{cases} \kappa(0) = \kappa(\pi) = \frac{a}{b^2} \\ \kappa(\frac{\pi}{2}) = \kappa(\frac{3\pi}{2}) = \frac{b}{a^2} \end{cases}. \tag{5.9}$$

Exempel: Låt $a = 4$ och $b = 2$, då blir krökningen

$$\begin{cases} \kappa(0) = \kappa(\pi) = 1 \\ \kappa(\frac{\pi}{2}) = \kappa(\frac{3\pi}{2}) = \frac{1}{8} \end{cases}.$$

Se (5.1) för visualisering av krökningsflödet av en ellips.



Figur 5.1: Hastighetsvektorerna är ritade och respektive värde på krökningen är utskrivet.

5.1.4 Ytdiffusion för en ellips

Från krökningsflödet av en ellips har vi enligt ekvation (5.8) att

$$\kappa(t) = \frac{ab}{\sqrt{(a^2\sin^2(t) + b^2\cos^2(t))^3}},$$

vilket kommer användas för beräkning av $\Delta_\gamma\kappa$. Kedjeregeln medför följande

$$\Delta_\gamma\kappa = \frac{\partial^2}{\partial s^2}\kappa = \frac{\partial}{\partial s}\left(\frac{\partial}{\partial s}\kappa\right) = \frac{\partial t}{\partial s}\frac{\partial}{\partial t}\left(\frac{\partial t}{\partial s}\frac{\partial}{\partial t}\kappa\right),$$

där

$$\frac{\partial t}{\partial s} = \frac{1}{|\gamma'(t)|} = \frac{1}{\sqrt{a^2\sin^2(t) + b^2\cos^2(t)}}$$

enligt (5.6). Alltså blir

$$\Delta_\gamma\kappa(t) = -3ab\frac{(a^2-b^2)(\cos^2(t)-\sin^2(t))}{(a^2\sin^2(t)+b^2\cos^2(t))^{7/2}} - \frac{6(\sin(t)\cos(t))^2(a^2-b^2)^2}{(a^2\sin^2(t)+b^2\cos^2(t))^{9/2}}, \tag{5.10}$$

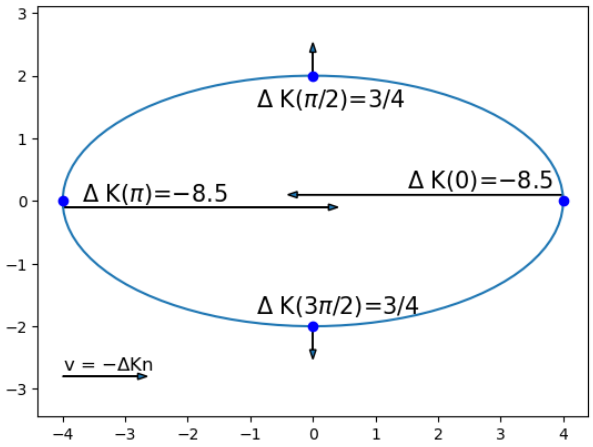
se appendix A.6 för beräkningarna. Observera att sista termen i (5.10) är lika med noll om $t \in \{0, \frac{\pi}{2}\,\pi, \frac{3\pi}{2}\}$, vilket gör dessa punkter lätta att beräkna:

$$\begin{cases} \Delta_\gamma\kappa(0) = \Delta_\gamma\kappa(\pi) = \frac{-3ab(a^2-b^2)}{b^{7/2}} \\ \Delta_\gamma\kappa(\frac{\pi}{2}) = \Delta_\gamma\kappa(\frac{3\pi}{2}) = \frac{3ab(a^2-b^2)}{a^{7/2}} \end{cases}. \tag{5.11}$$

Exempel: Låt $a = 4$ och $b = 2$, då får vi att

$$\begin{cases} \Delta_\gamma\kappa(0) = \Delta_\gamma\kappa(\pi) = -3\sqrt{2^3} \approx -8.5 \\ \Delta_\gamma\kappa(\frac{\pi}{2}) = \Delta_\gamma\kappa(\frac{3\pi}{2}) = \frac{3}{4} \end{cases}. \tag{5.12}$$

Se figur (5.2) för visualisering av ytdiffusionen.



Figur 5.2: De blåa punkterna visar vart varje hastighetsvektor börjar, då de för $t = 0$ och $t = \pi$ förskjutits i y-led för läsbarhetens-skull.

5.1.5 Beräkning för jämförelse med numerisk derivata

Ellipsen som simulerades numeriskt i upplösning 100x100 är på formen

$$\frac{(x-50)^2}{(25\sqrt{2})^2} + \frac{(y-50)^2}{(\frac{25\sqrt{2}}{2})^2} = 1,$$

och då är

$$\begin{cases} a = (25\sqrt{2})^2 \\ b = (\frac{25\sqrt{2}}{2})^2 \end{cases}. \tag{5.13}$$

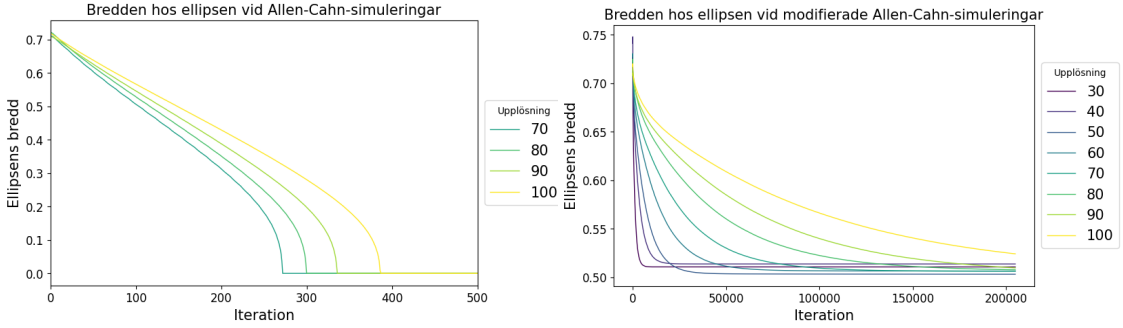
Värdena från (5.13) insatt i (5.11) ger följande värden på $\Delta_\gamma \kappa$

$$\begin{cases} \Delta_\gamma \kappa(0) = \Delta_\gamma \kappa(\pi) = 90(2)^{3/4} \approx -151.36 \\ \Delta_\gamma \kappa(\frac{\pi}{2}) = \Delta_\gamma \kappa(\frac{3\pi}{2}) = \frac{45(2)^{1/4}}{4} \approx 13.38 \end{cases}. \tag{5.14}$$

5.2 Numeriska beräkningar

5.2.1 Diffusiv skalning

Vid simuleringar med olika upplösningar observerades en tydlig tidsskillnad, kallad diffusiv skalning. För att jämföra analyser i samma tidsskala beräknas den diffusiva skalningen. Inre ellipsbredd mättes för varje iteration per upplösning, både för simuleringar med delta från Cahn-tal och konstant δ. Bredden normaliseras för enklare jämförelser mellan upplösningar.



(a) Bredden av ellipsen vid Allen-Cahn simuleringar med användning av Cahn-tal

(b) Bredden av ellipsen vid den modifierad Allen-Cahn simuleringar med användning av Cahn-tal

Figur 5.3: Grafer med normaliserade breddens storlek på y-axel och iteration på x-axel.

Figurerna 5.3a och 5.3b visar normaliserade ellipsdiametrar för varje iteration och upplösning med Cahn-talet. Figur 5.3a använder Allen-Cahn ekvationen (3.22) och 5.3b den modifierade Allen-Cahn (3.24). Appendix C.1 förklarar varför endast upplösningarna 70-100 visas i 5.3a.

Observera att utvecklingen av ellipsens bredd per upplösning skiljer sig åt i figurerna 5.3a och 5.3b. Se appendix B.1 för beräkningen av den diffusiva skalningen. För Allen-Cahn ekvationen är den diffusiva skalningen $\Delta x^2 \backsim \Delta t$ enligt teorin, men resultaten visar att den modifierade Allen-Cahn ekvationen samt Allen-Cahn ekvationen (se Appendix C.1) inte följer samma skalning. Tabellen nedan presenterar resultat för samtliga simuleringsvarianter, tillsammans med deras korresponderande värde på n i den diffusiva skalningen $\Delta x^n \backsim \Delta t$.

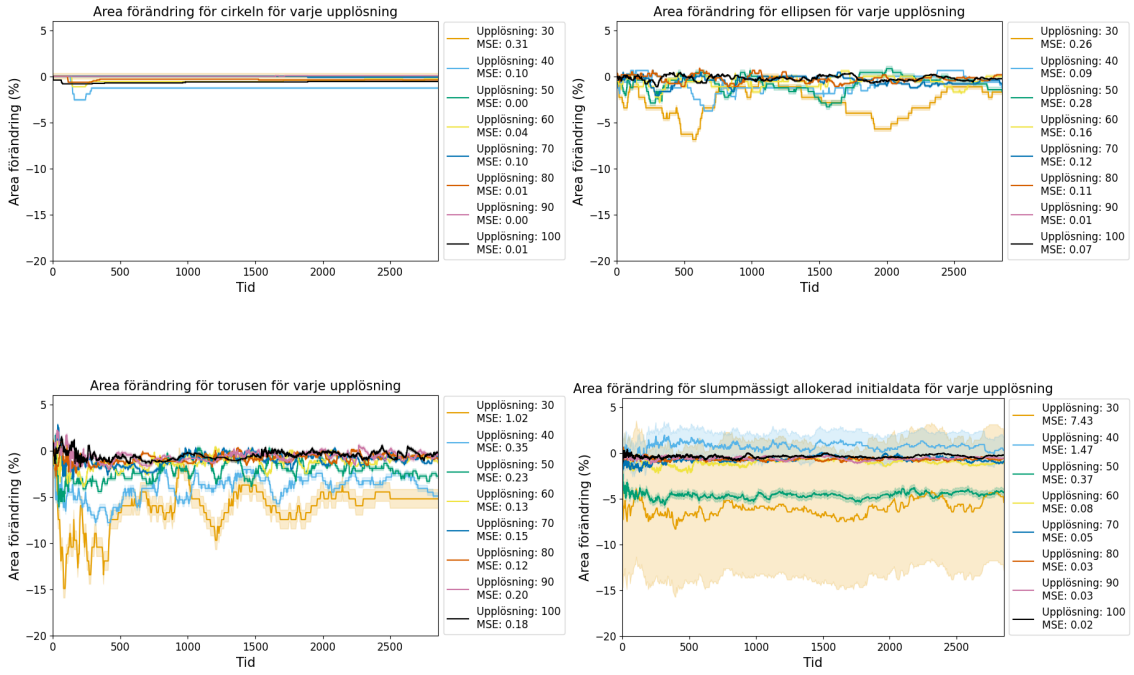
Simuleringstyp	δ	**Medelvärde av n**	**SD av n**	**95% Konfidensintervall**
Modifierade Allen-Cahn ekvationen	$Cahn - tal$	3.55	0.03	[3.4951, 3.6003]
Allen-Cahn ekvationen	$Cahn - tal$	0.95	0.02	[0.9101, 0.9840]

Tabell 5.1: Diffusiva skalningen för simuleringarna

Simuleringstiden beräknas enligt Tid $=$ Iteration$(\frac{30}{\text{Upplösning}})^n$ för varje simuleringstyp. Vidare analyser fokuserar enbart på tid och begränsas till den längsta tiden uppnådd av upplösning 100.

5.2.2 Areaförändring med konstant delta

Areaförändringen studerades över tid för fyra olika former: cirkel, ellips, torus, och slumpmässigt allokerat initialdata, se appendix B.2 för beräkning. Areaförändringen är undersökt med simuleringsdata som har ett konstant $\delta = 3$.

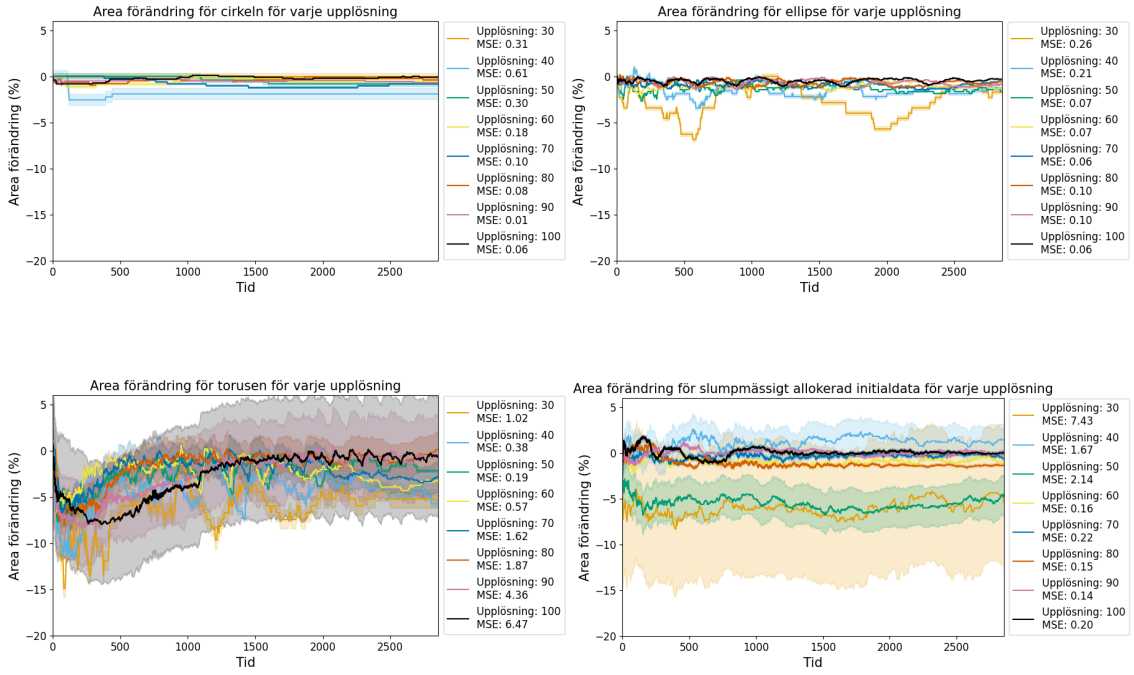


Figur 5.4: Areaförändring för konstant $\delta = 3$. x-axeln som representerar tiden går från 0 till $\backsim$ 2850 och y-axeln som representerar procentuell areaförändring som skiljer sig för de 4 olika formerna. Till höger om varje graf kan en ruta med medelkvadratsavvikelsen hittas.

Graferna illustrerar areaförändringen som en funktion av tiden, där varje upplösning av formerna representeras av en unik färg. I figurerna representeras medelkvadratsavvikelsen för areaförändringen runt själva grafen. Det kan noteras att areaförändringen skiljer sig bland formerna och upplösningarna. I cirkeln med upplösning 90 har minsta medelkvadratsavvikelsen som är 0 och största medelkvadratsavvikelsen 7.43 finns hos slumpmässigt allokerat initialdata vid upplösningen 30. Vi kan notera att areaförändringen är mer stabil vid hög upplösning, jämför vi de olika formerna kan det noteras att torusen har högst procentuell areavariation i lägre upplösningar.

5.2.3 Areaförändring med Cahn-tal

I följande grafer är det samma figurer, färger och metod som används för att undersöka areaförändringen i figur 5.4, enda skillnaden är datan. I figuren används simuleringsdata genererad med Cahn-tal.

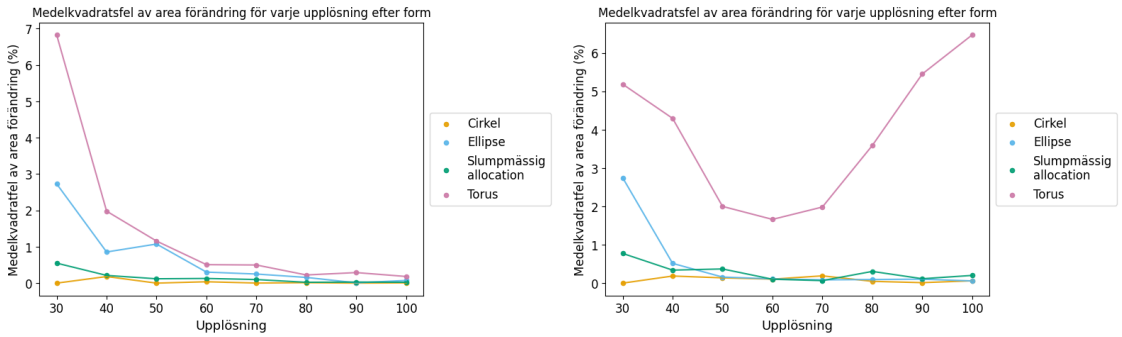


Figur 5.5: Areaförändring för simulation med Cahn-tal. x-axeln som representerar tiden går från 0 till ∽ 2850 och y-axeln som representerar procentuell areaförändring som skiljer sig för de 4 olika formerna. Till höger om varje graf kan en ruta med medelkvadratsavvikelsen hittas.

Även i dessa grafer illustreras areaförändringen som en funktion av tiden och där varje upplösningen representerar en färg. Vi notera att graferna är mer instabila men jämnar ut sig i slutet vid tiden 2500. Även här särskiljer sig torusen från resterande former eftersom den har stor areaförändring i alla upplösningar.

5.2.4 Medelkvadratsavvikelse

I följande grafer representeras medelkvadratsavvikelsen för areaförändringen för cirkeln, ellipsen, torusen och slumpmässigt allokerat initialdata.



(a) medelkvadratsavvikelse av areaförändring för varje upplösning efter form med $\delta = 3$.

(b) Medelkvadratsavvikelse av areaförändring för varje upplösning efter form simulerat med Cahn-tal.

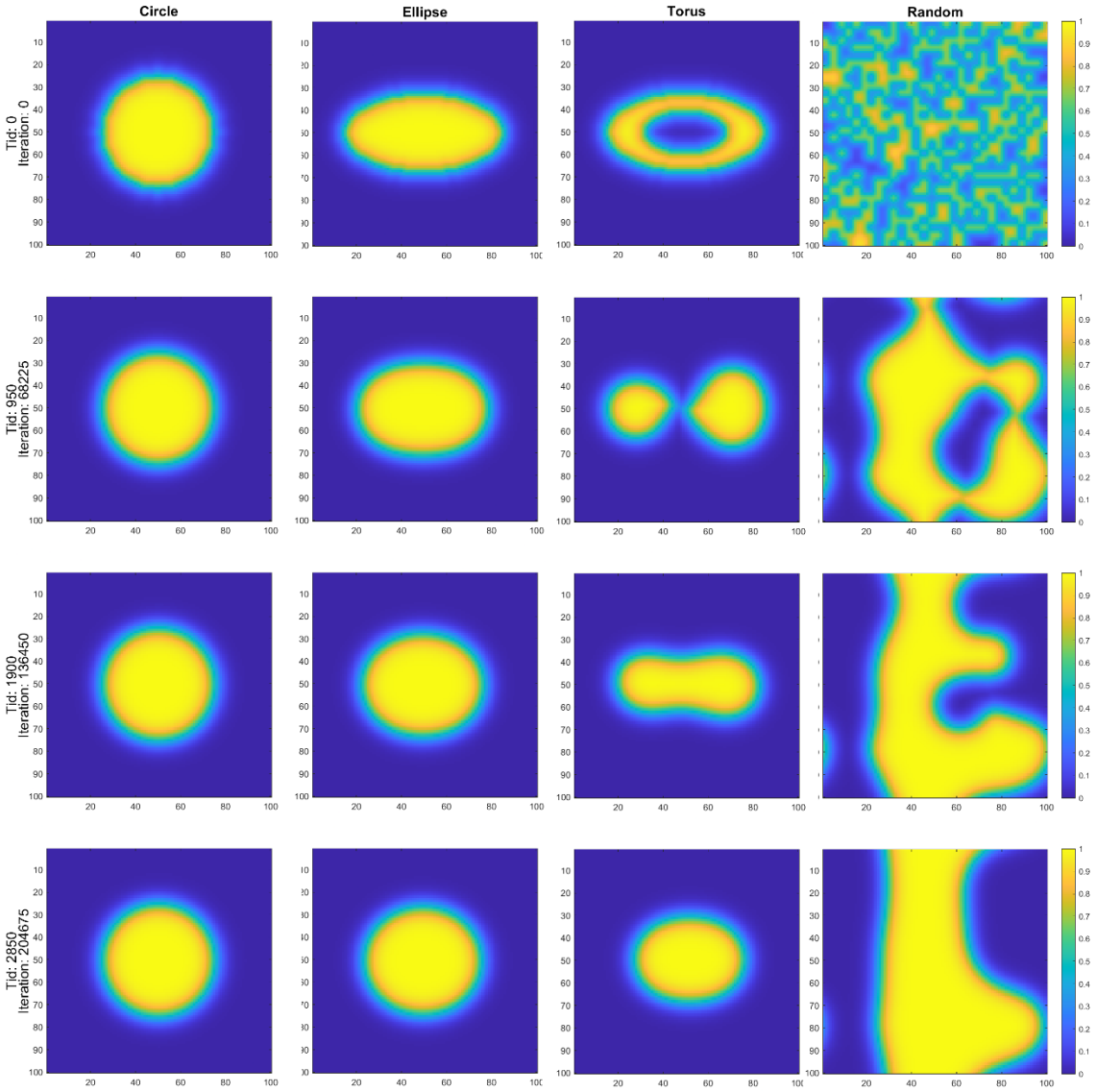
Figur 5.6: x-axeln går från upplösningen 30 till 100 medan y-axeln representerar medelkvadratfel av area förändring (%) och går från 0 till 7.

Figur 5.6a presenterar areaförändringen med konstant $\delta = 3$, medan figur 5.6b visar

areaförändringen för Cahn-talet. Varje form särskiljs genom användning av en unik färg. Graferna i figur 5.6 belyser hur medelkvadratsavvikelsen för areaförändringen förändras under utveckling när upplösningen ökar. Vid högre upplösningar kan vi notera att för fixt δ så går alla formers medelkvadratsfel mot noll, medan när Cahn-tal används så går alla formers medelkvadratsfel mot noll förutom torusen. Torusen börjar gå mot noll men vid upplösning 60 så ökar medelkvadratsfelet och den trenden fortsätter, se appendix C.2 för förklaring.

5.2.5 Evolution hos former

Figuren nedan visar evolutionen för de olika formerna som studerats.



Figur 5.7: Evolution för olika former med modifierade Allen-Cahn ekvationen. I figuren visas datan från upplösning 100 x 100, $M = 1$ och $\delta = 10$ som valts efter användning av Cahn-tal. Varje rad presenterar en tidpunkt som formen är tagen från. Höger om varje rad visas en färgskala som går från noll till ett.

Notera att formens yta är allt över $\frac{1}{2}$ på färgskalan. Vi kan notera i illustrationerna att med tiden så rör sig alla former förutom den slumpmässiga allokerade initialdatan mot en

cirkulär form.

5.2.6 Initalderivata för krökningen hos en ellips

Derivatan för ellipsen i figur 5.7, med upplösning 100x100 och $\delta = 10$ samt upplösningen 100x100 och $\delta = 3$ uppskattas i början av simulationen genom följande uppställning

$$\frac{\Delta höjd}{\Delta bredd},$$

där $höjd$ är ellipsens höjd och $bredd$ är ellipsens bredd. Uppställningen motsvarar

$$\frac{\Delta\kappa(\frac{\pi}{2})}{\Delta\kappa(0)} \approx \frac{13.38}{-151.36} \approx -0.088, \tag{5.15}$$

enligt resultat (5.11) och

$$\frac{\Delta höjd}{\Delta bredd} = \frac{-2\Delta\kappa(\frac{\pi}{2})\Delta t}{-2\Delta\kappa(\frac{0}{2})\Delta t} = \frac{\Delta\kappa(\frac{\pi}{2})}{\Delta\kappa(0)}.$$

Medelderivatan för iterationer i de tidiga intervallen [15,60] respektive [15,30] när $\delta = 3$ är $\frac{\Delta höjd}{\Delta bredd} \approx -0.253$ och när $\delta = 10$ är $\frac{\Delta höjd}{\Delta bredd} \approx -0.0229$. De tidiga intervallen valdes då grafen över breddförändringen var jämn först efter iteration 15.

6 Diskussion

6.1 Diskussion av resultat

6.1.1 Analytiska beräkningar

Vi observerar att cirkeln är stationär i ytdiffusion, resultatet i avsnitt 5.1.2 tyder på att formen inte ändrar sig. För krökningsflöde bevaras cirkelformen då krökningen är konstant över hela randen, enligt (5.2), men krymper med konstant hastighet enligt teoriavsnittet för krökningsflöde. Ellipser krymper under krökningsflöde och de spetsigare delarna på ellipsen krymper snabbare än de trubbiga ändarna. Observera detta i figur (5.1). Ytdiffusion för en ellips får ellipsen att långsamt växa på de trubbiga delarna och krympa relativt snabbt på de spetsiga delarna, detta observeras i figur (5.2). Krökningsflödet för en ellips är till skillnad från en cirkel, inte konstant, utan beroende av formen på randen. Vi kan notera att ellipsen jämnar ut sig i ytdiffusion, precis som för krökningsflöde, men arean kommer bevaras enligt (3.15).

6.1.2 Diskussion numeriska beräkningar

Diffusiv skalning för Allen-Cahn

I figur 5.3a där Allen-Cahn ekvationen simuleras kan vi observera att ellipsens bredden går mot noll för de fyra olika upplösningar. Högre upplösning kräver fler iterationer än låg upplösning för att bredden ska gå mot noll. När vi jämför den teoretiska diffusiva skalningen, där en förändring i upplösning motsvarar samma förändring i kvadrat hos tiden, verkar det inte stämma överens med resultatet. Se tabell 5.1 för uppmätt data.

Diffusiv skalning för modifierade Allen-Cahn

I figur 5.3b simuleras den modifierade Allen-Cahn ekvationen och där kan vi observera att alla upplösningar från 30 till 100 når en stabil form vid bredden 0.5. Diffusiva skalningen för modifierade Allen-Cahn skiljer sig från Allen-Cahn, för den modifierade ekvationen blir relationen $\Delta t \backsim \Delta x^{3.55}$ istället för $\Delta t \backsim \Delta x^{0.95}$. Skillnaden i diffusiv skalning tyder på att ekvationerna inte simulerar samma diffusion eller geometriska flöde.

Areaförändring

I figur 5.4 kan vi notera areaförändringen för olika figurer i olika upplösningar med ett konstant $\delta = 3$. Valet av $\delta = 3$ gjordes efter tester av olika δ, och $\delta = 3$ gav bäst resultat, mer om detta kan hittas under Appendix D - felkällor. Målet är att medelkvadratsavvikelsen (MSE) ska vara noll. Om vi studerar de olika formerna i de olika upplösningarna i figur 5.4, kan vi notera att cirkeln ger bäst resultat i upplösning 90, där MSE är noll. Vi noterar att högre upplösning av figur ger bättre resultat, samma sak observeras i figur 5.6a.

Om vi jämför figurerna 5.4 med 5.5 så kan man observera att medelkvadratsavvikelsen för areaförändringen är större vid användning av Cahn-tal än med konstant δ. Detta ihop med att säkerheten verkar öka för högre upplösning visar på att ett litet δ relativt upplösning ger bättre bervarning av area. Om medelkvadratsavvikelsen hos arean är nära noll innebär det att arean bevaras väl, vilket indikerar att modifierade Allen-Cahn kan beskriva ytdiffusion.

Viktigt att notera är att vi inte haft möjligt att undersöka högre upplösningar än 100, så vi kan inte säga med säkerhet om högre upplösning alltid ger bättre areabervarning. Se kapitel 6.3 för fortsatt förklaring.

Evolution hos former

För att avgöra om modifierade Allen-Cahn simuleringarna beskriver ytdiffusion så behöver vi också analysera om formerna utvecklas i enlighet med ytdiffusion. I diskussionen för de analytiska beräkningarna observerades det att cirkeln är en stationär form, och att ellipsen jämnar ut sig, så trubbiga delar blir spetsigare och tvärt om. Följande observationer görs från figur 5.7. Cirkelformen bevaras, som förväntat för ytdiffusion. Ellipsen växer på höjden och krymper på bredden. Den rör sig långsammare ju mer cirkelformad den blir, och i sista tidspunkten är den nästan en cirkel. Det går därför att anta att den skulle bli stabil när den når en cirkelform. Torusen splittras i två runda ytor som sen växer samman och bildar en form som liknar ellipsen utveckling vid tid 950. Vi antar att torusen kommer fortsätta utvecklas som ellipsen i tidspunkt 950 och också gå mot en cirkel-form. Slumpmässigt allokerat initaldata klumpar ihop sig precis som torusen ihop sig till större, sammanhängande ytor, däremot formar den en rand tvärs över bilden. Viktigt att notera är att bilden upprepas periodiskt i alla riktningar, så därför påverkas bilden även av ytor utanför bild, om dessa ligger tillräckligt nära. Med en tom ram runt slumpmässigt allokerat initialdata, så det som beskådas inte kan påverkas av ytor utanför bild, kan man föreställa sig att ytorna slås samman till en cirkel.

Initialderivata för krökningen hos en ellips

Från resultatet från (5.11) och avsnitt 5.2.6 är det svårt att dra en konkret slutsats huruvida modifierade Allen-Cahn ger upphov till samma initialderivata som ytdiffusion. Anledningen till detta är att vi endast har två data punkter, där den ena är lite större än vad som förväntas från den analytiska beräkningen (5.15), och den andra lite mindre.

6.2 Slutsats

Det går att dra slutsaten från det numeriska resultatet att arean bevaras väl när δ är litet relativt upplösningen, och mobiliteten är ett. Detta är en viktig observation då arean bevaras under ytdiffusion. Utöver dessa observationer stämmer även bilderna från simuleringen väl med vad som kan förväntas ske under ytdiffusion, när vi jämför med de analytiska beräkningarna. Detta påvisar att den modifierade Allen-Cahn ekvationen beskriver ytdiffusion väl när dessa val av parametrar görs.

6.3 Framtida studier

Under arbetets gång har vi stött på ett antal svårigheter som kan tas i åtanke vid framtida forskning för bättre resultat. En avgörande förändring är att använda en högprestanda dator. Vi var tvungna att hålla oss till upplösning ≤ 100 för annars hade koden tagit för lång tid. 100 000 iterationer tog tre dagar, och med en uppskattning kunde vi dra slutsatsen att upplösning 256 hade behövt 16 miljoner iterationer vilket vår dator och tidsram inte hade kunnat hantera. Därför hade en högrprestanda dator underlättat men också gett möjlighet till ett mer noggrant resultat.

I vår rapport kunde den numeriska datan visa att arean bevaras vilket innebär att den modifierade Allen-Cahn beskriver ytdiffusion. I framtida studier så kan det vara användbart att bevisa analytiskt vad som händer när δ går mot noll och om arean alltid bevaras. Det är även av intresse att fortsatt undersöka hur initialderivatan förhåller sig till den analytiska, då vi i denna studie endast testat två olika värden på δ per upplösning.

Litteraturförteckning

[1] Du Q, Feng X. The Phase Field Method for Geometric Moving Interfaces and Their Numerical Approximations. arXiv; 2019. Hämtad från: https://arxiv.org/abs/1902.04924. Doi: 10.48550/ARXIV.1902.04924.

[2] CoSiMa - koncept för digitaliserad utveckling av framtidens hållbara mjuka material. Göteborg: Research Chalmers; u.d. [citerad 5 maj 2023]. Hämtad från: https://research.chalmers.se/project/10485.

[3] CoSiMa - Koncept för industriell utveckling av framtidens hållbara mjuka material. Göteborg: Research Chalmers; u.d. [citerad 5 maj 2023]. Hämtad från: https://research.chalmers.se/project/8442.

[4] Innovationsmyndigheten Vinnova. CoSiMa - koncept för digitaliserad utveckling av framtidens hållbara mjuka material [Internet]. Göteborg: Innovationsmyndigheten Vinnova; u.d. [citerad 8 februari 2023]. Hämtad från: https://www.vinnova.se/p/cosima---koncept-for-digitaliserad-utveckling-av-framtidens-hallbara-mjuka-material/.

[5] A Persson LCB. Analys i flera variabler. Polen: Studentlitteratur; 2017.

[6] do Carmo MP. Differential Geometry of Curves and Surfaces. New Jersey, US: Prentice-Hall; 1976.

[7] Y Sun CB. Sharp interface tracking using the pahe-fiels equation. Science Direct; 2007. Hämtad från: https://www.sciencedirect.com/science/article/pii/S0021999106002531https. Doi: https://doi.org/10.1016/j.jcp.2006.05.025.

[8] Zhu XP. Lectures on Mean Curvature Flows. USA: American Mathematical Society and International Press; 2002.

[9] Giga Y. Surface Evolution Equations. Polen: Birkhäuser; 2006.

[10] Yun Gang Chen SG Yoshikazu Giga. Uniqueness and existence of viscosity solutions of generalized mean curvature flow equations. J. Differential Geom.; 1991. Hämtad från: https://projecteuclid.org/journals/journal-of-differential-geometry/volume-33/issue-3/Uniqueness-and-existence-of-viscosity-solutions-of-generalized-mean-curvature/10.4310/jdg/1214446564.full?tab=ArticleFirstPage. Doi: 10.4310/jdg/1214446564.

[11] Spruck LCEJ. Motion of level sets by mean curvature. I. J. Differential Geom.; 1991. Hämtad från: https://projecteuclid.org/journals/journal-of-differential-geometry/volume-33/issue-3/Motion-of-level-sets-by-mean-curvature-I/10.4310/jdg/1214446559.full. Doi: 10.4310/jdg/1214446559.

[12] Kobayashi R. A brief introduction to phase field method. AIP Publishing; 2010. Hämtad från: https://aip.scitation.org/doi/pdf/10.1063/1.3476232. Doi: https://doi.org/10.1063/1.3476232.

[13] Krüger KSSV Kusumaatmaja. The Lattice Boltzmann Method: Principles and Practice. Switzerland: Springer; 2017. Doi: 10.1007/978-3-319-44649-3.

A Appendix 1 – teori

A.1 Notationer

Notation	Betydelse
γ	Kurva
$\mathbf{v}$	Hastighetsvektor
$\mathbf{v_n}$	Hastighet i normalriktningen
$\mathbf{n}$	Normalvektor
κ	Krökningen
Δ_γ	Laplace-operator
s	Båglängdsparameter
ϕ	Nivåmängdsfunktionen
Γ_t	Kurva i $\mathbb{R}^2$, annars hyperyta i $\mathbb{R}^n$.
M	Mobilitetsfaktorn
$\mathbf{T}$	Tangenten
$\mathbf{B}$	Bi-normalen
Cn	Cahntalet
δ	Diffusa gränsskiktet bredd
Δx	Skalning av upplösning
Δt	Skalning av tid

A.2 Ordlista

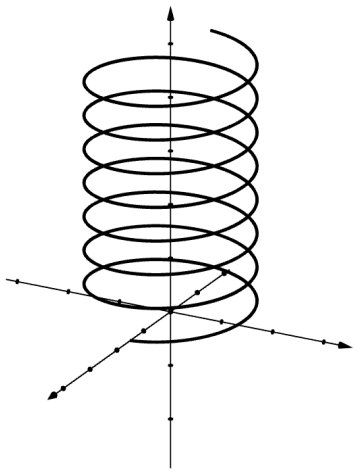
Svenska	Engelska
Bågländ	Arc length
Diffusiv skalning	Diffusive scaling
Diffust gränsskikt	Diffuse interface
Fastfältsmetod	Phase field method
Krökningsflöde	Mean curvature flow
Medelkvadratsavvikelse	Mean square error
Nivåmängdsmetod	Level-set method
Utvecklingsmetoder för hyperytor	Interface evolution methods
Ytdiffusion	Surface diffusion

A.3 Exempel till teorin

Exempel 2.1: Parametrisering av en helix [6]
Betrakat kurvan:

$$\gamma(t) = (acos(t), asin(t), bt), \quad t \in \mathbb{R}, \tag{A.1}$$

Där a och b är konstanter, och $t \in I$. Denna parametrisering ger en helix som vindlar runt z-axeln med en radie av a och en lutning av b. Denna kurva har sitt spår i $\mathbb{R}^3$ på cylindern $x^2 + y^2 = a^2$.



Figur A.1: En tredimensionell kurva som bildar en spiralform, beskriven av parametriseringen (A.1)

A.4 Båglängdsparametrisering av ellipsen

Givet ellipsen,

$$\gamma(t) = (acos(t), bsin(t)).$$

Vi har då,

$$\begin{aligned} x'(t)dt &= \frac{dx}{dt} \cdot dt, \\ y'(t)dt &= \frac{dy}{dt} \cdot dt, \\ ds &= [\text{Via Pythagoras sats}] = \sqrt{dx^2 + dy^2} \end{aligned}$$

Detta implicerar att,

$$ds = \sqrt{(x'(t)dt)^2 + (y'(t)dt)^2} = \sqrt{(x'(t))^2 + (y'(t))^2}dt$$

$$\Rightarrow s = \int ds = \int_0^{2\pi} \sqrt{(x'(t))^2 + (y'(t))^2}dt = \int_0^{2\pi} \sqrt{(-asin(t))^2 + (bcos(t))^2}dt$$

Denna integral är inte elementär, i stället krävs numeriska metoder för att beräkna den. Detta visar att en ellips inte kan båglängdsparametriseras.

A.5 Beräkningar för krökningsflöde av ellips

Låt γ vara en ellips på formen

$$1 = \frac{x^2}{a^2} + \frac{y^2}{b^2},$$

där $a, b \in \mathbb{R}$ är konstanter. Parametriseringen av γ är

$$\gamma(t) = (acos(t), bsin(t)), \quad t \in [0, 2\pi).$$

Tangenten $\mathbf{T}$ beräknas till

$$\mathbf{T}(t) = \frac{\gamma'(t)}{|\gamma'(t)|} = \frac{(-asin(t), bcos(t))}{\sqrt{a^2sin^2(t) + b^2cos^2(t)}}.$$

Tangentens derivata blir då:

$$\mathbf{T}'(t) = \frac{\sqrt{a^2sin^2(t)+b^2cos^2(t)}(-acos(t),-bsin(t)) - (-asin(t)bcos(t))\frac{(a^2-b^2)sin(t)cos(t)}{\sqrt{a^2sin^2(t)+b^2cos^2(t)}}}{a^2sin^s(t)+b^2cos^2(t)}.$$

Vi sorterar derivatans olika element och får en vektor på formen (x,y), där

$$x = \frac{-acos(t)}{\sqrt{a^2sin^2(t)+b^2cos^2(t)}} + \frac{asin(t)(a^2-b^2)sin(t)cos(t)}{(a^2sin^2(t)+b^2cos^2(t))^{3/2}}$$

och

$$y = \frac{-bsin(t)}{\sqrt{a^2sin^2(t)+b^2cos^2(t)}} - \frac{(a^2b-b^3)sin(t)cos^2(t)}{(a^2sin^2(t)+b^2cos^2(t))^{3/2}}).$$

Genom att skriva om uttrycken på gemensam nämnare blir uttrycken betydligt enklare:

$$x = \frac{-acos(t)}{\sqrt{a^2sin^2(t)+b^2cos^2(t)}} + \frac{asin(t)(a^2-b^2)sin(t)cos(t)}{(a^2sin^2(t)+b^2cos^2(t))^{3/2}} =$$

$$= \frac{a^3sin^2(t)cos(t) - ab^2sin^2(t)cos(t) - acos(t)(a^2sin^2(t)+b^2cos(t))}{(a^2sin^2(t)+b^2cos^2(t))^{3/2}} =$$

$$= \frac{a^3sin^2(t)cos(t) - ab^2sin^2(t)cos(t) - a^3cos(t)sin^2(t) - ab^2cos^3(t)}{(a^2sin^2(t)+b^2cos^2(t))^{3/2}} =$$

$$= \frac{-ab^2cos(t)(sin^2(t)+cos^2(t))}{(a^2sin^2(t)+b^2cos^2(t))^{3/2}} =$$

$$= \frac{-ab^2cos(t)}{(a^2sin^2(t)+b^2cos^2(t))^{3/2}}$$

och för y:

$$y = \frac{-bsin(t)}{\sqrt{a^2sin^2(t)+b^2cos^2(t)}} - \frac{(a^2b-b^3)sin(t)cos^2(t)}{(a^2sin^2(t)+b^2cos^2(t))^{3/2}}) =$$

$$= \frac{-a^2bsin^3(t) - b^3sin(t)cos^2(t) - a^2bsin(t)cos^2(t) + b^3sin(t)cos^2(t)}{(a^2sin^2(t)+b^2cos^2(t))^{3/2}} =$$

$$= \frac{-a^2bsin(t)(sin^2(t)+cos^2(t))}{(a^2sin^2(t)+b^2cos^2(t))^{3/2}} =$$

$$\frac{-a^2bsin(t)}{(a^2sin^2(t)+b^2cos^2(t))^{3/2}}.$$

Vi får alltså att

$$\mathbf{T}'(t) = \left(\frac{-ab^2cos(t)}{(a^2sin^2(t)+b^2cos^2(t))^{3/2}}, \frac{-a^2bsin(t)}{(a^2sin^2(t)+b^2cos^2(t))^{3/2}}\right).$$

Krökningsflödet $\kappa(t) = \frac{|\mathbf{T}'|}{|\gamma'(t)|}$ beräknas till följande

$$\frac{|\mathbf{T}'|}{|\gamma'(t)|} = \sqrt{\frac{a^2b^4cos^2(t)+a^4b^2sin^2(t)}{(a^2sin^2(t)+b^2cos^2(t))^4}} = \frac{ab\sqrt{a^2sin^2(t)+b^2cos^2(t)}}{(a^2sin^2(t)+b^2cos^2(t))^2} = \frac{ab}{(a^2sin^2(t)+b^2cos^2(t))^{3/2}}.$$

A.6 Beräkningar för ytdiffusion av ellips

Låt γ vara en ellips på formen

$$1 = \frac{x^2}{a^2} + \frac{y^2}{b^2},$$

där $a, b \in \mathbb{R}$ är konstanter. Parametriseringen av γ är

$$\gamma(t) = (acos(t), bsin(t)), \quad t \in [0, 2\pi).$$

Krökningen definieras som $\Delta\kappa$, där κ är krökninsflödet hos ellipsen. I tidigare avsnitt beräknades detta till

$$\kappa(t) = \frac{|\mathbf{T}'|}{|\gamma'(t)|} = \frac{ab}{(a^2sin^2(t) + b^2cos^2(t))^{3/2}}.$$

Kedjeregeln medför följande

$$\Delta\kappa = \frac{\partial^2}{\partial s^2}\kappa = \frac{\partial}{\partial s}\left(\frac{\partial}{\partial s}\kappa\right) = \frac{\partial t}{\partial s}\frac{\partial}{\partial t}\left(\frac{\partial t}{\partial s}\frac{\partial}{\partial t}\kappa\right),$$

där

$$\frac{\partial t}{\partial s} = \frac{1}{|\gamma'(t)|} = \frac{1}{\sqrt{a^2sin^2(t) + b^2cos^2(t)}}$$

efter resultatet av (5.6).

Vi har att

$$\Delta\kappa = \frac{\partial t}{\partial s}\frac{\partial}{\partial t}\left(\frac{\partial t}{\partial s}\frac{\partial}{\partial t}\kappa\right) = \frac{ab}{|\gamma'(t)|}\frac{\partial}{\partial t}\left(\frac{1}{|\gamma'(t)|}\frac{\partial}{\partial t}(a^2sin^2(t) + b^2cos^2(t))^{3/4}\right).$$

Vi deriverar innersta uttrycket och byter ut andra $|\gamma'(t)|$ mot $\sqrt{a^2sin^2(t) + b^2cos^2(t)}$, så nu är

$$\Delta\kappa = \frac{ab}{|\gamma'(t)|}\frac{\partial}{\partial t}\left(\frac{3(2(b^2 - a^2)sin(t)cos(t))}{2(a^2sin^2(t) + b^2cos^2(t))^3}\right) = \frac{3ab(b^2 - a^2)}{|\gamma'(t)|}\frac{\partial}{\partial t}\frac{sin(t)cos(t)}{(a^2sin^2(t) + b^2cos^2(t))^3}.$$

Sista faktorn i uttrycket deriveras med hjälp av kvotregeln:

$$f(t) = \frac{g(t)}{h(t)}, \quad f'(t) = \frac{g'(t)h(t) - g(t)h'(t)}{(h(t)^2)},$$

där

$$\begin{cases} g(t) = sin(t)cos(t)) \\ h(t) = (a^2sin^2(t) + b^2cos^2(t))^3 \end{cases} \tag{A.2}$$

och

$$\begin{cases} g'(t) = cos^2(t) - sin^2(t) \\ h'(t) = 6(a^2 - b^2)sin(t)cos(t)(a^2sin^2(t) + b^2cos^2(t))^2 \end{cases}. \tag{A.3}$$

Om (A.2) och (A.3) sätts in i kvotregeln fås följande resultat:

$$\frac{3ab(b^2 - a^2)}{|\gamma'(t)|}\frac{\partial}{\partial t}\frac{sin(t)cos(t)}{(a^2sin^2(t) + b^2cos^2(t))^3} =$$

$$\frac{-3ab(a^2 - b^2)}{|\gamma'(t)|}\frac{(cos^2(t) - sin^2(t))(a^2sin^2(t) + b^2cos^2(t))^3 - 6(sin(t)cos(t))^2(a^2 - b^2)(a^2sin^2(t) + b^2cos^2(t))^2}{(a^2sin^2(t) + b^2cos^2(t))^6}$$

$$= -3ab\frac{(a^2 - b^2)(cos^2(t) - sin^2(t))}{(a^2sin^2(t) + b^2cos^2(t))^{7/2}} - \frac{6(sin(t)cos(t))^2(a^2 - b^2)}{(a^2sin^2(t) + b^2cos^2(t))^{9/2}}.$$

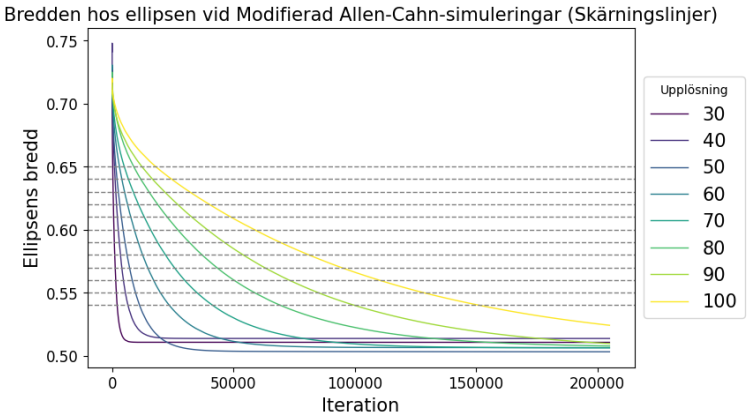
Alltså är

$$\Delta\kappa(t) = -3ab\frac{(a^2 - b^2)(cos^2(t) - sin^2(t))}{(a^2sin^2(t) + b^2cos^2(t))^{7/2}} - \frac{6(sin(t)cos(t))^2(a^2 - b^2)}{(a^2sin^2(t) + b^2cos^2(t))^{9/2}}.$$

B Appendix 2 – Numerisk

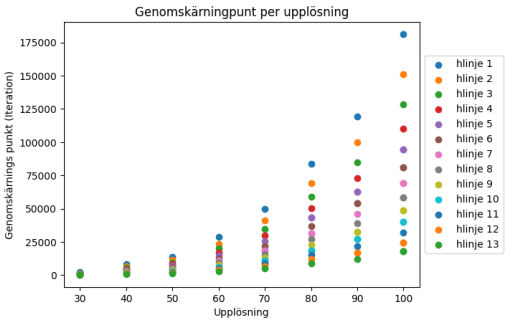
B.1 Uträkning av diffusiva skalning

För att beräkna den diffusiva skalningen används data från ellipsens bredd för varje upplösning. Vi väljer ett intervall inom området där vi kan observera en utveckling i bredden. Med simuleringarna på den modifierade Allen-Cahn var intervallet $[0{,}65, 0{,}53]$. Flera punkter väljs inom detta intervall och vi samlar in iterationen där varje ellips, för varje upplösning hade samma bredd. Figuren nedan illustrerar detta och visar flera horisontella linjer som skär graferna.

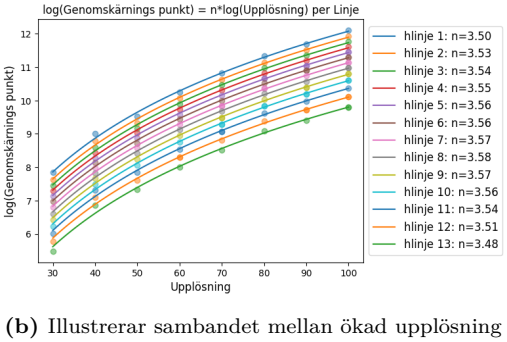


Figur B.1: Skärningslinjer genom bredden av ellipsen vid den modifierad Allen-Cahn simuleringar med användning av Cahn-tal

Skärningspunkter samlas in tillsammans med motsvarande längd av bredden. Då mäts antalet iterationer för varje upplösning vid samma längd av bredden för ellipsen. Därefter ritas datapunkterna med respektive horisontella linjer i figur B.2a för varje upplösning, vilket avslöjar en exponentiell utveckling. Datapunkterna passas till modellen $log(iteration) = a + n \times log(upplösning)$. Vi är intresserade av värdet för n, eftersom a bara ger oss startpunkten för utvecklingen. Värdet n ger relationen mellan upplösning och tid: $\Delta x^n \backsim \Delta t$. Figuren nedan visar $log(iteration)$ som en funktion av upplösning.



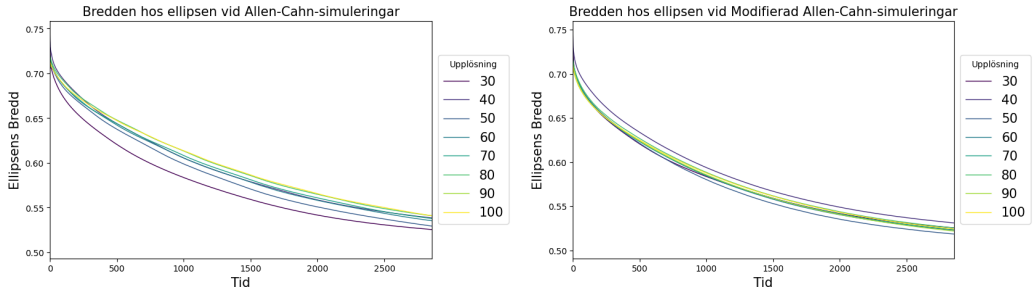
(a) Skärningspunkter per Skärningslinje, visar en exponentiell utveckling av iterationer i förhållande till upplösning.



(b) Illustrerar sambandet mellan ökad upplösning och antalet iterationer för att uppnå samma form, samt den statistiska analysen av n för att beskriva denna relation.

Figur B.2b visar att ökning av upplösningen ökar antalet iterationer som krävs för att uppnå samma form. Värdet för n analyseras statistiskt, där vi beräknar

medelvärdet, standardavvikelsen och ett konfidensintervall. Medelvärdet av n används för att beräkna tidsutvecklingen för varje simulering. Tiden beräknas som Tid = iteration $\times$ $(\text{upplösning}_0/\text{upplösning}_i)^n$ där i representerar olika upplösningar. Genom att använda tid istället för iteration presenteras utvecklingen av bredden för alla upplösningar i figurerna B.3a och B.3a, med konstant $\delta = 3$ respektive δ från Cahn-tal.



(a) Bredden av ellipsen vid den modifierad Allen-Cahn simuleringar över tid för konstant $\delta = 3$

(b) Bredden av ellipsen vid den modifierad Allen-Cahn simuleringar över tid för δ från Cahntal

Det observeras att utvecklingen av bredden över tid är betydligt jämnare när man använder tid istället för iteration, vilket framgår av en jämförelse mellan figur 5.3b och figur B.3b.

B.2 Ekvationen för area ändring

$$\mathrm{A}_{\%_i} = \frac{(A_0 - A_i)}{A_0} \times 100 \tag{B.1}$$

Areaförändring $\mathrm{A}_{\%_i}$ vid iteration i beräknas från den ursprungliga arean A_0 till area A_i där $i \in [0, N]$ där N är antalet iterationer som används. Resultatet av areaförändringen är i procent.

B.3 Parametrar för simuleringarna

I vår studie genomförde vi simuleringar med olika parametrar för att analysera och förstå beteendet i Gesualdo. Simuleringarna utfördes i tre tester, där varje test representeras av en separat tabell.

I det första testet utförde vi simuleringar för att observera hur olika parameterkombinationer påverkade dessa. Tabell B.1 visar parametrarna som användes i dessa simuleringar. Genom att undersöka alla möjliga kombinationer utförde vi 144 simuleringar för fyra olika former, med 100 000 iterationer.

Initiala simuleringar (Alla kombinationer)

Upplösningar	32, 64, 128, 256
M	0.5, 1, 1.5
δ	1, 2, 5

Tabell B.1: Första simuleringar

Efter att ha analyserat resultaten från första testet observerade vi vissa beteendeproblem relaterade till höga mobilitetsvärden (M) och små värden för δ. För att åtgärda problemen

gjordes nya simuleringar med uppdaterade parametrar, som visas i Tabell B.2. Återigen utförde vi 144 simuleringar för fyra former, med alla parameterkombinationer och med 100 000 iterationer.

Andra simuleringsrundan (Alla kombinationer)

Upplösningar	32, 64, 128, 256
M	1, $\frac{3.5}{3}$, 1.5
δ	3, 5, 7

Tabell B.2: Andra simuleringsrundan

Baserat på resultatet från de två testerna kom vi fram till parametrarna som presenteras i Tabell B.3. I detta test fokuserade vi på två olika metoder: en med δ-värden härledda från Cahn-talet och en annan med konstant δ. I sista simuleringarna utfördes 200 500 iterationer.

Slutgiltiga simuleringar

δ **från Cahn-tal =** 0.14140938

Upplösningar	30	40	50	60	70	80	90	100
δ	3	4	5	6	7	8	9	10
M	1							

Konstant delta = 3

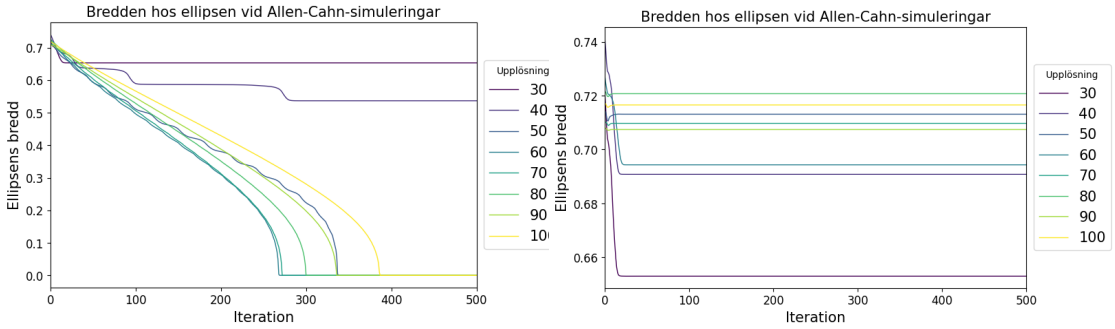
Upplösningar	30	40	50	60	70	80	90	100
δ	3							
M	1							

Tabell B.3: Slutgiltiga simuleringar

C Appendix 3 – Felkällor

C.1 Anomalier vid Allen-Cahn simuleringar

I följande bilder kan bredden hos ellipsen vid Allen-Cahn observeras med användning av Cahn-tal samt $\delta = 3$. I graferna kan åtta olika upplösningar observeras: 30, 40, 50, 60, 70, 80, 90 och 100. Bredden undersöks i förhållande till antal iterationer. Vi kan notera att bredden rör sig mellan 0.74 och 0 för konstant δ och 0.7 och 0 för Cahn-talet, medan iterationer går från 0 till 500 för båda fallen.

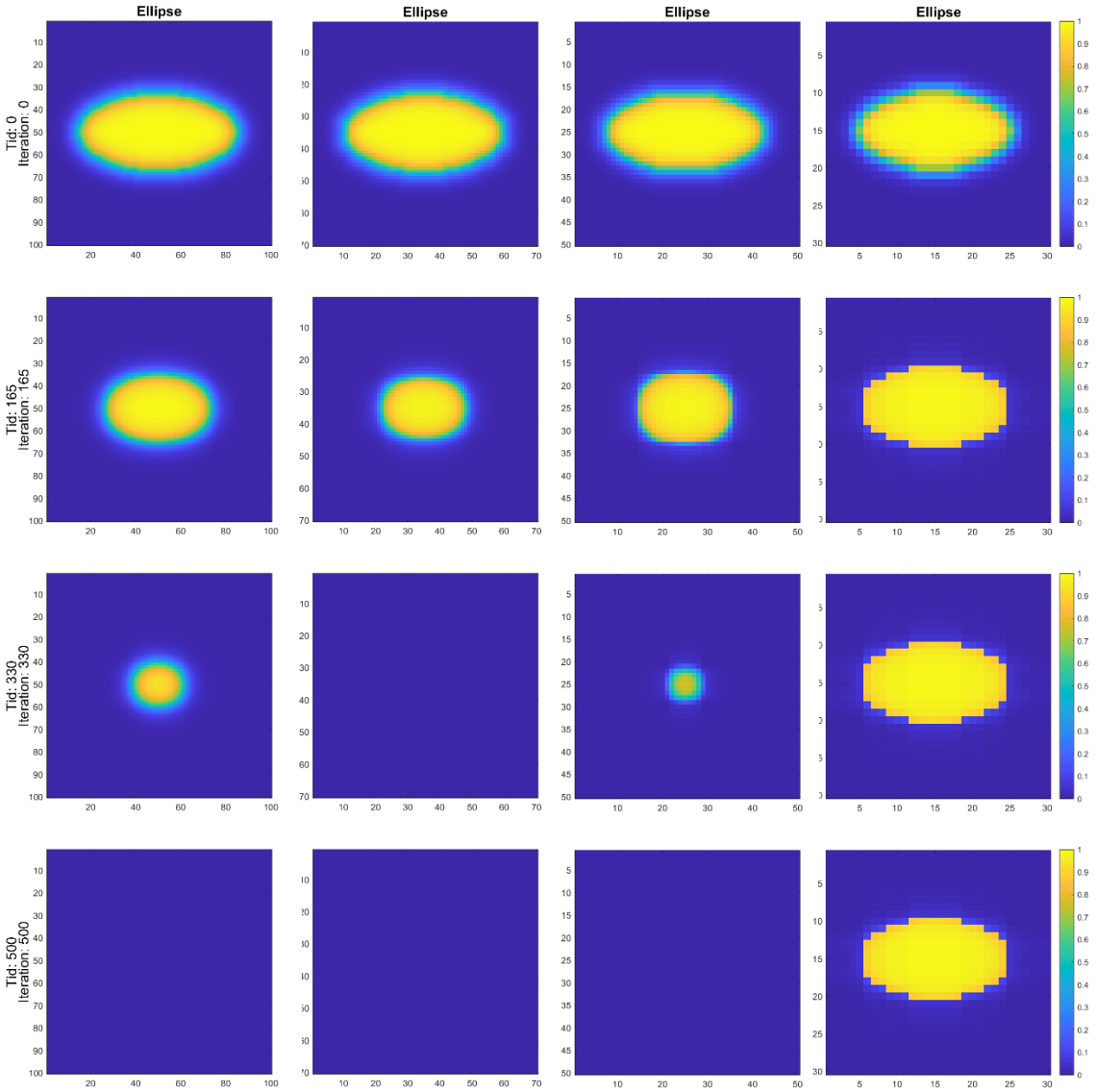


(a) Bredden av ellipsen vid Allen-Cahn simuleringar över tid för δ från Cahn-tal

(b) Bredden av ellipsen vid Allen-Cahn simuleringar över tid för konstant $\delta = 3$

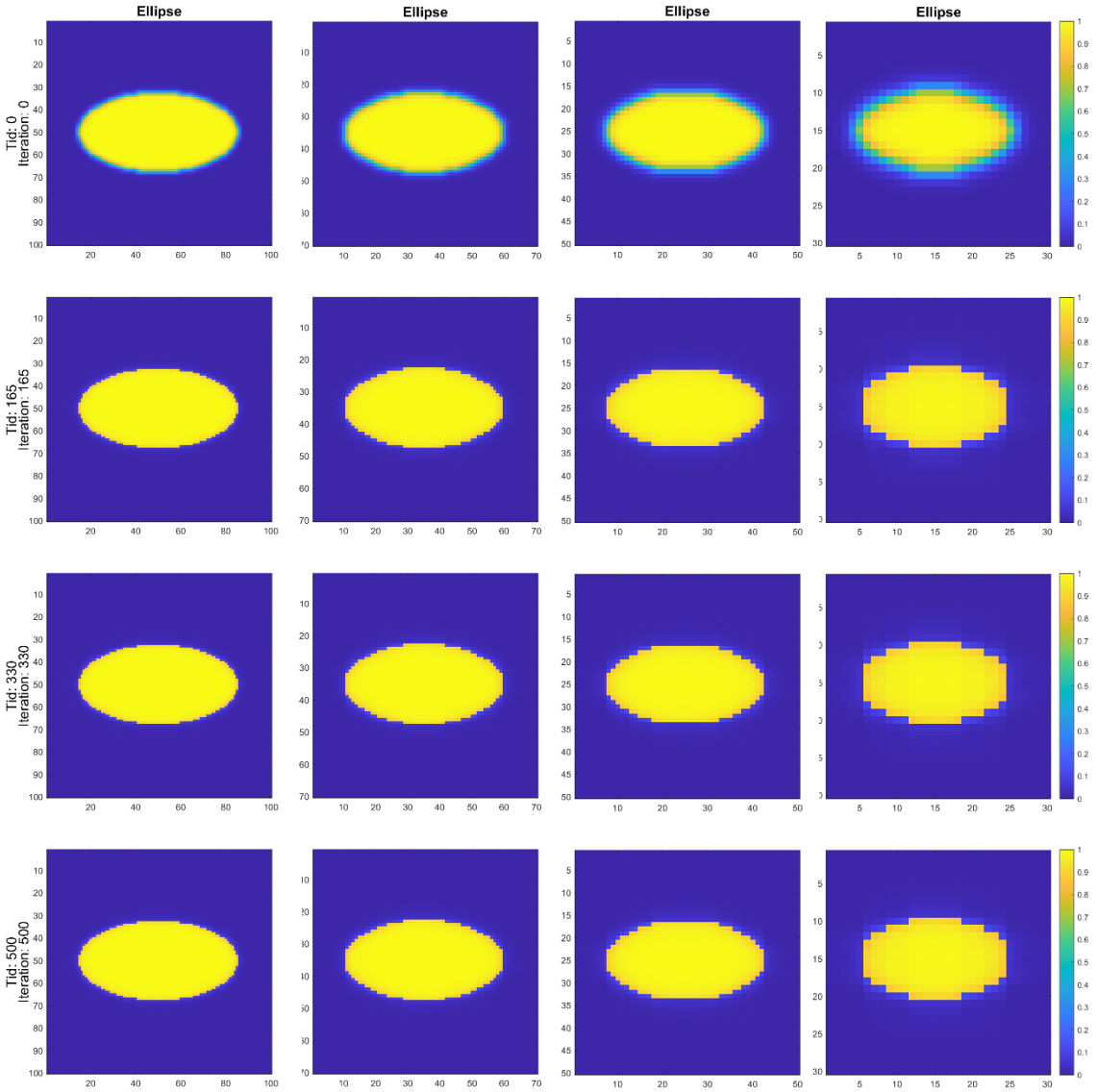
Vi kan observera i grafen C.1a att storleken på bredden minskar för upplösningar och tillslut blir noll för alla förutom 30 och 40. Enligt teorin så bör inte graferna bete sig som de gör och anledning till detta kan bero på att δ är för stort relativt upplösning. Vi kan notera att samma sak gäller för C.1b då graferna inte följer teorin. I figur C.1b kan vi notera att de olika upplösningarna fastnar vid olika storlek på bredden och stannar där. Observera att upplösningarna 50 och 60 är instabila eftersom de uppvisar snabba fluktuationer och varierande värden. På grund av detta tas de inte med i den slutliga analysen.

I graferna ovan kunde förändringen av ellipsens bredd observeras och i följande bilder kommer en simuleringarna illustreras. I följande 16 bilder kan 4 upplösningar observeras, upplösningarna går kolumnvis och iterationera radvis. Kolumnen längst till höger illustrerar upplösning 100 sen 70, 50, och 30 och iterationerna som visas radvis är 0, 165, 330, och 500. I figurerna kan vi observera hur ellipserna vid olika upplösningar krymper, vi kan se att ellipsen av upplösning 70 krymper snabbast till en punkt.



Figur C.2: Evolution för ellipsen med upplösning 100, 70, 50, och 30 från Allen-Cahn simuleringarna med Cn-värden.

I följande figurer är det samma upplösningar och iterationer som observeras som i figur 5.7 skillnaden är att figurerna representerar datan från graf C.1b. I följande figurer kan vi se att arean knappt förändras. Det vi kan observera är att det diffusa gränsskiktet förändras och blir skarpare för lägre upplösningar och framförallt för upplösning 30 där det diffusa gränsskiktet försvinner och figuren slutar utvecklas.

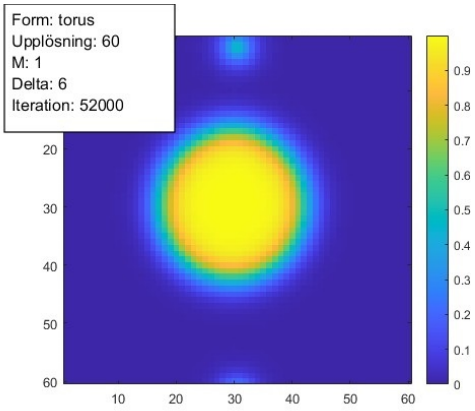


Figur C.3: Evolution för ellipsen med upplösning 100, 70, 50, och 30 från Allen-Cahn simuleringarna med $\delta = 3$.

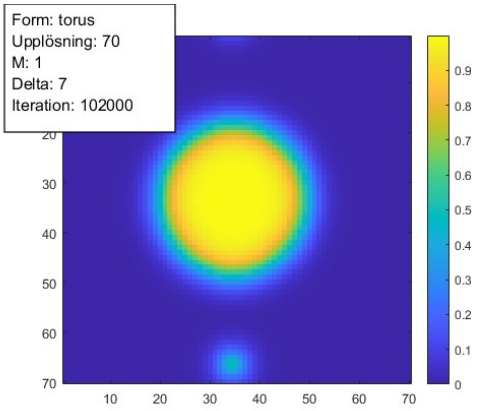
C.2 Anomalier av torusen vid Modifierade Allen-Cahn simuleringar

De följande två figurerna illustrerar torusens deformation med modifierade Allen-Cahn ekvationen och där Cahn-talet har använts för att välja δ. Figuren C.4a visar torusen i upplösning 60 efter 52 000 iterationer och med $\delta = 6$ medan C.4b visar torusen i upplösning 70 efter 102 000 iterationer $\delta = 6$.

Anledning till att detta är en avvikelse är på grund av ljusblåa cirklar uppstår över och under den gula cirkeln. Detta bidrar till en felkälla eftersom de ljusblåa cirklarna inkluderas vid beräkning av arean, se figur 5.6b.



(a) Torusen vid upplösning 60, vid iteration 52000 där δ erhölls från Cahn-talet.



(b) Torusen vid upplösningen 70, vid iteration 102000 där δ erhölls från Cahn-talet.

D Appendix 4 – källkod

D.1 Python källkod

D.1.1 Parallell körning av Gesualdo_2phase-simuleringar med olika inställningar

```
import itertools
import subprocess
import asyncio

# Define the properties
shapes = ["ellipse", "circle", "torus", "random"]
sizes = [30, 40, 50, 60, 70, 80, 90, 100]
taus = [3.5]
deltas = [3, 4, 5, 6, 7, 8, 9, 10]

# Generate all combinations
combinations = list(itertools.product(shapes, sizes, taus, deltas))

# Initialize a list to store unsuccessful commands
unsuccessful_commands = []

async def run_command(command):
    process = await asyncio.create_subprocess_shell(
        command,
        stdout=asyncio.subprocess.PIPE,
        stderr=asyncio.subprocess.STDOUT,
    )

    while True:
        line = await process.stdout.readline()
        if not line:
            break

        line_str = line.decode().strip()
        print(line_str)

        if "NaN detected" in line_str:
            process.terminate()
            await process.wait()
            raise RuntimeError(f"'NaN␣detected' found
                                in the output of command '{command}'")

    return await process.wait()

async def main():
    # Execute the command for each combination
    for combo in combinations:
        shape, size, tau, delta = combo
        settings_file =
            f"simulations/generated_xml/
                    gesualdo_{shape}_{size}_{tau}_{delta}.xml"
        command =
            f"mpiexec -n 4 Gesualdo_2phase.exe {settings_file}"

        try:
            await run_command(command)
        except (subprocess.CalledProcessError, RuntimeError) as e:
            print(f"An error occurred while
```

```
                    running the command '{command}'. Error: {e}")
                unsuccessful_commands.append((command, str(e)))
                # Continue with the next command
                continue

    # Write the unsuccessful commands and their error messages to a log file
    with open("error_log.txt", "w") as log_file:
        for command, error in unsuccessful_commands:
            log_file.write(f"Command: {command}\nError: {error}\n\n")

# Run the main coroutine
asyncio.run(main())
```

D.1.2 Uppdatering av XML-inställningsfiler för simuleringar baserat på olika kombinationer

```
import os
from itertools import product
from xml.etree.ElementTree import parse, tostring

def update_xml(xml_tree, shape, nx, ny, tau_AC, interfaceWidth):
    root = xml_tree.getroot()

    # Update Geometry
    root.find(".//size/nx").text = str(nx)
    root.find(".//size/ny").text = str(ny)

    # Update TwoPhaseFlow
    root.find(".//phaseField/tau_AC").text = str(tau_AC)
    root.find(".//phaseField/interfaceWidth").text = str(interfaceWidth)

    # Update initialCondition
    root.find(".//initialCondition/phaseField/filename").text =
                    f'simulations/init_files
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/{shape}_s{ny:03d}_d{interfaceWidth:01d}.vtk'

    # Update Output
    root.find(".//Output/outputDir").text =
                    f'simulations/results
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣/{shape}/{ny}/{tau_AC}/{interfaceWidth}'

    return root

def save_xml(root, filename):
    xml_declaration = '<?xml␣version="1.0"␣?>\n'
    with open(filename, 'w') as f:
        f.write(xml_declaration)
        f.write(tostring(root, encoding="unicode"))

if not os.path.exists('simulations/generated_xml'):
    os.makedirs('simulations/generated_xml')

shapes = ['ellipse']
nx_values = [30, 40, 50, 60, 70, 80, 90, 100]
ny_values = [30, 40, 50, 60, 70, 80, 90, 100]
tau_AC_values = [3.5]
interfaceWidth_values = [3, 4, 5, 6, 7, 8, 9, 10]

combinations = [(shape, nx, ny, tau_AC, interfaceWidth)
                for shape in shapes
                for nx, ny in zip(nx_values, ny_values)
```

```
                for tau_AC, interfaceWidth in product(tau_AC_values,
                                                      interfaceWidth_values)]

template_tree = parse('template.xml')

for i, (shape, nx, ny, tau_AC, interfaceWidth) in enumerate(combinations):
    updated_root = update_xml(template_tree, shape,
                              nx, ny, tau_AC, interfaceWidth)
    output_filename = f'simulations/generated_xml/
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣gesualdo_{shape}_{ny}_{tau_AC}_{interfaceWidth}.xml'
    save_xml(updated_root, output_filename)
```

D.1.3 Beräkning av skärningspunkter och tidsförskjutningar i Modifierade Allen-Cahn-simuleringar del 1

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

def line_intersections(x, y, hline):
    intersections = []
    x = x.to_numpy()
    y = y.to_numpy()
    for i in range(len(y) - 1):
        if (y[i] <= hline and y[i + 1] >= hline) or
            (y[i] >= hline and y[i + 1] <= hline):
            x_intersection = x[i] + (x[i + 1] - x[i]) *
                (hline - y[i]) / (y[i + 1] - y[i])
            intersections.append(x_intersection)
    return intersections

data = pd.read_csv("/work/Cn-test/
                   full_dataset_ellipse-35-(3-10).csv")
#data = pd.read_csv("/work/Delta-test/
#                    full_dataset_ellipse-35-3.csv")
#data = pd.read_csv("/work/Cn-test/
#                    full_dataset_ellipse-AC-Cn-35-(3-10).csv")

grouped_data = data.groupby("size")

# Define the colormap
cmap = plt.cm.get_cmap("viridis")

min_size = data["size"].min()
max_size = data["size"].max()

fig, ax = plt.subplots()
for name, group in grouped_data:
    if name not in []:#[30, 40, 50, 60]:
        normalized_size = (name - min_size)
        / (max_size - min_size)
        color = cmap(normalized_size)

        ax.plot(group["iteration"],
                group["distance"], c=color,
                linewidth=1, label=name)

ax.set_xlabel("Iteration",fontsize=15)
ax.set_ylabel("Diameter",fontsize=15)
#ax.set_xlim(0, 500)

```

```
for hline in np.arange(0.65, 0.53, -0.001):
    ax.axhline(hline, color="gray",
               linestyle="--", linewidth=0.5)

ax.legend(loc='center left',
          bbox_to_anchor=(1.0, 0.5),
          title="Upplösning",fontsize=15)
plt.show()

intersections_data = []
for name, group in grouped_data:
    if name not in []:#[30, 40, 50, 60]:
        for hline in np.arange(0.65, 0.53, -0.001):
            intersections =
            line_intersections(group["iteration"],
                               group["distance"], hline)
            for intersection in intersections:
                if not np.isnan(intersection):
                    intersections_data.append({
                        "Size": name, "Line":
                        hline, "Intersection":
                        intersection})
                    print(f"Size: {name},
                          Line: {hline:.2f},
                          Intersection: {intersection:.2f}")

# Save intersections data to a CSV file
intersections_df = pd.DataFrame(intersections_data)
intersections_df.to_csv("intersections.csv", index=False)
```

D.1.4 Beräkning av skärningspunkter och tidsförskjutningar i Modifierade Allen-Cahn-simuleringar del 2

```
import numpy as np
import scipy.stats as stats

def func(x, n, a):
    return a + n * np.log(x)

# Store n values in a list
n_values = []

fig, ax = plt.subplots()
for i, (line_name, group) in enumerate(df.groupby('Line')):
    x_data = group['Size']
    y_data = np.log(group['Intersection'])

    # Curve fitting
    popt, _ = curve_fit(func, x_data, y_data)
    n, a = popt
    n_values.append(n)

    # Plot the fitted curve
    x_range = np.linspace(min(x_data),
                          max(x_data), 100)
    ax.plot(x_range, func(x_range, n, a),
            label=f"hlinje {i+1}: n={n:.2f}")

    # Scatter plot
    ax.scatter(x_data, y_data, marker='o', alpha=0.5)

```

```
ax.set_xlabel('Upplösning',fontsize=12)
ax.set_ylabel('log(Genomskärnings punkt)',fontsize=12)
ax.legend(loc='center left', bbox_to_anchor=(1.0, 0.5),fontsize=12)
ax.set_title('log(Genomskärnings punkt) =
          n*log(Upplösning) per Linje',fontsize=12)
plt.show()

# Calculate and print the mean and
# standard deviation of n values
mean_n = np.mean(n_values)
std_n = np.std(n_values)

# Calculate the confidence interval
alpha = 0.95  # for a 95% confidence interval
degrees_of_freedom = len(n_values) - 1
confidence_interval =
    stats.t.interval(alpha,
                     degrees_of_freedom,
                     loc=mean_n, scale=std_n)

print(f"Mean of n values: {mean_n:.2f}")
print(f"Standard deviation of n values: {std_n:.2f}")
print(f"{int(alpha*100)}% Confidence interval:
      {confidence_interval}")
```

D.1.5 Beräkning av medelvärde, standardavvikelse och konfidensintervall för förändring av ellipsbredd vid olika iterationer

```
import pandas as pd
import numpy as np
import scipy.stats as stats

def main():
    file_path = "/work/diamiter_100_
        dataset_ellipse-35-10.csv"
    df = pd.read_csv(file_path)

    delta_values = [3, 3.5, 4, 4.5, 5, 5.5, 6,
                    6.5, 7, 7.5, 8, 8.5, 9, 9.5, 10]

    print("Enter two different
            iterations (between 0 and 200500):")
    iteration1 = 15 #int(input("Iteration 1: "))
    iteration2 = 60 #int(input("Iteration 2: "))

    for delta in delta_values:
        print(f"\nProcessing data
              for delta: {delta}")

        # Filter the dataframe based on
        #  the current delta value
        df_filtered = df[df['delta'] == delta]

        ratios = []
        for i in range(iteration1, iteration2, 5):
            distance_x1, distance_y1 =
                get_distance(df_filtered, i)
            distance_x2, distance_y2 =
                get_distance(df_filtered, i + 1)
            ratio = (distance_y2 - distance_y1)
                / (distance_x2 - distance_x1)
```

```
            ratios.append(ratio)

        mean = np.mean(ratios)
        sd = np.std(ratios)

        # Calculate the confidence interval with a
        # confidence level of 95%
        ci = stats.t.interval(0.95,
                              len(ratios)-1, loc=mean,
                              scale=stats.sem(ratios))

        print(f"Mean of ratios for delta {delta}: {mean}")
        print(f"Standard deviation of ratios for delta {delta}: {sd}")
        print(f"95% confidence interval for delta {delta}: {ci}")

def get_distance(df, iteration):
    row = df[df['iteration'] == iteration].iloc[0]
    distance_x = row['distance_x']
    distance_y = row['distance_y']
    return distance_x, distance_y

if __name__ == "__main__":
    main()
```

D.2 MATLAB källkod

D.2.1 Generering av initiala fält för tvåfasflödessimuleringar med olika geometrier

```
% read in the image
%img = imread("image.jpg");

% create grid
Nx=100;  % number of grid points in x and y (must be same as in input file)
Ny=Nx;
delta = 6; % interface width
x=1:Nx;
y=1:Ny;
[X,Y,Z]=ndgrid(x,y,1:3);  % grid coordinate matrices

% scale the image down to size 100 by 100
%img = imresize(img, [Nx Ny]);

% convert the image to black and white
%img_bw = im2bw(img);

% create a matrix containing only 1 and 0 for black and white
%img_matrix = double(img_bw);

% invert the 1 and 0 in the matrix
%img_matrix = 1 - img_matrix;

% add an extra dimension to the matrix that is a copy of the
% first layer three times
%img_matrix = repmat(img_matrix, [1 1 3]);

% create shape (with 1 inside, 0 outside)
R = Ny/4;  % radius of disc
%A=((X-Nx/2).^2 + (Y-Ny/2).^2 <= R^2) > ((X-Nx/2).^2 + (Y-Ny/2).^2 <= (R-15)^2);
% indicator function for a donught

```

```
33
34 %A=(2*(X-Nx/2).^2 + 0.5*(Y-Ny/2).^2 <= R^2) > (2*(X-Nx/2).^2 + 0.5*(Y-Ny/2).^2
35 %                        <= (R-R/2)^2);  % indicator function for a donught wide
36
37 %A=((X-Nx/2).^2 + (Y-Nx/2).^2 <= R^2);  % indicator function for a disc
38
39
40 A=(2*(X-Nx/2).^2 + 0.5*(Y-Ny/2).^2 <= R^2);  % indicator function for a ELIPSE
41
42 %A=img_matrix;
43
44
45 % Create a 100x100 matrix of random 1s and 0s
46 %M = randi([0, 1], Nx, Ny);
47
48 % Scale the M matrix up to Nx1 by Ny1
49 %M_resized = imresize(M, [Nx, Ny], 'nearest');
50
51 % Convert the matrix into a 100x100x3 matrix where
52 % each matrix is a copy of each other
53 %A = repmat(M_resized, [1, 1, 3]);
54
55
56 % Create the phi profile
57 SD=bwdist(A) - bwdist(1-A);  % signed distance function
58 SDs=smooth3(SD,'box');  % smooth to avoid jump at edge
59 F=0.5*(1-tanh(2*SDs/delta));  % stationary profile function
60 % Note: this method is not perfect, i.e. will not create the exact profile.
61 % For a disc it could be done exactly, but this method could be applied
62 % for general shapes.
63
64 % show data
65 figure(1)
66 clf
67 imagesc(F(:,:,2))
68 colorbar
69 axis equal tight
70
71
72 % write VTK file for input to Gesualdo
73 filename = sprintf(['init_generation/random_s%03d_d%01d.vtk'], Ny, delta);
74 %filename = 'disc_100_r30.vtk';
75 writeVTK(F,filename,'binary');
76
77
78
79 %% 3D view
80
81 % Create x and y coordinate matrices using meshgrid
82 [X,Y] = meshgrid(1:Nx, 1:Ny);
83
84 % Create z coordinate matrix
85 Z = F(:,:,2);
86
87 % Generate 3D surface plot
88 figure(2 )
89 surf(X,Y,Z, LineStyle="none", EdgeColor="black")
90
91 % Add labels and title
92 xlabel('X')
93 ylabel('Y')
```

```
zlabel('Z')
title('3D Surface Plot')
%% Plot the contour phi=0.5
figure(3)
clf
contour(F(:,:,2),[0.5 0.5],'black')
axis equal tight

%% Plot the function values along center line
figure(4)
clf
plot(F(:,Ny/2,2),'black')
```

D.2.2 Skapa sammanslagna dataset från VTK-filer och beräkna geometriska egenskaper

```
% Define the possible values for each variable
shapes = {'circle', 'torus', 'random', 'ellipse'};
sizes = [30, 40, 50, 60, 70, 80, 90, 100];
taus = [3.5];
deltas = [3, 4, 5, 6, 7, 8, 9, 10];

% Loop over all the possible combinations of variables
for i = 1:length(shapes)
    % Define the output file name
    output_file_name = sprintf('datasets/full_dataset_%s-35-(3-10).csv', ...
        shapes{i});

    % Open the output file in write mode and create a CSV writer
    file = fopen(output_file_name, 'w');
    fprintf(file, 'shape,size,tau,delta,iteration,area,mass_loss,distance\n');
    shape = shapes{i};

    if (shape == "ellipse")
        iter= 1;
    else
        iter = 5;
    end

    for j = 1:length(sizes)
        size = sizes(j);
        for k = 1:length(taus)
            % Construct the folder path
            folder_path = fullfile('simulations', 'results', shapes{i}, ...
                num2str(sizes(j)), num2str(taus(k)), num2str(deltas(j)));

            % Loop over the iterations
            for m = 0:iter:205000
                % Construct the file name
                file_name = sprintf('vtk_twophase_%08d.vtk', m);

                % Construct the full file path
                file_path = fullfile(folder_path, file_name);


                % Check if the file exists
                if exist(file_path, 'file') == 2
                    % Read the volume
                    A = vtk_read_volume(file_path);
                    Area = nnz(A >= 0.5);

```

```
46                      % Gain inital Area
47                      if(m == 0)
48                          Area_initial = nnz(A >= 0.5);
49                      end
50
51                      % Mass loss from initial to current iteration
52                      mass_loss = ( (Area - Area_initial) / Area_initial ) * 100;
53
54                      % Only gain distance if its an ellipse
55                      if ((shape == "ellipse") || (shape == "circle") ...
56                              || (shape == "torus"))
57                          % Gain horizontal line distance
58
59                          x_indices = find((A(size/2, :, 2) >= 0.5));
60                          x1 = x_indices(1);
61                          x2 = x_indices(end);
62
63                          % Linear interpolation to find the more accurate
64                          % point where the value is 0.5
65                          if x1 > 1
66                              x0_left = x1 - 1;
67                              y0_left = A(size/2, x0_left, 2);
68                              y1_left = A(size/2, x1, 2);
69                              x1_interpolated = x0_left + (0.5 - y0_left) *
70                                      (x1 - x0_left) / (y1_left - y0_left);
71                          else
72                              x1_interpolated = x1;
73                          end
74
75                          if x2 < size
76                              x0_right = x2;
77                              y0_right = A(size/2, x0_right, 2);
78                              x1_right = x2 + 1;
79                              y1_right = A(size/2, x1_right, 2);
80                              x2_interpolated = x0_right + (0.5 - y0_right)
81                                              * (x1_right - x0_right) /
82                                              (y1_right - y0_right);
83                          else
84                              x2_interpolated = x2;
85                          end
86
87                          % Extract contour points for size1 and size2
88                          contour_points = [x1_interpolated,
89                              0; x2_interpolated, 0];
90
91                          % Scale contour points of size2 to
92                          % match the scale of size1
93                          contour_points_rescaled = ((contour_points - 1) ...
94                              / (size - 1));
95
96                          % Calculate distance
97                          d = pdist(contour_points_rescaled, 'euclidean');
98                       else
99                          d = NaN;
100                      end
101
102
103                     % Write the row to the CSV file
104                     fprintf(file, '%s,%d,%g,%d,%d,%g,%g,%g\n', shapes{i}, ...
105                         sizes(j), taus(k), deltas(j), m, Area, mass_loss, d);
106                 else
```

```
                    % Write a row with NaN for the Area
                    fprintf(file, '%s,%d,%g,%d,%d,NaN\n', shapes{i}, ...
                        sizes(j), taus(k), deltas(j), m);
                end
            end
        end
    end
    % Close the output file
    fclose(file);
end
```

D.2.3 Skapa sammanslagna dataset från VTK-filer och beräkna bredden och höjden för ellipsen

```
% Define the possible values for each variable
shapes = {'ellipse'};
sizes = [100,100,100,100,100,100,100,100,100,100,100,100,100
        ,100,100,100,100,100,100];
taus = [3.5];
deltas = [1, 1.5 , 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5,
            7, 7.5, 8, 8.5, 9, 9.5, 10];

% Loop over all the possible combinations of variables
for i = 1:length(shapes)

    % Define the output file name
    output_file_name = sprintf('datasets/diamiter_100_dataset
                                    _%s-35-10.csv', shapes{i});

    % Open the output file in write mode and create a CSV writer
    file = fopen(output_file_name, 'w');
    fprintf(file,'shape,size,tau,delta,iteration, distance_x,
                         distance_y\n');

    shape = shapes{i};
    if (shape == "ellipse")
        iter= 1;
    else
        iter = 5;
    end
    for j = 1:length(sizes)
        size = sizes(j);
        for k = 1:length(taus)
            % Loop over the iterations
            for m = 0:1:200
                % Construct the file name
                file_path = sprintf('G:/Kandidat/simulations/
                results/%s/%d/%g/%g/vtk_twophase_%08d.vtk',
                shape, size, taus(k), deltas(j), m);

                % Check if the file exists
                if exist(file_path, 'file') == 2
                    % Read the volume
                    A = vtk_read_volume(file_path);
                    Area = nnz(A >= 0.5);


                    % Only gain distance if its an ellipse
                    if ((shape == "ellipse") ||
                        (shape == "circle") ||
                        (shape == "torus"))
```

```
% Gain horizontal line distance

x_indices = find((A(size/2, :, 2) >= 0.5));
x1 = x_indices(1);
x2 = x_indices(end);

y_indices = find((A(:, size/2, 2) >= 0.5));
y1 = y_indices(1);
y2 = y_indices(end);

% Linear interpolation to find the more
    accurate point where the value is 0.5
if x1 > 1
    x0_left = x1 - 1;
    y0_left = A(size/2, x0_left, 2);
    y1_left = A(size/2, x1, 2);
    x1_interpolated = x0_left +
        (0.5 - y0_left) * (x1 - x0_left)
            / (y1_left - y0_left);
else
    x1_interpolated = x1;
end

if x2 < size
    x0_right = x2;
    y0_right = A(size/2, x0_right, 2);
    x1_right = x2 + 1;
    y1_right = A(size/2, x1_right, 2);
    x2_interpolated = x0_right +
        (0.5 - y0_right) * (x1_right - x0_right)
            / (y1_right - y0_right);
else
    x2_interpolated = x2;
end

% Extract contour points for size1 and size2
contour_points = [x1_interpolated,
    0; x2_interpolated, 0];

% Scale contour points of size2
    to match the scale of size1
contour_points_rescaled =
    ((contour_points - 1) / (size - 1));
% Calculate distance
d_x = pdist(contour_points_rescaled, 'euclidean');


% Linear interpolation to find the more
    accurate point where the value is 0.5
if y1 > 1
    x0_left = y1 - 1;
    y0_left = A(x0_left, size/2, 2);
    y1_left = A(y1, size/2, 2);
    y1_interpolated = x0_left +
        (0.5 - y0_left) * (y1 - x0_left)
            / (y1_left - y0_left);
else
    y1_interpolated = y1;
end

if y2 < size
```

```
                            x0_right = y2;
                            y0_right = A(x0_right, size/2, 2);
                            x1_right = y2 + 1;
                            y1_right = A(x1_right, size/2, 2);
                            y2_interpolated = x0_right +
                                (0.5 - y0_right) * (x1_right - x0_right)
                                    / (y1_right - y0_right);
                        else
                            y2_interpolated = y2;
                        end

                        % Extract contour points for
                            size1 and size2
                        contour_points = [y1_interpolated,
                            0; y2_interpolated, 0];

                        % Scale contour points of size2
                            to match the scale of size1
                        contour_points_rescaled =
                            ((contour_points - 1) / (size - 1));

                        % Calculate distance
                        d_y = pdist(contour_points_rescaled, 'euclidean');
                    else
                        d = NaN;
                    end


                    % Write the row to the CSV file
                    %fprintf('%s,%d,%g,%d,%d,dx:%f,dy:%f\n', shapes{i},
                        sizes(j), taus(k), deltas(j), m, d_x, d_y);

                    % Write the row to the CSV file
                    fprintf(file, '%s,%d,%g,%g,%d,%f,%f\n', shapes{i},
                        sizes(j), taus(k), deltas(j), m, d_x, d_y);
                else
                    % Write a row with NaN for the Area
                    fprintf('%s,%d,%g,%g,%d,NaN\n', shapes{i},
                        sizes(j), taus(k), deltas(j), m);
                end
            end
        end
    end
    % Close the output file
    fclose(file);
end
```

D.2.4 Generering av simuleringsbilder för olika parametrar

```
I_30 = [0, 500, 1000, 1600];
I_40 = [0, 1580, 3160, 5055];
I_50 = [0, 3860, 7715, 12345];
I_60 = [0, 8000, 16000, 25600];
I_70 = [0, 14820, 29640, 47430];
I_80 = [0, 25280, 50570, 80910];
I_90 = [0, 40500, 81000, 129600];
I_100 = [0, 61730, 123455, 197530];

I = {I_30, I_40, I_50, I_60, I_70, I_80, I_90, I_100};
% Parameter arrays
shapes = {'circle', 'ellipse', 'torus', 'random'};
```

```
sizes = [30, 40, 50, 60, 70, 80, 90, 100];
taus = [3.5];
deltas = [3, 4, 5, 6, 7, 8, 9, 10];

% Loop through all the parameter combinations
for s = 1:length(shapes)
    shape = shapes{s};
    for j = 1:length(sizes)
        size = sizes(j);
        for tau = taus
            delta = deltas(j);
            % Loop through the files and read each one
            for i = 1:4

                % Read the data from the file
                filename_A = sprintf(['simulations/results/%s/%d/%g/%d/' ...
                    'vtk_twophase_%08d.vtk'], shape, size, tau, delta, ...
                    I{j}(i));
                A = vtk_read_volume(filename_A);

                % Create the figure and write it to a image
                figure(7)
                clf
                imagesc(A(:,:,2))
                colorbar
                colorbar('off')
                axis equal tight
                if (shape == "random")
                    colorbar
                end
                str3 = ['\fontsize{16}' upper(shape(1)) shape(2:end)];
                if (I{j}(i) == 00000000)
                    title(str3)
                end

                % Find the limits of the current axis
                ax = gca;
                x_limits = ax.XLim;
                y_limits = ax.YLim;

                % Add vertical text outside the plot area
                text_iter_str = ['Iteration: ' num2str(I{j}(i))];
                text_time_str = ['Tid: ' num2str(I_30(i))];
                if (shape == "circle")
                    text(x_limits(1) - 10.5, (y_limits(1) + y_limits(2)) ...
                        / 2, text_iter_str, 'Rotation', 90, ...
                        'HorizontalAlignment', 'center', 'FontSize', 16);
                    text(x_limits(1) - 16, (y_limits(1) + y_limits(2)) ...
                        / 2, text_time_str, 'Rotation', 90, ...
                        'HorizontalAlignment', 'center', 'FontSize', 16);
                end
                iteration = (I{j}(i) / ((30/size)^4));
                % Add text box with shape, size, tau, and delta information
                %info_str = sprintf('Shape: %s\nSize: %d\nTau:
                %g\nDelta: %d\nIteration: %d\nTime: %d', shape, size,
                % tau, delta, I{j}(i), I_30(i));
                %annotation('textbox', [0.05 0.7 0.3 0.31],
                % 'String', info_str, 'FontSize', 12, 'BackgroundColor',
                % 'w', 'EdgeColor', 'k', 'LineWidth', 1);

                % Save figure as a PNG file with the
```

```
                % correct file name format
                filename = sprintf(['results_images/%s_%d_%g_%d_%08d' ...
                    '.jpg'], shape, size, tau, delta, I{j}(i));
                % Create necessary directories if they don't exist
                [filepath, ~, ~] = fileparts(filename);
                if ~exist(filepath, 'dir')
                    mkdir(filepath);
                end

                print(filename, '-dpng', '-noui');
            end

        end
    end
end
```

D.2.5 Skapa videor av tvåfasflödessimuleringar med olika geometrier

```
%% Write out Video
% Set the iteration count and the total number of files to be read
iter = 100;
total = 10000/100;

% Set the frame rate for the video
frame_rate = 30;

% Parameter arrays
shapes = {'circle', 'ellipse', 'torus', 'random'};
sizes = [32, 64, 128, 256];
taus = [3.5, 4, 5];
deltas = [3, 5, 7];

% Create the 'results_videos' directory if it doesn't exist
if ~exist('results_videos', 'dir')
    mkdir('results_videos');
end

% Loop through all the parameter combinations
for s = 1:length(shapes)
    shape = shapes{s};
    for size = sizes
        for tau = taus
            for delta = deltas
                % Check if all files exist in the folder
                all_files_exist = true;
                for i = 0:total-1
                    filename_A = sprintf(['simulations/' ...
                        'results/%s/%d/%g/%d/vtk_twophase_%08d.vtk'], ...
                        shape, size, tau, delta, i*iter);
                    if ~exist(filename_A, 'file')
                        all_files_exist = false;
                        break;
                    end
                end

                if ~all_files_exist
                    fprintf(['Skipping folder for %s, size: %d,' ...
                        ' tau: %g, delta: %d (not enough files)\n'], ...
                        shape, size, tau, delta);
                    continue;
                end
```

```
44
45                  % Create a VideoWriter object to write the video file
46                  video_name = sprintf('results_videos/%s_%d_%g_%d.avi', ...
47                      shape, size, tau, delta);
48                  video_writer = VideoWriter(video_name, 'Motion JPEG AVI');
49                  video_writer.FrameRate = frame_rate;
50                  open(video_writer);
51
52                  % Loop through the files and read each one
53                  for i = 0:total-1
54                      % Read the data from the file
55                      filename_A = sprintf(['simulations/' ...
56                          'results/%s/%d/%g/%d/vtk_twophase_%08d.vtk'], ...
57                          shape, size, tau, delta, i*iter);
58                      A = vtk_read_volume(filename_A);
59
60                      % Create the figure and write it to the video file
61                      figure(7)
62                      clf
63                      imagesc(A(:,:,2))
64                      colorbar
65                      axis equal tight
66
67                      % Write the current frame to the video file
68                      writeVideo(video_writer, getframe(gcf));
69                  end
70
71                  % Close the video file
72                  close(video_writer);
73                  fprintf(['Done creating the video for %s, ' ...
74                      'size: %d, tau: %g, delta: %d\n'], shape, size, ...
75                      tau, delta);
76              end
77          end
78      end
79  end
```

D.2.6 Skapa en video med alla kombinationer av tvåfasflödessimuleringar

```
1  % Set the iteration count and the total number of files to be read
2  iter = 1000;
3  total = 100000/iter;
4
5  % Parameter arrays
6  shapes = {'random'};
7  sizes = [32, 64, 128, 256];
8  taus = [3.5, 4, 5];
9  deltas = [3, 5, 7];
10
11 % Number of black screen frames to display between combinations
12 break_frames = 10;
13
14 % Create a video writer for the entire video
15 video_filename = 'results_videos/all_combinations.avi';
16 [video_filepath, ~, ~] = fileparts(video_filename);
17 if ~exist(video_filepath, 'dir')
18     mkdir(video_filepath);
19 end
20
21 v = VideoWriter(video_filename);
```

```
open(v);

% Loop through all the parameter combinations
for s = 1:length(shapes)
    shape = shapes{s};
    for size = sizes
        for tau = taus
            for delta = deltas
                % Add black screen frames with upcoming
                % combination information
                for b = 1:break_frames
                    % Create a black screen
                    figure(7)
                    clf
                    black_screen = zeros(256, 256);
                    imagesc(black_screen)
                    colorbar
                    axis equal tight

                    % Add text box with upcoming shape,
                    % size, tau, and delta information
                    info_str = sprintf(['Upcoming:\nShape: ' ...
                        '%s\nSize: %d\nTau: %g\nDelta: %d'], ...
                        shape, size, tau, delta);
                    annotation('textbox', [0.05 0.7 0.3 0.3], ...
                        'String', info_str, 'FontSize', 12, ...
                        'BackgroundColor', 'w', 'EdgeColor', 'k', ...
                        'LineWidth', 1);

                    % Write the current frame to the video
                    frame = getframe(gcf);
                    writeVideo(v, frame);
                end

                % Loop through the files and read each one
                for i = 0:total-1
                    % Read the data from the file
                    filename_A = sprintf(['simulations/results/%s/' ...
                        '%d/%g/%d/vtk_twophase_%08d.vtk'], shape, ...
                        size, tau, delta, i*iter);
                    A = vtk_read_volume(filename_A);

                    % Create the figure and write it to an image
                    figure(7)
                    clf
                    imagesc(A(:,:,2))
                    colorbar
                    axis equal tight

                    % Add text box with shape, size,
                    % tau, and delta information
                    info_str = sprintf(['Shape: %s\nSize: %d\nTau: ' ...
                        '%g\nDelta: %d\nIteration: %d'], shape, size, ...
                        tau, delta, i*iter);
                    annotation('textbox', [0.05 0.7 0.3 0.3], ...
                        'String', info_str, 'FontSize', 12, ...
                        'BackgroundColor', 'w', 'EdgeColor', 'k', ...
                        'LineWidth', 1);

                    % Write the current frame to the video
                    frame = getframe(gcf);
```

```
                    writeVideo(v, frame);
                end
            end
        end
    end
end

% Close the video writer
close(v);

% Display the video
disp('Playing video with all combinations...');
implay(video_filename);
```