CHALMERS

GÖTEBORGS UNIVERSITET

Stokastisk modellering av glioblastom

Stochastic modeling of glioblastoma

Kandidatarbete inom civilingenjörsutbildningen vid Chalmers

Isak Hulthe
Lucas Kuszli
Max Brodén
My Irenheim Jarlhed
Patrik Franzén Dennis
Viggo Segersten

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2025

Stokastisk modellering av glioblastom

Kandidatarbete i matematik inom civilingenjörsprogrammet Globala System vid Chalmers
My Irenheim Jarlhed, Viggo Segersten

Kandidatarbete i matematik inom civilingenjörsprogrammet Bioteknik vid Chalmers

Max Brodén

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk fysik vid Chalmers
Isak Hulthe

Kandidatarbete i matematik inom civilingenjörsprogrammet Teknisk matematik vid Chalmers
Lucas Kuszli, Patrik Franzén Dennis

Handledare: Adam Malik
Philip Gerlee

Institutionen för Matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2025

Förord

Bidraget till korrekturläsning och redaktionellt ansvar tilldelas samtliga medverkande. Textstrukturs ansvar låg främst hos Patrik och Isak. En loggbok har förts över alla gruppmedlemmars enskilda arbete under arbetet, vilken också anger gemensamma prestationer och möten. Nedan följer individuella bidrag från alla i kandidatgruppen:

Max tog ansvaret att assistera My i anteckningar på möten. Informationsinhämtning och inläsning på de mer biologiska teorin låg främst på Max, då anpassning krävdes i teorin, baserat på vilka resultat som faktiskt valdes att presenteras. Max skrev dessutom en mindre del i modellbyggandet i python. Han såg också till att de olika källhänvisningar och referenser kom på plats.

Patrik var helt ansvarig för såväl utvecklingen som implementeringen av modellen och P*-algoritmen som används i slutgiltiga arbetet. Detta gäller hela processen från initiering av tumören till fenotypval och kostnadsfunktion. Patrik var också ansvarig för koden som möjliggör parameter- simulering och generering samt simuleringen av tumören i stor, med en tillhörande grafiskt användarsnitt skriven i programmeringsspråket Go. Vidare skrevs kod för att samla data för såväl simulerad som experimentell data. All kod i detta arbete samlades i GitHub som ansvarades av Patrik, där kodgranskning och versionshantering gjordes. Patrik hjälpte även till med strukturen och skrivande av slutsatsen, som skulle följa och besvara frågeställningen. Tillsammans med My skrevs diskussionen om kluster i 7.1. Grammatik och strukturen av rapporten i sin helhet gjordes också av Patrik, där det bestod främst av korrekturläsning och textredigering. Slutligen skrevs och ansvarades helt följande avsnitten av Patrik: 3, 4.2, 4.3, 5.2, 5.4, 7.3, 8.

Till Isak attribueras första iterationen av modellen i Python tillsammans med Lucas, där ansvaret främst var bygget av ”cell-cell-repulsion” mekanismen, klassutbygge i kodbasen och parallellisering av simuleringsprocessen. Isak tillägnas också idé bakom och implementering av visualiseringsverktygen för modelldata som ska visas, som användes i samtliga iterationer av modellen. I den slutgiltiga simuleringsprocessen med P*-modellen tillhandahöll och ansvarade Isak för gemensamt arbetsblad, där ”simulerings batches” valdes av samtliga medverkande. Korrelationstesterna och figurerna i rapporten gjordes av Isak med hjälp av insamlad data från Lucas. Slutligen skrevs avsnitten: inledning 1, bakgrund 2 (tillsammans med Viggo) och korrelationsavsnitten 5.5, 6.4, samt 7.1 av Isak.

Till en början av projektet lästes tidigare studier om vad som har gjorts på liknande områden detta gjordes av samtliga medlemmar, däribland My. Hon har även haft huvudansvar för anteckningar på möten. Utöver det har hon haft ansvar för allt som haft med fraktaldimension att göra, både utvecklande av kod för att ta fram måtten och också skrivit delarna i texten som rör den, 4.4, 5.1 (med Viggo), 7.2. Samt 6.3 om subtypes.

Lucas stod tillsammans med Isak för en första version av go or grow modellen. Där huvudfokus för Lucas var grundmodellerande och utbyggnad av diverse datapipelines för att kunna simulera i parameterrymden och sedan använda detta i framtida analys. Efter introduktionen av P*, lades allt fokus istället på datagenerering och parameterestimeringarna. Lucas står för utvecklandet av koden av den korsvalidering och parameteruppskattning som finns beskriven i 6.2 och6.3. Lucas sammanställde den stora mängd syntetisk data som gruppen gemensamt genererade. Avsnitt skrivna till del, eller helt av Lucas: 4.3, 5.4, 6.2, 6.3, och 7.1. Men anvsvarar också för hela appendix A.

Viggo har haft stort ansvar för bildhanteringen och dess algoritm B samt ”pipeline” för att extrahera data från såväl *in vitro* data som simulerad data. I rapporten har Viggo bidragit främst med populärvetenskaplig presentation, bakgrund 2, resultat 6.1 och bildhantering 5.1, samhälleliga samt etiska aspekter 8 och första utvärdering av modellen, 5.3 i metoden.

Populärvetenskaplig presentation

Glioblastom är en av de mest aggressiva och dödliga formerna av hjärntumör. Den är snabbväxande och sprider sig på oförutsägbara sätt vilket gör att den är svår att prognostisera och behandla. Trots medicinska framsteg under de senaste decennierna har inte överlevnaden för patienter förbättrats påtagligt. Den genomsnittliga överlevnaden, efter prognos, är omkring ett år. För att göra verkliga framsteg krävs det att man utforskar nya verktyg och ett av de mest lovande verktygen är just matematisk modellering.

I detta kandidatarbete har vi tagit fram en matematisk modell som försöker efterlikna hur glioblastomtumörer växer genom att beskriva hur enskilda tumörceller rör sig, delar sig och reagerar på sin omgivning. Genom att beskriva hur tumörcellerna rör sig kan vi testa hypoteser, upptäcka mönster och analysera mekanismer som annars skulle vara svåra eller omöjliga att studera på traditionellt sätt i ett laboratorium. Modellering och simulering håller på att ta en allt viktigare plats inom medicinsk forskning, inte nödvändigtvis som ersättning av experiment, utan som ett komplement som kan effektivisera, fördjupa och ersätta vissa steg i den experimentella processen.

Vår modell bygger på en grundläggande men kraftfull idé: istället för att försöka exakt beskriva varje enskild cell, eller exakt modellera en tumörs tillväxt med matematiska ekvationer, så modellerar vi sannolikheter och tendenser. Biologiska system är inte deterministiska generellt sett, utan karaktäriseras av slumpmässighet. I vårt fall innebär det att celler inte kommer att bete sig likadant ens under identiska förhållanden. Genom att inkludera slumpmässighet i vår modell är tanken att få en mer realistisk och flexibel beskrivning av tumörens utveckling.

En särskilt intressant hypotes vi vill kunna påvisa hos glioblastomtumören är den så kallade go or grow- mekanismen. Det är en egenskap som innebär att tumörceller växlar mellan att röra sig (go) och att dela sig (grow). Denna dynamik verkar vara central för tumörens spridning. Vår modell tar också hänsyn till hur celler rör sig bort från andra celler (”cell-cell-repulsion”).

En modell blir särskilt värdefull om den kan jämföras med verklig data. Vi har därför fått tillhandahållet experimentdata från Uppsala universitet, där forskare odlat glioblastomceller från ett antal olika patienter. Det innebär att vi får en inblick i hur olika former av glioblastom ter sig, vilket är viktigt eftersom tumörens beteende kan variera mycket från patient till patient. Dessa experimentdata består av bilder på glioblastomtumörer som växer i ett 3D tillväxtmedium, tillsammans med relevant information om patienten som cellerna är tagna från. Med hjälp av bildbehandlingsmetoder kan vi extrahera information om tumörernas attribut och hur de ändras över tid. Vi kollar då på hur tumörens inre del, eller kärna, samt den mer spridda invasiva delen växer. Tillsammans med hur tätheten av celler vid olika avstånd från mitten av tumören ändras över tid, ger detta en beskrivning av tillväxten som går att jämföra med våra simulerade tumörer.

Nästa steg är att justera vår modell så att den stämmer så bra som möjligt med verkligheten, alltså experimentdata. Det gör vi med hjälp av en anpassad statistisk parameterskattning. Med dessa verktyg kan vi uppskatta modellens parametrar, alltså de värden som styr tumörcellens beteende. På så sätt kan vi också säga något om vilka faktorer som är mest avgörande för hur snabbt och långt en tumör växer.

Målet med arbetet har varit att skapa ett verktyg som på sikt kan hjälpa forskare förstå glioblastom bättre, men också förbättra möjligheterna till tidigare diagnos, mer individanpassad behandling och bättre prognoser. Eftersom simuleringarna sker på dator, finns även potential att minska behovet av djurförsök, vilket är bra både ur etiska och praktiska synpunkter.

På längre sikt hoppas vi att vår modell kan integreras i större, mer avancerade system, till exempel inom maskininlärning och artificiell intelligens, där den kan användas för att analysera data och generera nya hypoteser. Kombinationen av matematik, biologi och dataanalys har förändrat hur vi bedriver medicinsk forskning och kommer förhoppningsvis leda till att ge drabbade av glioblastom och andra svåra sjukdomar bättre besked i framtiden.

Sammandrag

Glioblastom (GBM) är en av de mest aggressiva och dödliga formerna av hjärncancer, med mycket varierande och invasiva tillväxtmönster som innebär stora utmaningar för prognos och behandling. Denna kandidatuppsats presenterar en stokastisk, agentbaserad modell för att simulera dynamiken hos glioblastomtumörer, med fokus på de biologiska mekanismerna go or growöch ”cell-cell repulsion”. I modellen antar varje tumörcell slumpmässigt en av tre fenotyper: ”idle”, ”go” eller ”grow”. En specialutvecklad sökalgoritm kallad P*, som inkluderar stokasticitet och en kostnadsfunktion, styr cellernas migration och delning mot mindre tätbefolkade områden i ett tredimensionellt rutnät.

För att validera modellen användes in vitro-data från Uppsala universitet, bestående av time-lapse-bilder av glioblastomsfäroider odlade i 3D-kollagenmatriser. Bildanalys användes för att extrahera centrala mätvärden som kärnradie, invasiv radie och fraktaldimension, vilka därefter jämfördes med motsvarande simulerade tumörer. Parameterestimering gjordes med hjälp av parametersvep för att kalibrera modellen. Resultaten visar att modellen kan reproducera observerad tillväxtdynamik för vissa tumörer, och de härledda parametrarna uppvisar korrelationer med kliniska data såsom patientöverlevnad och tumörsubtyp. Arbetet framhäver potentialen hos stokastisk modellering som ett kompletterande verktyg inom cancerforskningen.

Abstract

Glioblastoma (GBM) is one of the most aggressive and lethal types of brain cancer, with highly variable and invasive growth patterns that pose major challenges for prediction and treatment. This bachelor's thesis presents a stochastic, agent-based model aimed at simulating the dynamics of glioblastoma tumors, with emphasis on the biological mechanisms ”go or grow” and cell-cell repulsion. In the model, each tumor cell probabilistically adopts one of three phenotypes: ”idle”, ”go”, or ”grow”. A custom search algorithm P*, incorperates stochasticity and a cost function, which is used to guide the migration and division of cells toward less populated regions in a three-dimensional grid.

To validate the model, *in vitro* data from Uppsala University was used, consisting of time-lapse images of glioblastoma spheroids grown in 3D collagen matrices. Image analysis extracted key metrics such as core radius, invasive radius, and fractal dimension, which were then compared to corresponding simulated tumors. Parameter estimation was made by using parameter sweep, calibrating the model. The results demonstrate that the model can reproduce observed growth dynamics for some tumors, and the inferred parameters show correlations with clinical data such as patient survival and tumor subtype. This work highlights the potential of stochastic modeling as a complementary tool in cancer research.

Innehåll

1 Inledning

Den mest aggressiva och även vanligaste formen av hjärncancern gliom heter glioblastom (GBM) [1]. GBM är svår att behandla kirurgiskt eftersom den invaderar frisk hjärnvävnad [2] vilket gör att diagnostiserade patienter sällan överlever längre än 8-15 månader [3]. Forskning kring GBM är mycket aktuell och görs inom flera discipliner och på många olika sätt, däribland matematisk modellering [4]. Målet med matematiska modeller är att kunna modellera GBM:s beteende i en dator och med hjälp av modellen kunna dra slutsatser om hur sjukdomens förlopp kommer se ut för en patient. Detta kan i sin tur bidra till förbättrade behandlingsstrategier och därigenom minska patientdödligheten [5].

2 Bakgrund

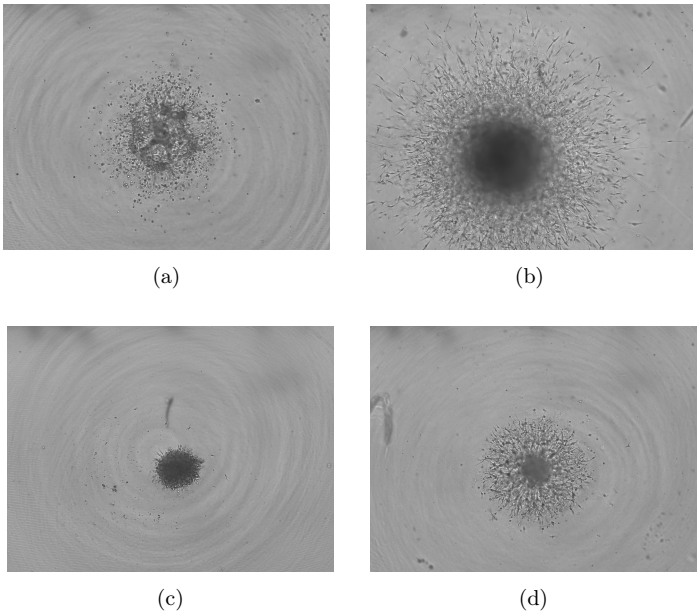
En av de stora utmaningarna inom medicinsk modellering är att avgöra hur väl en modell kan efterlikna det verkliga beteendet hos komplexa biologiska system, såsom tumörer. Kritiker menar att de data som används för att optimera modeller ofta förenklar de verkliga förhållandena där systemen existerar. Förhoppningen är att dessa förenklingar kan motiveras genom biologiskt baserade mekanismer som valts för modellen [6]. I fallet med glioblastom har både stokastiska modeller [7] och modeller baserade på partiella differentialekvationer använts, med goda belägg för att matematisk modellering ger insikt i tumörbeteende [4].

I denna studie används stokastisk modellering, vilket innebär användning av olika matematiska metoder för att beskriva hur slumpvariabler påverkas och utvecklas över tid [8]. En viktig skillnad mellan stokastiska och deterministiska modeller är att stokastiska modeller tar hänsyn till osäkerheter och variationer, vilket innebär att samma initialvärden kan leda till olika resultat. Denna modelltyp möjliggör simulering av tumörers dynamiska egenskaper och expansion. För att anpassa modellen till verkliga tumörer används så kallade metrik: kvantitativa mått som används för att jämföra modellerade och verkliga tumörers egenskaper. Skillnader i dessa mått används sedan för att justera modellens parametrar genom statistisk parameterskattning [2].

Den typ av experimentella data som används i detta arbete består av bildserier av multicellulära tumörsfäroider, alltså kluster av glioblastomceller odlade i en tredimensionell kollagenmatris [9]. Bildmaterialet består av tidsserier som visar hur tumörerna växer och förändras över tid. Dessa bilder har tillhandahållits av våra handledare och kommer från Uppsala Universitet.

Det experimentella materialet består av bildserier över olika cellinjer. Varje bildserie tillhör en viss cellinje. En cellinje består av tumörceller som har isolerats från en patient och därefter odlats vidare under kontrollerade förhållanden i laboratoriemiljö. Cellerna i en cellinje delar samma genetiska ursprung och egenskaper, vilket gör dem användbara för jämförande studier och reproducerbara experiment. I vårt material ingår totalt 25 olika cellinjer med 8 replikat per linje. Till varje cellinje finns även tillhörande patientinformation, och subtyper av tumörer, vilket möjliggör koppling mellan modellbeteende och tumörernas ursprungliga egenskaper. Denna information kommer från biobanken Human Glioblastoma Cell Culture (HGCC) [10].

När tumörsfäroider växer uppstår typiskt ett yttre område som är mindre tätt än mitten. Den inre regionen kallas ofta för kärna, medan den yttre delen benämns som det invasiva området [2]. Tillväxtmönstret tros vara kopplat till två biologiska fenomen: ”cell-cell-repulsion” och fenotypbyte. ”Cell-cell-repulsion” innebär att celler tenderar att stöta bort varandra, vilket kan driva utvandring från tumörens kärna. Även om ”cell-cell-repulsion” saknar starka biologiska bevis, har det visat sig vara en användbar mekanism för att förklara empiriska mönster i hur celler migrerar från tumörsfäroider [7]. Fenotypbytet, ofta kallat ”go or grow”-mekanismen, beskriver hur glioblastomceller pendlar mellan att dela sig (”grow”) och att migrera (”go”) [7]. Studier tyder på att dessa två tillstånd är ömsesidigt uteslutande, alltså att celler inte både kan röra sig och dela sig samtidigt [11]–[13]. Genom att stokastiskt modellera både ”cell-cell-repulsion” och ”go or grow”-mekanismerna kan man motivera modellens dynamik med biologiska observationer från både *in vitro* och *in vivo* studier. Det ska dock sägas att det tidigare nämnda tillväxtmönstret med en tät kärna och en invasiv del inte alltid infinner sig hos de *in vitro* bilder vi undersöker. I figur 1 exemplifieras olika utseenden hos tumörerna.



(a) (b) (c) (d)

Figur 1: Olika utseenden på *in vitro* tumörer. Samtliga bilder är från sista tidssteget i respektive tidsserie.

3 Syfte & Frågeställning

Syftet med detta arbete är att öka förståelsen för glioblastoms tillväxt och spridning genom att undersöka och modellera *in vitro* data. Modellen utvecklas för att simulera det dynamiska beteendet hos GBM-celler, med fokus på att fånga dynamiska egenskaper såsom ”go-or-grow” och ”cell-cell-repulsion” interaktioner. Genom simuleringar ämnar vi undersöka modellens förmåga att beskriva observerade data och uppskatta parametrar från *in vitro* data och dess dynamiska egenskaper med hjälp av parametertest, samt att använda dessa parametrar för att särskilja och klassificera olika subtyper av GBM.

Vidare avser arbetet att utvärdera huruvida de skattade modellparametrarna visar någon korrelation med kliniska variabler, specifikt patienternas överlevnadstid och ålder. Studien inkluderar även en undersökning av hur väl de specifika måtten: kärnradie, invasiv radie samt fraktaldimension, lämpar sig för att beskriva och kvantifiera GBM-tumörers form och tillväxt. Genom att studera hur olika modellparametrar påverkar tumörens tillväxt, spridning och form hoppas vi, med stöd av experimentell tidsseriedata, kunna tillhandahålla eventuell förståelse och prognostisering av tumören, både under obehandlade förhållanden och vid behandling.

På längre sikt kan resultaten från detta arbete bidra till att vidareutveckla modellen och algoritmen för att inkludera ytterligare egenskaper hos GBM, samt generera syntetiserade data som underlag för framtida tillämpningar av avancerade maskininlärningsmetoder, exempelvis ekvationsinlärning.

För att tydligt strukturera arbetet och uppnå det övergripande syftet kommer följande frågeställningar att besvaras:

(i) Kan vi formulera en modell som kan beskriva dynamiken hos glioblastomceller?

(ii) Kan vi, med hjälp av modellens förmåga och parametertest, uppskatta parametrar från *in vitro* data?

(iii) Kan de skattade modellparametrarna användas för att särskilja och klassificera GBM-undertyper?

(iv) Visar de skattade parametrarna någon korrelation med glioblastompatienternas överlevnadstid och ålder, som kan jämföras med en tidigare studie med samma patientdata?

4 Teori

4.1 Biologisk teori

Glioblastomceller kan odlas genom kollagenmatriser för att låta tumörceller växa i så kallade tumörsfäroider[2]. När försök har gjorts för att standardisera forskningen av glioblastom, till exempel tillväxt i kollagenmatriser, har man skapat Human Glioblastoma Cell Culture (HGCC) [10]. Det är en biobank, bestående av 48 olika mänskliga GBM cellinjer. De olika cellinjer representerar alla transkriptionella subtyper av GBM. Dessa fyra molekylära subtyper är proneurala, klassiska, neurala och mesenkymaler. Den proneurala subtypen, PL, associeras ofta med yngre patienter och en aning bättre överlevnadsprognos. Hos klassiska, CL, kan man se en tendens av högre proliferation, och därmed delar sig i högre grad, samt växer snabbare i storlek. De neurala, NL, kommer istället att ha ett uttryck av neuronala gener, och kan bero på kontamination från omgivande hjärnvävnad, vilket gör det svårare att karaktärisera något tumörspecifikt uttryck, och det gör att de anses vara de mest ovanliga. Slutligen så karaktäriseras den Mesenchymala, MS, subtypen som den mest invasiva av de fyra [14].

Glioblastoms beteende som kan delas upp i två olika delar. Centrum av tumören innehåller en tät kärna av prolifererande celler, vilket innebär att cellerna förökar sig genom att dela sig och öka i antal. Det yttre lagret innehåller istället invasiva och migrerande celler med hög motilitet (förmåga att självständigt förflyttas). [13] introducerar en hypotes som kommit att kallas ”go or grow”, för att beskriva cellens typ av beteende. Enligt hypotesen kommer en gliomcell att antingen befinna sig i ett go stadie eller ett grow stadie. Här kommer ”go” stadiet att motsvara det migratoriska, medan ”grow” motsvarar det proliferativa tillståndet. Hypotesen grundar sig i att cellen endast kan befinna sig i ett utav dessa två stadien. Vilken fenotyp, som bestäms av egenskaper, som cellen befinner sig i kommer att styras utav cellmiljöns biologiska och fysiologiska förhållanden. Innehåller miljön en hög celldensitet alternativt en god tillgång på substrat, minskar cellens motilitet och den antar en proliferativ fenotyp. Den börjar alltså dela sig och fokusera på tillväxt snarare än migration. Är celldensiteten lägre, så kommer cellen övergå till ett mer invasivt beteende där de istället väljer att migrera för att hitta en plats med bättre tillväxtmöjligheter. Ett flertal studier har genomförts för att förstå anledningen till denna typ av beteende. ”Go or grow” hypotesen är därmed högaktuell för att förstå beteendet och spridningen av cancerceller. Det ger också en förståelse till varför vissa tumörceller undgår behandling, då migrerande celler befinner sig i ett mer icke-proliferativt tillstånd, som bidrar till en lägre känslighet för strålning och andra behandlingar [13].

Ett annat essentiellt fenomen inom modellering av glioblastom är ”cell-cell-repulsion”, vilket innebär att tumörcellerna inte packa ihop sig alltför kompakt. Biologiskt sett kommer det från kontaktinhibition av migration, som innebär att när två celler kolliderar rör de sig ifrån varandra. Beteendet kan simuleras genom att införa en repulsiv kraft mellan celler som befinner sig nära varandra, och leder till att celler sprider ut sig istället för att klumpa ihop sig. I modellen av [7] används ”cell-cell-repulsion” för att beskriva cellernas tendens att röra sig ifrån tumörsfäroiden [7].

4.2 Modellen

I vår tumörmodell betraktas varje cell som en diskret agent som vid varje tidssteg Δt kan befinna sig i ett av tre fenotyper: ”idle”, ”go” eller ”grow”. Dessa motsvarar vila, migration respektive delning. Modellen utspelar sig på ett heltalsdiskret, kubiskt rutnät i tre dimensioner $[115, 115]^3$, där varje koordinat rymmer högst en cell. Vid varje tidssteg väljs cellens tillstånd stokastiskt enligt fördefinierade sannolikheter; vid go flyttas cellen, vid ”grow” dupliceras den så att en dottercell placeras. Valet av ny position styrs av den stokastiska sökalgoritmen P*.

4.2.1 Stokastiskt val av celltillstånd

Varje cell inleder tidssteget med att dra en likformig slumpvariabel $r \sim \mathrm{U}(0, 1)$. Cellen antar tillståndet $X_{\text{tillstånd}}$ utifrån sannolikheterna $p_{\text{go}}, p_{\text{grow}}, p_{\text{idle}}$ enligt

$$X_{\text{tillstånd}} = \begin{cases} \text{go}, & r < p_{\text{go}}, \\ \text{grow}, & p_{\text{go}} \leq r < p_{\text{grow}} + p_{\text{go}}, \\ \text{idle}, & p_{\text{grow}} + p_{\text{go}} \leq r < p_{\text{grow}} + p_{\text{go}} + p_{\text{idle}}, \end{cases} \tag{1}$$

där summan nedan garanterar en väldefinierad fördelning.

$$\sum \mathbb{P}(X_{\text{tillstånd}} = i) = 1, \quad i \in \mathcal{I} = \{\text{Go, Grow, Idle}\}. \tag{2}$$

Väljs tillstånden ”go” eller ”grow” påbörjas P*-algoritmen från cellens nuvarande koordinat som beräknar den nya positionen för en cell, och som beskrivs i mer detalj i delavsnitt 4.2.2. Skillnaden är att go lämnar startpositionen tom medan ”grow” låter den ursprungliga cellen stanna kvar och etablerar en ny cell på målpositionen.

4.2.2 Algoritmen för cell- migration och delning

För att fånga och efterlikna tumörens dynamiska egenskaper enligt avsnitt 4, har vi utvecklat en stokastisk sökalgoritm som verkar på varje enskild cell vid varje tidssteg. Denna hybridvariant av en sökalgoritm kallas för P*, och har i sin helhet sannolikt inte förekommit i tidigare forskning. Orsaken till valet av att bygga en sådan algoritm bygger på att modellen behöver styra cellerna mot mer gynnsamma positioner. Dessa anses vara de positionerna med låg lokal täthet enligt ”cell-cell repulsion” och med minst avstånd från startpositionen, då cellerna anses vara lata och vill minimera sin energiförbrukning.

Den klassiska kostnadsfunktionen som används i sökalgortimer ger oss naturliga verktyg för detta eftersom dess totalkostnad

$$f = g + h,$$

låter oss kombinera en faktisk rörelsekostnad g och en heuristik h som i vår implementation utgörs av antalet redan ockuperade grannar i den omgivande $3 \times 3 \times 3$-kuben. På så vis beaktas de nämnda delarna som anses vara mest relevanta. Dessutom ger kostnadsfunktionen en naturlig fortsättning för framtida utvidgningar, utan att algoritmens grundläggande struktur behöver ändras. Detta diskuteras i avsnitt 7.3.

För varje ledig position v_i som kan nås inom ett heltalsavstånd D, eller så kallad sökdjup, med giltig grannposition i beräknas kostnaden

$$c_i = g_i + \rho_i, \tag{3}$$

där g_i betecknar det euklidiska avståndet nuvarande och ny position och ρ_i är antalet ockuperade positioner i den omgivande $3 \times 3 \times 3$-kuben. Detta illustreras i figur 2, där den omgivande kuben för varje kandidat tas fram för att beräkna antal grannar.

För att välja den billigaste kandidaten sett till kostnaden vägs varje position med en exponentiell avtagande faktor

$$w_i = \exp\Big(-\frac{c_i}{c_{\max}}\Big), \quad c_{\max} = \max_i c_i. \tag{4}$$

Normaliseringen med $c_{\max}$ gör att samtliga vikter hamnar i intervallet $[e^{-1}, 1]$. Denna exponentiella normalisering eliminerar de artifaktiskt raka kanter som annars uppstår och tumören antar mer av en radialsymmetrisk invasiv radie som bättre återspeglar *in vitro* data.

Efter beräknande av vikter för samtliga grannpositioner summeras de till

$$S = \sum_{i=1}^{N} w_i, \tag{5}$$

där N är antal vakanta positioner inom det tillåtna sökdjupet D. Denna summa tjänar två syften: om $S = 0$ saknas lediga positioner och cellen förblir ”idle”; annars fungerar S som en normaliseringskonstant.

Vikter översätts sedan till sannolikheter genom att skala med summan så att sannolikheterna summerar till 1. Sannolikheten att just position v_i väljs blir

$$p_{v_i} = \frac{w_i}{S}, \quad i = 1, \ldots, N. \tag{6}$$

Detta implementeras enligt C.1 genom att multiplicera ett nytt likformigt slumptal $u \sim U(0, S)$ generas och bildar de kumulativa vikterna

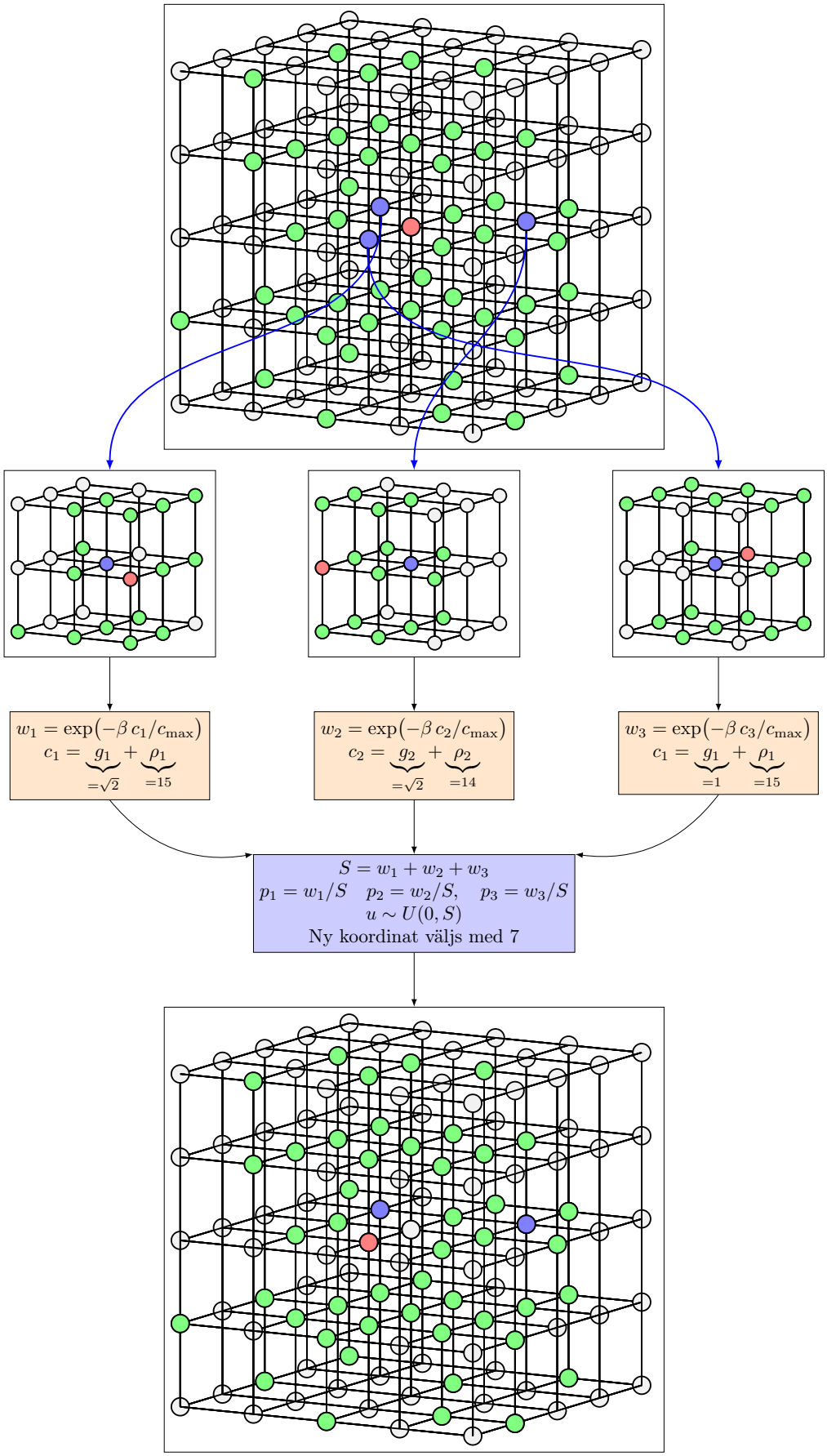
$$W_k = \sum_{i=1}^{k} w_i, \quad k = 1, \ldots, N.$$

Till sist bestäms målpositionen m_k enligt

$$m(u) = \begin{cases} m_1, & 0 \le u < W_1, \\ m_2, & W_1 \le u < W_2, \\ \vdots, & \vdots \\ m_k, & W_{k-1} \le u < W_k, \\ \vdots, & \vdots \\ m_N, & W_{N-1} \le u < S. \end{cases} \tag{7}$$

Det vill säga att första index k vars vikt överskrider slumptalet u. I koden C.1 syns detta som en enkel loop där u successivt reduceras med w_i tills u blir negativt.

Ett exempel på hur algoritmen fungerar illustreras i figur 2. Figuren visar en startposition med en röd nod och representerar en cell i ”go”-tillstånd. De gröna noderna representerar ockuperade celler, medan övriga noder illustrerar tillgängliga positioner inom cellens rörelseräckvidd $D = 1$. Vidare demonstrerar figuren hur algoritmen utvärderar varje möjlig icke-ockuperad position genom att beräkna dess kostnad enligt ekvation 3, där både antalet lokalt ockuperade grannceller och avståndet från startpositionen tas i beaktande. Slutligen visar illustrationen hur algoritmen viktar dessa kostnader och använder utfallet av en slumpvariabel för att besluta vilken av de icke-ockuperade positionerna cellen slutligen antar.



Figur 2: Visar algoritmen ur ett förenklat scenario.

4.3 Parameteruppskattning

Traditionella bayesianska metoder kräver explicit beräkning av sannolikheten för observerade data givet en viss parametervärdesättning, $p(y|\theta)$, där y representerar observerade data och θ parametervärdena. I komplexa modeller kan detta ofta vara komplicerat eller praktiskt omöjligt [15]. ABC liknande metoder kringgår denna svårighet genom att istället simulera data från modellen under olika parametervärden θ_i och sedan jämföra de simulerade resultaten med observerade data med hjälp av specificerade metriker [16]. Därför inspireras parameteruppskattningen i denna studie av approximate Bayesian Computation (ABC), där vi gör en så kallad parametersvep istället för att dra parametrar ur en priori-fördelning. Ett parametersvep är en process där parameterns värde itererar över ett intervall av möjliga värden. I vårt fall innebär det att vi utför en simulering för olika parameterkombinationer $\theta_i = (p_{\text{Go}}, p_{\text{Grow}})$ över ett intervall av parametervärden, där i betecknar och indexerar ett unikt parameterpar.

Syftet med parametertest är att uppskatta kunna approximera modellens parametrar, givet observationerna och simulerade data

$$\begin{aligned} \vec{y}_{\text{sim}} &= \big(y_{\text{sim}}(\theta_i, t_0), y_{\text{sim}}(\theta_i, t_1), \ldots, y_{\text{sim}}(\theta_i, t_n)\big), \\ \vec{y}_{\text{obs}} &= \big(y_{\text{obs}}(t_0), y_{\text{obs}}(t_1), \ldots, y_{\text{obs}}(t_n)\big), \end{aligned} \tag{8}$$

där tidsstegen är diskreta inom intervallet $[0, n]$, det vill säga $t \in \{t_0, t_1, \ldots, t_n\}$. Från (8) kan vi få fram valda måtten μ för såväl observationerna som simulerad data enligt

$$\begin{aligned} \vec{\mu}_{\text{obs}}(\vec{y}_{\text{obs}}) &= \big(\mu(y_{\text{obs}}(0)), \mu(y_{\text{obs}}(1)), \ldots, \mu(y_{\text{obs}}(n))\big), \\ \vec{\mu}_{\text{sim}}(\vec{y}_{\text{sim}}) &= \big(\mu(y_{\text{sim}}(\theta_i, t_0)), \mu(y_{\text{sim}}(\theta_i, t_1)), \ldots, \mu(y_{\text{sim}}(\theta_i, t_n))\big). \end{aligned} \tag{9}$$

Samlingsnamnet av alla mått kallas för samlingstatistik, och lagras i en vektor.

För att avgöra avvikelsen mellan observation- och simulerade datas samlingstatistik, används euklidiska avståndet

$$\left\| \vec{\mu}_m(y_{obs}, t) - \vec{\mu}_m(y_{sim}^{(i)}, t) \right\| = \sqrt{\sum_{k=0}^{t} \Big[\mu_m(y_{obs}, k) - \mu_m(y_{sim}^{(i)}, k) \Big]^2}. \tag{10}$$

Vi kan nu genom att ange en toleransnivå ϵ, enbart acceptera de avvikelserna i 10 som ligger inom toleransnivån. På så sätt erhålls parametrar ur modellen som stämmer överens med experimentell data och kan därmed uppskatta parametrar.

Parameteruppskattningen summeras av följande steg:

1. Sammanfatta observerade data y_{obs} med en lämplig sammanfattningsstatistik $\mu(y_{\text{obs}})$.
2. Dra parametervärden θ_i från ett jämnt fördelade antal parametrar över ett intervall.
3. Simulera data $y_{\text{sim}}^{(i)}$ med modellen givet parametervärdena θ_i och beräkna motsvarande sammanfattningsstatistik $\mu(y_i)$.
4. Beräkna distansen eller avvikelsen $\sigma(\vec{\mu}(y_{\text{sim}}), \vec{\mu}(y_{textobs})$ mellan sammanfattningsstatistiken från observerade och simulerade data. Parametervärden θ_i accepteras om distansen är mindre än eller lika med ett förutbestämt toleransvärde ϵ:

$$\sigma(\vec{\mu}(y_{\text{sim}}), \vec{\mu}(y_{textobs})) \leq \epsilon.$$

5. Den posteriora fördelningen $p(\theta|\mu(y))$ approximieras av fördelningen för de accepterade parametervärdena.

4.4 Fraktaldimension

Fraktaldimensionen är ett mått på huruvida bilden ger mer information vid högre upplösning. Dimensionen beräknas genom att bilden delas in i ett rutnät där varje ruta har sidlängd ϵ och $N(\epsilon)$ är antalet rutor som innehåller någon del av formen. Vidare ges fraktaldimensionen, d av

$$d = \lim_{\epsilon \to 0} \frac{\ln(N(\epsilon))}{\ln(1/\epsilon)} \tag{11}$$

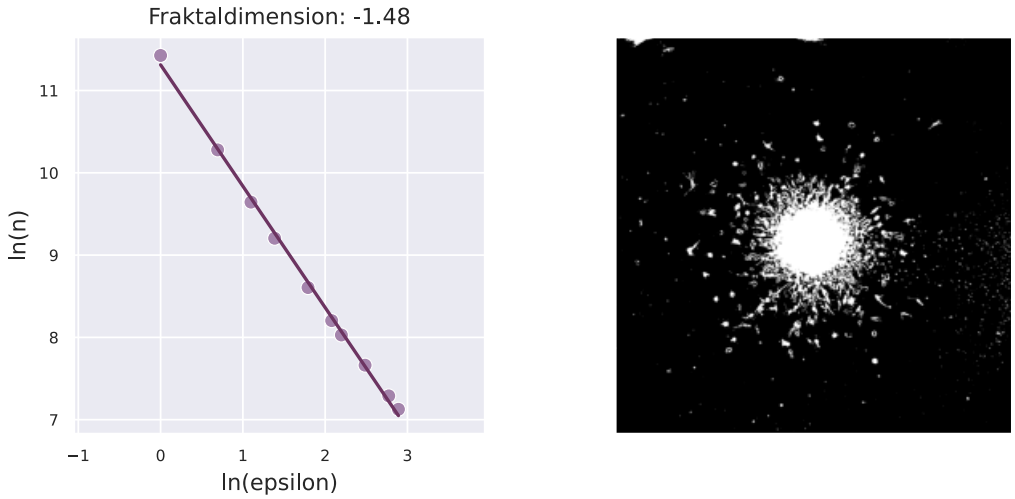
[17]. När upplösningen $\epsilon \to 0$ kan samband 11 skrivas om till proportionen

$$N(\epsilon) \propto A + \epsilon^d. \tag{12}$$

där A är en konstant. Detta implementeras genom att låta ϵ vara litet. Vidare kan 12 användas för att beräkna fraktaldimensionen genom att betrakta

$$\ln(N(\epsilon)) \propto \ln(A) + d \cdot \ln(\epsilon). \tag{13}$$

Vi kan observera att ekvation 13 ser ut som räta linjens ekvation $y = k \cdot x + m$, där k är linjens lutning och m är vart linjen korsar y-axeln. Detta gör att fraktaldimensionen d kan hittas genom att beräkna lutningen på linjen som approximeras av $\ln N(\epsilon)$ och $\ln \epsilon$. Genom att ha $N(\epsilon)$ och $\ln \epsilon$ på y- respektive x-axeln, kan vi använda linjär regression för att hitta riktningskoefficienten d. Detta illustreras i figur 3.



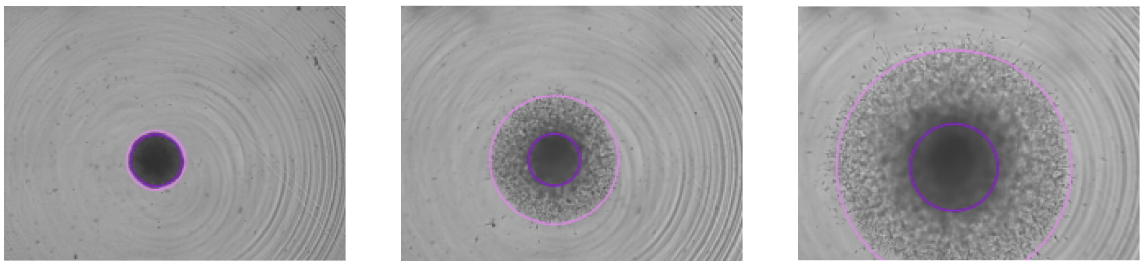
Figur 3: Bilden till höger är tumören vars fraktaldimension beräknas. Bilden till vänster har $\ln(\epsilon)$ på x-axeln och $\ln(N(\epsilon))$ på y-axeln. Punkterna är antalet rutor, N, med sidlängd, ϵ. Linjen är en linjär regression av dessa punkter och lutningen på linjen är tumörens fraktaldimension.

5 Metod

5.1 Bildhantering

Från såväl *in vitro* som simulerade bilder vill vi extrahera en del information, som blir underlag för de metriker vi senare använder för att utvärdera modellen. Från varje bild på en tumör, verklig och simulerad, vill vi ta fram dess kärnradie, invasiva radie, densitetsprofil, samt fraktaldimension. Bildbehandlingsmetoden som används i detta arbete är inspirerad av [4], men den har implementerats i Python istället för MATLAB. Förbehandlingen av bilderna följer i huvudsak samma steg, vilka beskrivs närmare i Appendix B. Efter förbehandlingen har även egna metoder som avviker från [4] använts.

För att extrahera data från tumörbilderna skapades en algoritm som inleder med flera förbehandlingssteg. Dessa steg inkluderar bland annat konvertering av bilder till gråskala, beräkning av bildens medianvärde och användning av ett Gaussiskt filter för att minska brus. Som proxy för densitet används pixelintensiteten, ett tillvägagångssätt som är hämtat från [4]. Det vi kallar densitetskurvan är alltså i själva verket en kvantifiering av hur den genomsnittliga pixelintensiteten ändras när vi rör oss ut från tumörens mitt. För att identifiera radien för den invasiva delen analyserades bildens densitetskurva, och simple moving average används för att jämna ut densitetsfördelningen. Derivator av denna utjämnade densitetskurva undersöktes för att lokalisera gränsen där densiteten började avta, vilket anger gränsen för den invasiva delen av tumören. För att bestämma kärnradien användes ett tröskelvärde på 75% av den maximala densiteten, ett värde som baserades på egen empiri. Exempel på *in vitro*-tumörer med radier bestämda av algoritmen visas i figur 4. Hur densitetsprofilen beräknas återges i appendix B tillsammans med ytterligare förklaringar om bildhanteringen och valda parametrar. På den simulerade data användes samma algoritm men där somliga parametrar modifierats för att ta hänsyn till den ändrade upplösningen i rummet.



(a) Första bilden i tidsserie. Kärnradie och invasiv radie sammanfaller.

(b) Bild från mitten av en tidsserie. Tumören har nu en tydlig invasiv del.

(c) Sista bilden i en tidsserie.

Figur 4: Tre bilder från en och samma tidsserie. Den yttre cirkeln markerar den invasiva radien och den inre cirkeln markerar kärnradien.

Fraktaldimensionen beräknas enligt avsnitt 4.4. Till skillnad från när övriga måtten beräknas, utförs inte brusreduceringsstegen på *in vitro* data eftersom de påverkar bildens skärpa, vilket i sin tur påverkar fraktaldimensionen. För att kunna beräkna (ϵ) beskärs bilderna som från början hade dimensioner 1536×1152 pixlar, till kvadrater. *In vitro* data beskärs till en sidlängd på 1152 pixlar. Vidare centreras tumören i högsta möjliga mån. Detta för att delar av tumören inte ska skäras bort och därmed behålla all relevant information. De simulerade bilderna behöver ej centreras, då samtliga initieras i centrumet av bilden. Däremot beskärs de simulerade bilderna till en sidlängd på 180 pixlar trots att de redan är kvadratiska för att bilden ska vara jämnt delbar med många små tal. Sidlängderna ϵ på rutorna i rutnäten som används på *in vitro* data visas i tabellen nedan.

Typ	Sidlängd (pixel)
$\epsilon_{in\ vitro}$	1, 2, 3, 4, 6, 8, 9, 12, 16, 18
$\epsilon_{\text{simulering}}$	1, 2, 3, 4, 5, 6, 9, 10, 12, 15

Eftersom förhållandet mellan tumörens storlek och rutnäten varierar mellan *in vitro* och simulerade data samt att bruset i *in vitro* data varierar mellan replikat, beräknas absoluta skillnaden mellan fraktaldimensionen vid första tidssteget och tidssteg i. Med andra ord beräknas $d_i - d_0$. Denna differens beräknas istället för fraktaldimension vid endast tidpunkten i.

Målet med bildbehandlingen har varit att använda en enhetlig algoritm, med samma variabler och tröskelvärden för att extrahera data från alla bilder. Detta har dock lett till vissa utmaningar, i synnerhet med att extrahera radier på tumörer som är glesa, och diskuteras vidare i 7.2. Vidare är den största utmaningen i bildhanteringen att skapa just en algoritm som generaliserar väl för

bilder på tumörer med olika egenskaper. Ett av dessa problem varierande densitetsprofiler som beror på skillnaden i upplösningen på klinisk- och simuleringsdata. Detta löstes genom att min-max normalisera och linjärt interpolera densiteten.

5.2 Simulering av Modellen

5.2.1 Parametergenerering

Innan simuleringen påbörjas genereras de parametrar som ska skattas: ”go”, ”grow” och ”idle”. Dessa inkluderar sannolikheter enligt avsnitt 4.3, antal iterationer, rymdens storlek samt tumörens initiala radie. Enligt avsnitt 4.3 utförs ett parametersvep. För att göra detta utförs simuleringarna i så kallade ”batches”, där varje batch består av simuleringar med en fix vald p_{Grow} inom intervallet [0.0,0.3] och stigande go-värden. Inom en batch är det valda initiala go-värdet $_{\text{Go}} = 0.001$ och ökar successivt tills ekvation 14 är lika med noll. Detta eftersom p_{Grow} är konstant och p_{Idle} bestäms av

$$1 - p_{\text{Go}} - p_{\text{Grow}}, \tag{14}$$

avbryts parametergenereringen när p_{Go} når ett värde som gör att ekvation 2 bryts. Det vill säga att totala sannolikheten överskrider 1. De genererade parametrarna sparas i JSON-filen `multi_params.json` för senare användning.

5.2.2 Simuleringsprocessen

Simuleringen implementeras enligt C.2. Samtliga simuleringar valdes att initieras med en homogen tumör med en radie på 10 placeras och centrum i origo. Dimensionerna på både rymden och tumören valdes för att undvika att tumörtillväxten begränsas av utrymmesbrist och ger en slutgiltig symmetrisk bilddimension på 230×230. Rymdens storlek ansågs också vara optimal ur ett upplösningsperspektiv, vilket gör att tumörens dynamik kan observeras och antalet celler hanteras effektivt. Det initiala antalet celler beräknades till 4 169, och med en majoritet i ”grow”-sannolikheten kan vi uppnå över 3 000 000 celler efter 120 tidssteg. I alla våra simuleringar diskretiserar vi tiden i 120 tidssteg som motsvarar experimentens längd på 162 timmar. Valet av 120 tidssteg säkerställer att dynamiken fångas samtidigt som färre tidssteg är fördelaktigt då man har begränsade beräkningsresurser och tid.

När den initiala tumörformen är etablerad påbörjas iterationerna. 120 iterationer bestämdes genom en kvalitativ bedömning tillräckligt för att observera tumörens tillväxt och dynamik samtidigt som simuleringens exekveringstid begränsas.

Som skrivet i avsnitt 4.2.2 ska cellernas fenotyp väljas för varje tidssteg. Detta utförs genom att cellerna väljs slumpmässigt ur en blandad lista av samtliga celler, utan återläggning. Denna metod garanterar att cellerna väljs jämnt över hela tumörytan och inte endast från områden med hög densitet, vilket skulle leda till att inga celler förflyttar eller förökar sig. Efter varje avslutad iteration projiceras alla celler på xy-planet för att visualisera tumörens densitet.

Resultaten sparas som PNG-bilder med inverterad gråskala densitet. Dessa bilder bearbetas sedan enligt 5.1, där den invasiva radien, kärnradien, densitetsprofiler och fraktaldimension extraheras. Slutligen består den genererade datamängden består av 127299 med unika parameteruppsättningar enligt det föregående avsnittet.

5.3 Första utvärdering av modellen

En första utvärdering av modellen görs för att se om den kan efterlikna *in vitro* data. Jämförelse av två exempelpar (A och B) av simulerade och experimentellt observerade tumörer, både visuellt och kvantitativt görs. Som exempelpar A valdes tumörer som uppvisar en snabb tillväxt och tydlig distinktion mellan kärna och invasiv del, medan vi låter par B representera tumörer karaktäriserade av mindre tillväxt och en inte lika tydlig invasiv del. För varje tumörpar visas bilder vid tre motsvarande tidpunkter, där bedöms huruvida den simulerade tumörens utseende och struktur

påminner om den verkliga. Jämförelse av första, mellersta och sista bilderna från respektive tidsserie görs.

För att förstärka den ovannämnda jämförelsen visas även den normerade utvecklingen av kärnradien och den invasiva radien över tid, för både simulering och experiment. Eftersom de simulerade tumörerna i körningarna inte uppnår samma storleksordning på radierna som de experimentella, troligen på grund av det begränsade antalet tidssteg, vore en direkt jämförelse av absoluta värden missvisande. Därför normaliserades samtliga radier till ett intervall mellan 0 och 1. Denna normalisering gör att analysen kan fokusera på den relativa utvecklingen och formen på tillväxtkurvorna, snarare än på de faktiska storlekarna. Det möjliggör en mer rättvisande bedömning av huruvida modellen lyckas efterlikna tillväxtens takt och förlopp över tid, även om den simulerade tumören ännu inte hunnit växa sig lika stor som i verkligheten.

För besvara första frågeställningen används Mean Squared Error (MSE), Mean Absolute Error (MAE) och Pearsons korrelationskoefficient (r) hos de normerade radierna. MSE mäter det genomsnittliga kvadratiska felet mellan de simulerade och verkliga värdena, vilket gör måttet särskilt känsligt för större avvikelser. Detta gör MSE användbart för att upptäcka om modellen ibland misslyckas grovt med att efterlikna den faktiska tillväxten. MAE, å andra sidan, beräknar det genomsnittliga absoluta felet och ger ett mer direkt mått på den genomsnittliga avvikelsen mellan kurvorna. Pearsons korrelationskoefficient r mäter det linjära sambandet mellan de två kurvorna, oberoende av deras exakta skalor. I det här fallet är r särskilt relevant eftersom vi vill undersöka om modellen lyckas efterlikna själva formen och utvecklingen av tumörens tillväxt över tid, snarare än att exakt förutsäga varje enskilt värde. Ett högt värde på r innebär att de simulerade och verkliga tillväxtkurvorna följer samma trend, även om de kan skilja sig något i absolut nivå.

Det är dock viktigt att understryka att vi undersöker två handplockade exempel som syftar till att visa modellens potentiella kapacitet att efterlikna biologiskt observerad tumörtillväxt. I datamängden förekommer tumörer med mycket varierande utseenden och tillväxtbeteenden, vilket innebär att modellens träffsäkerhet inte är lika hög för samtliga fall. En mer utförlig diskussion om modellens begränsningar och generaliserbarhet ges i 7.3.

5.4 Parameteruppskattning

I detta arbete valdes måtten: Fraktaldimension, invasiv- och kärnradie, radiell snittdensitet. Eftersom studien ämnar använda samma metrik på *in vitro* data är detta val av sammanfattningsstatistik lämplig. Vidare väljs $t \in [0, 27]$, vilket ska motsvara *in vitro* förfluten tid för timmar 72-234. Detta för att antalet simulerade datapunkter skulle överensstämma med antalet observationer. Fraktaldimension kunde inte användas på *in vitro* data på grund av hur de olika upplösningarna på simulerad och *in vitro* data påverkar fraktaldimension.

Anledningen till att parametersvep användes istället för ABC, det vill säga att dra parametrar ur en priori-fördelning, är på grund av det omfattande antalet simuleringar som skulle göras. Detta ledde till att simuleringarna gjordes på flera datorer samtidigt för att sedan spara metriken tillhörande simuleringen i en gemensam fil.

För samtlig metrik förutom radiell snittdensitet fungerar metoden beskriven i teoriavsnitt 4.3. Detta för att dels dimensionsfel, dels också skalningsfel förekommer i avståndsberäkningen, då radiell snittdensitet redan är vektorvärd. Istället utvidgas vektorerna med nollor tills längden på vektorn blir n. Detta resulterar sedan i en matris med dimension $t \times n$ som reduceras till en vektor med längd tn sådan att första raden ur matrisen är de första n elementen i vektorn. Med andra ord, summerar vi för snittdensiteter över tn vid beräkning av 10. Toleransen väljs dynamiskt genom att acceptera de θ_i som ger de 0.5% lägsta värdena av σ_i. Detta för att inte behöva justera tröskelvärdet manuellt för olika tidpunkter.

Valideringen av parameteruppskattningen utförs genom att 299 replikat valdes slumpmässigt av de 127299 simulerade. Replikaten agerade som observationer enligt parameterintervallen beskrivna i ekvation 2 och resterande data agerade som underlag för vilka parametrar som gav liknande metrik. För varje tidssteg och mått beräknades punktestimatet av de accepterade parametrarna för samt-

liga observationer, vilket vi använde som estimat för parametern. Slutligen beräknas medelvärdet av felen mellan observationernas kända parametrar och det estimerade värdet.

5.5 Korrelation mellan uppskattade parametrar och patientutfall

Studiens syfte är till stor del att undersöka huruvida modellens dynamik kan efterlikna verkligheten. Ett sätt att visa på modellens anknytning till verkligheten är att undersöka huruvida modellens parametrar var korrelerade till patientutfall. I fallet av patientdata från *Human Glioma Cell Culture* (HGCC) [10] motsvarades patientutfall av ålder och överlevnadstid. För att undersöka korrelationen användes samma metodik som i [4], då samma patientdata användes samtidigt som modellformuleringen har markanta skillnader. Därför ansågs det intressant att se skillnader och likheter i parameterkorrelation till patientutfall.

Metoden grundar sig i att beräkna korrelationen mellan ålder respektive överlevnad och modellparametrarna: p_{go}, p_{go}, p_{idle}. Dessutom gjordes korrelationstest på överlevnad anpassat efter ålder, eftersom det fanns tidigare belägg för att överlevnadstid påverkas av ålder vid diagnos [18]–[20]. En korrelationsberäkning mellan två variabler med en tredje som kontroll gjordes genom att beräkna den partiella korrelationen som i [4]:

$$\rho_{A,B\cdot C} = \frac{\rho_{A,B} - \rho_{A,C}\rho_{A,B}}{\sqrt{1-\rho_{A,c}^2}\sqrt{1-\rho_{B,C}^2}},$$

där A och B var variablerna för vilka korrelationen skulle beräknas och C var variabeln som skulle kontrolleras för. I korrelationstestet motsvarades A av modellparametrar, B av överlevnad och C av ålder. Som ett referensvärde för att avgöra styrkan av de beräknade korrelationskoefficienterna användes korrelationen mellan ålder och överlevnad hos patientdata i biobanken HGCC: $-0,46$ [4].

För måtten kärnradie, invasiv radie samt radiell snittdensitet gjordes varsin parameteruppskattning genom parametersvep, se teoriavsnitt 4.3, och därför gjordes korrelationstestet för varje enskild uppsättning av resulterande parametrar. Nämnt i metodavsnitt 5.4 är att fraktaldimensionsmåttet inte användes för parameterestimering eftersom omskalningsmetoden inte fungerade för måttet. Vissa cellinjer som parameteruppskattades exkluderades från korrelationstestet, eftersom de inte hade korrekt antal bilder i tidsserien, som skrivet i metodavsnitt 5.4. Vissa användes inte på grund av att patienterna fortfarande var vid liv vid tillfället för datainsamlingen till HGCC.

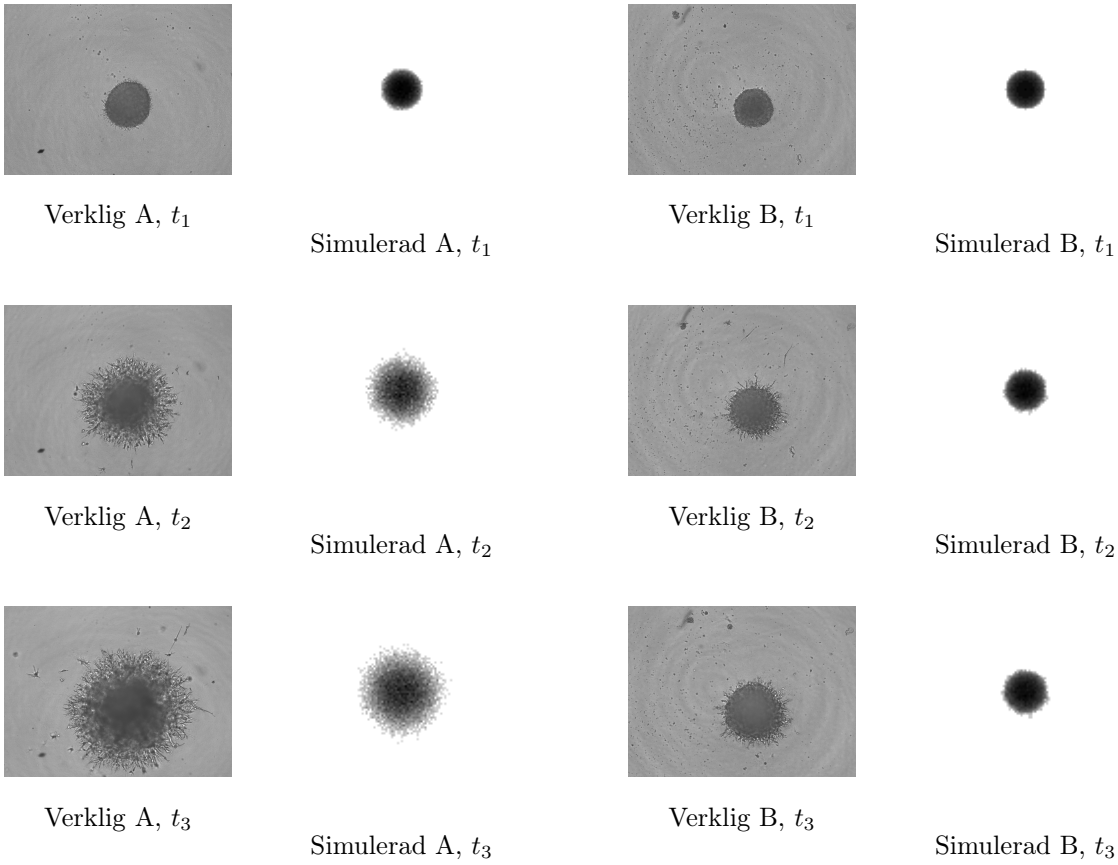
6 Resultat

6.1 Kan modellen beskriva dynamiken hos glioblastomceller?

För att undersöka om vår modell kan efterlikna glioblastomens tillväxt, genomför utvärdering av simuleringsresultaten som en rimlighetskontroll i relation till den första frågeställningen.

6.1.1 Visuell jämförelse

Figur 5 är en visuell jämförelse som bekräftar att modellen kan efterlikna utseendet av de två olika paren A och B väl.



Figur 5: Exempel på verkliga och simulerade tumörer vid tre olika tidpunkter för två dataset (A och B).

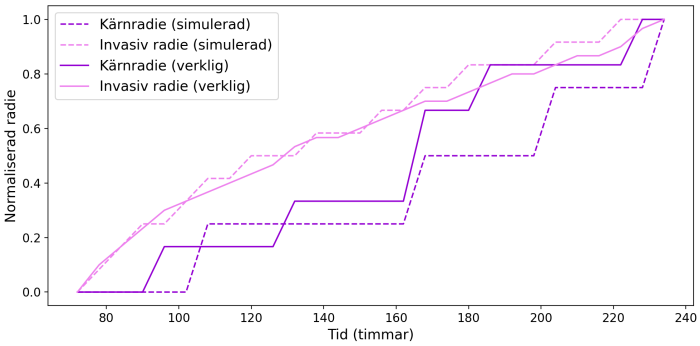
6.1.2 Kvalitativ jämförelse av radier över tid.

Pearson-korrelationen mellan simulering och verklighet för A var 0.94 för kärnradien och 0.99 för den invasiva radien, vilket tyder på att modellen i detta fall fångar den övergripande tillväxttakten mycket väl. Felmåtten visar att de simulerade radierna i genomsnitt avviker med cirka 0,12 (normaliserade enheter) i kärnan och endast 0,04 i den invasiva delen, samtidigt som de stora enstaka avvikelserna, viktade extra tungt i MSE, för den invasiva zonen är försumbara. Detta understryker att modellen i Par A inte bara fångar den övergripande formen (hög Pearson-r) utan även håller en låg felmarginal över tid.

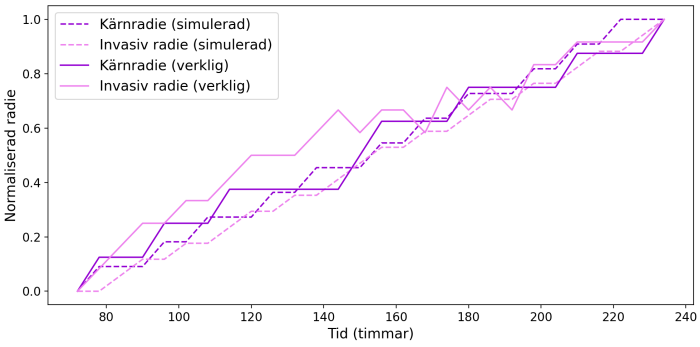
För Par B ser vi en något annorlunda bild jämfört med Par A, men fortfarande god överensstämmelse mellan simulering och experimentella data (se Tabell 1). Pearson-korrelationen för kärnradien uppgick till 0,98, vilket indikerar att tillväxtprofilens form fångas mycket väl även här. Felmåtten visar MSE = 0,0038 och MAE = 0,0499 för kärnan, vilket innebär att de genomsnittliga avvikelserna är små. För den invasiva radien var Pearson-r något lägre, 0,97, med MSE = 0,0165 och MAE = 0,1063. Detta pekar på att modellens reproduktion av invasiv expansion är god men att avvikelserna i absoluta termer är något större än för Par A. Sammantaget bekräftar dessa mått att modellen även för Par B har kapacitet att efterlikna den relativa tillväxttakten över tid, med hög korrelation och måttliga felmarginaler.

Tabell 1: Statistiska mått för Par A och Par B

	Par A	Par B
Kärnradie		
MSE	0,0226	0,0038
MAE	0,1161	0,0499
Pearson-r	0,94	0,9831
Invasiv radie		
MSE	0,0021	0,0165
MAE	0,0363	0,1063
Pearson-r	0,99	0,9673



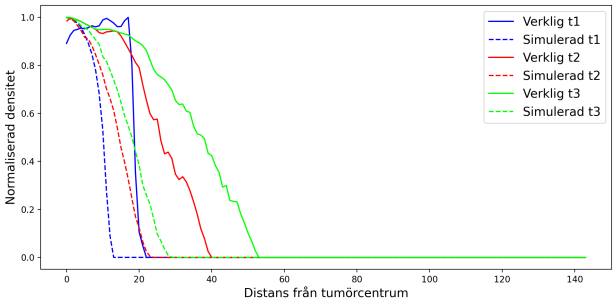
Par A



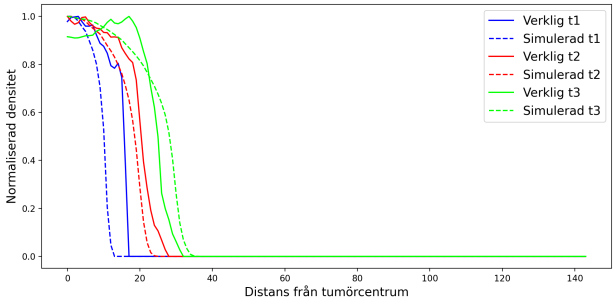
Par B

Figur 6: Jämförelse av normerade radier över tid (höger) och tillhörande statistiska mått (vänster).

6.1.3 Kvalitativ jämförelse av densitetsprofiler



(a) Jämförelse av densitetsprofiler, A



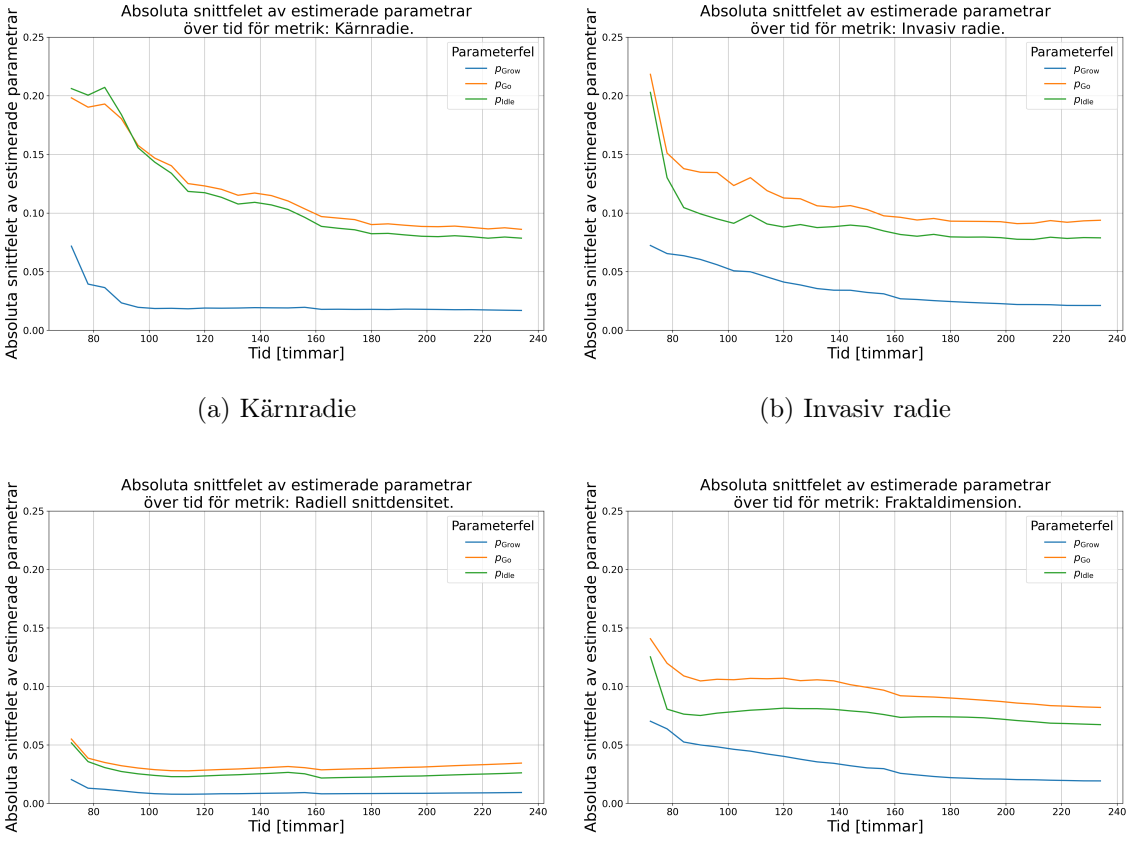
(b) Jämförelse av densitetsprofiler, B

Resultaten i figur 7a och 7b är inte lika tydliga som radieutvecklingen men illustrerar hur densitetsprofilerna jämförs i parameteruppskattningen. I båda fallen, men särskilt i 7a, är de simulerade tumörerna inte tillräckligt stora för att densitetsproilerna ska överlappa. Simulerade tumörer med andra parametervärden har visat sig kunna växa sig större och överlappa bättre med de verkliga densitetsprofilerna. De valda exempelparen A och B är alltså mindre bra för att motivera att densitetsprofilerna är jämförbara hos *in vitro* och simulerade tumörer.

6.2 Validering av parameterestimeringar på syntetisk data

De erhållna absoluta medelfelen över tid visas i figur 8. Eftersom den valda metoden utökar med data över tid, tenderar snittfelen generellt mot noll givet ökande tid, dock med avtagande hastighet. Det är värt att observera hur samtliga metrik lyckas finna en god approximation av ”grow”-parametern runt timme 180 och framåt och har lägst fel vid samtliga tidssteg. Snittfelet för de andra två parametrarna ser inte ut att konvergera mot noll i lika hög grad. Den radiella snittdensiteten presterar bäst, där den lyckas hålla lägst fel genom hela tidsserien. Vidare visar figuren hur felen för ”go” och ”idle” parametrarna ligger nära varandra. Slutligen ordnas prestandan på metriken, från bäst till sämst enligt: radiell snittdensitet, fraktaldimension, invasiv radie, kärnradie.

Även medelstandardavvikelsen för estimerade parametrar, över samma datamängd, beräknades och visas i figur 9, där data som används är för $t = 27$. Samtlig metrik visar att ”grow”-parametern har lägst spridning av underliggande accepterade värden. Utöver detta reflekterar figuren andra observationer gjorda från 8, så som rangordning och mindre osäkerhet kring underliggande värde.



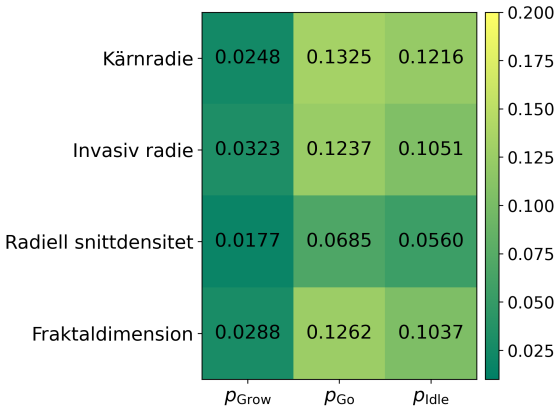
(a) Kärnradie

(b) Invasiv radie

(c) Radiell snittdensitet

(d) Fractal dimension

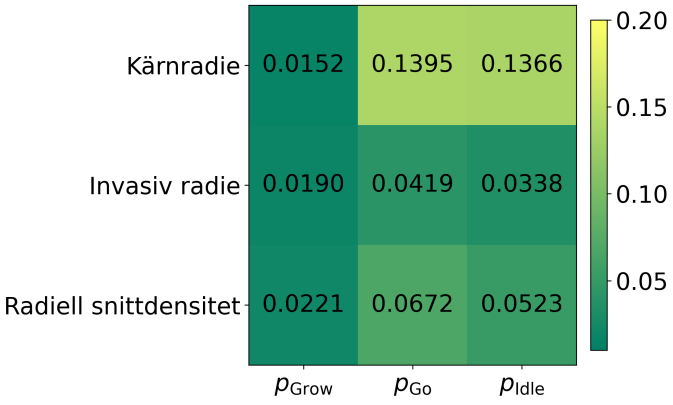
Figur 8: Visar snittfelen av medelvärdena (estimerade parametrarna) härledda från 299 slumpvalda stokastiska replikat över tid. Parametrarna som uppskattas är p_{Grow} (blå), p_{Go} (orange) och p_{Idle} (grön) över tidsserierna (motsvarande timmarna 72 - 234). Dessa fel är sedan beräknade för varje metrik (a, b, c och d) som används för att välja accepterade parameterförslag i parametersvepet.



Figur 9: Visar medelstandardavvikelsen av accepterade parametrar i parametersvep över 299 syntetiska replikat. Varje ruta i figuren representerar medelstandardavvikelsen för specifika metrik- och parameterpar. Tidsseriesteget som användes är $t = 27$.

6.3 Parameterestimering på *in vitro* data

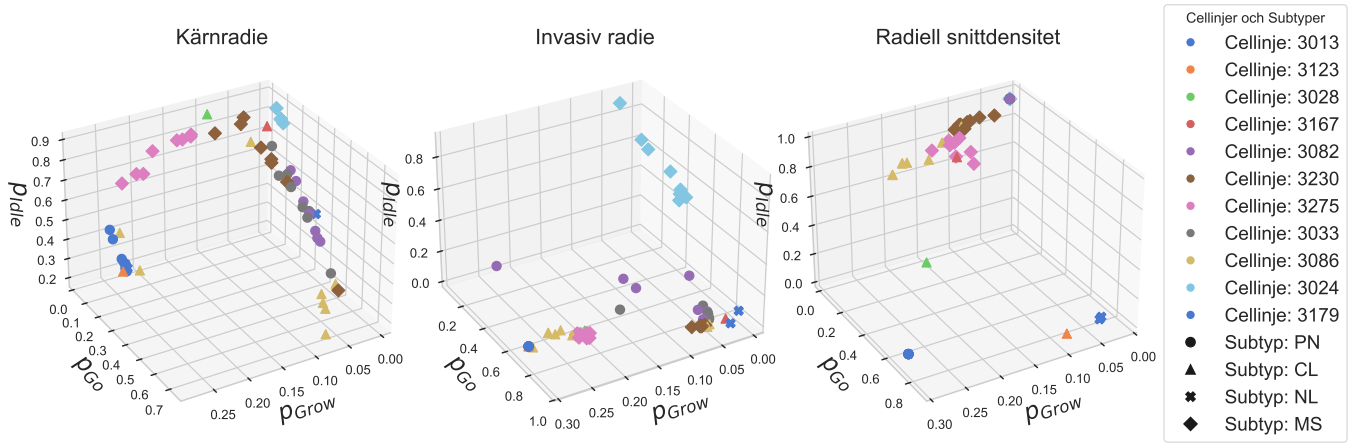
För att fastställa hur bra uppskattningen av underliggande parametrar för *in vitro* replikat är, används standardavvikelser. En kvantifiering gjordes genom att först beräkna standardavvikelsen för samtliga *in vitro* replikat för samtliga parametrar och metriker. Medelvärdet av standardavvikelserna är sedan det som visualiseras i figur 10. En liten medelstandardavvikelse indikerar att skillnaden mellan sammanfattningsstatistiken av den simulerade datamängden och *in vitro* data är liten. I kontrast ger en större medelstandardavvikelse det motsatta. Figur 10 visar att ”grow” parametern är lättast att estimera, vilket stämmer överens med det som konstaterades i 6.2. Utöver detta går det att fastställa att användandet av kärnradie som metrik hade högst medelstandardavvikelse för uppskattningen av *in vitro* parametrarna. Detta anses vara rimligt enligt jämförelse mellan de tidsserier och histogram som visas i appendix A.4, A.5 och A.7. Samtidigt stämmer värden väl överens med korresponderande värden i figur 9. Tydligast är skillnaden i medelstandardavvikelsen mellan den syntetiska valideringen och parameterestimering av *in vitro* data för invasiv radie. För underliggande data av medelstandardavvikelse *in vitro* se appendix A.1, A.2 och A.3.



Figur 10: Färgdiagram av medelstandardavvikelse för estimerade parametrar av *in-vitro* data. Varje ruta representerar denna medelstandardavvikelse för unika metrik- och parameterpar.

För att undersöka om de skattade modellparametrarna kan ge information om vilken subtyp en cellinje tillhör jämförs de estimerade modellparametrarna för var och en av måtten. Dessa är sammanställd och visas i figur 11 där färgen på punkterna indikerar vilken cellinje de tillhör och formen vilken subtyp av GBM. Figuren visar hur alla tre metriker visar tendenser till kluster. I

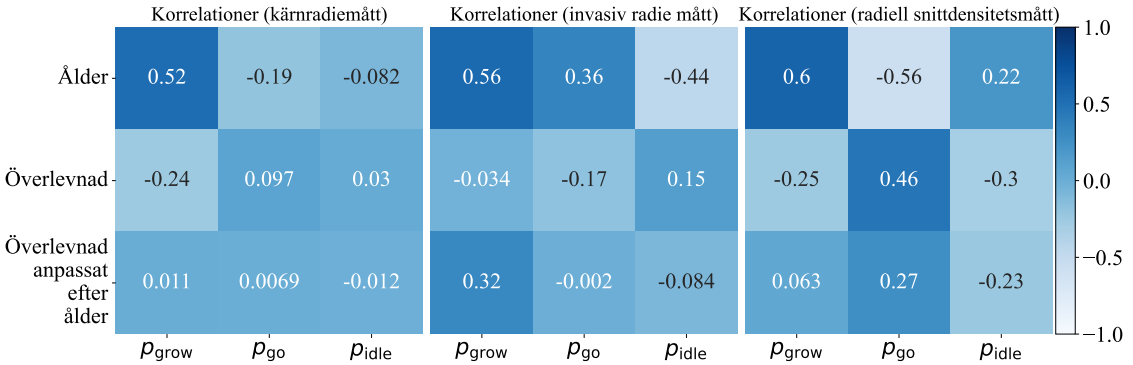
den vänstra figuren som visar för kärnradien, samlas alla replikat på en parabolliknande kurva och kan alltså anses vara ett kluster. Värt att notera att cellinjer, 3230, 3275 och 3024 som är av subtyp MS ligger högst upp i figuren för kärnradie, där ”idle” är stor. För invasiv radie är olika cellinjer av subtyp MS inte på samma plats på grafen men replikat inom cellinjerna visar kompakta kluster. Även radiell snittdensitet visar att subtyp MS än en gång befinner sig i delen av grafen med hög ”idle”. Alla replikat för cellinje 3179, den enda av subtyp NL, har för alla metriker liknande parameterskattningar och har en punktliknande kluster.



Figur 11: Visar parameterskattningar gjorda med de olika metrikerna, kärnradie, invasiv radie, radiellsnittdensitet. Färgen på markörerna indikerar cellinje och formen indikerar vilken subtyp den cellinjen är av.

6.4 Visar de skattade parametrarna någon korrelation med klinisk data?

I figur 12 visas de sammanställda korrelationerna i tre korrelationsmatriser, där första raden visar de beräknade korrelationerna mellan ålder och modellparametrar för samtliga mått i parameteruppskattningen. Andra raden visar korrelationerna mellan överlevnad och modellparametrarna, och sista raden visar korrelationerna mellan överlevnad anpassat efter ålder och modellparametrarna. Eftersom HGCC [10] rymmer ett relativt litet antal patienter, som nämnt i metodavsnitt 5.5, är det svårt att tolka resultaten i figur 12 kvantitativt. Därtill ska nämnas att endast en andel av cellinjerna slutligen användes på grund av fel antal bilder i tidsserien, som skrivet i metodavsnitt 4.3. Cellinjerna för vilka patienterna levde vid datainsamlingen används ej heller. För att ändå kunna visa resulterande trender från analysen används, som nämnt i metodavsnitt 5.5, referensvärdet $-0,46$ som ett mått på betydande korrelation. I figur 10 visas att p_{grow} är lättast att estimera,



Figur 12: Korrelation mellan de uppskattade modellparametrarna och patientens överlevnadstid och ålder. Rad 1 visar korrelation mellan ålder och modellparametrar. Rad 2 visar korrelation mellan överlevnad och modellparametrar. Rad 3 visar korrelation mellan överlevnad anpassat efter ålder och modellparametrar.

vilket motsvarar att parameterns korrelation kan tillskrivas mest betydelse. Figur 10 visar också att invasiv radie måttet och radiell snittdensitetsmåttet är ungefär jämlika i sina förmågor att parameteruppskatta, medan kärnradiemåttet är sämre. Därför är det främst invasiv radie- och radiell snittdensitets-måttets korrelationer som är tolkningsbara. Generellt sett är modellparametrarna starkast korrelerade med ålder. Överlevnad visar starkast korrelation med modellparametern p_{Go} följt av p_{Idle}, men endast för radiell snittdensitetsmåttet. Överlevnad anpassat efter ålder korrelerar främst med p_{Grow} för invasiv radiemåttet, och med p_{go} för radiellsnittdensitetsmåttet.

7 Slutsatser och Diskussion

7.1 Slutsatser

Enligt resultaten i avsnitt 6.1.1, 6.1.2 och 6.1.3 anser vi att modellen är tillräckligt välformulerad för att fånga de dynamiska karaktärsdrag studien avser att fånga. Det vill säga att proliferation och motilitet fångas, framgår av den radiella tillväxt hos kärn- och invasiva radier och radiella snittdensiten som visas i avsnitt 6.1.

Enligt resultaten i avsnitt 6.2 visade fraktaldimension på liknande potential som metriken kärn- och invasiv radie under korsvalideringen av syntetisk data, men undersöktes inte för *in vitro* data. Detta på grund av den otydliga översättningen mellan simulerade och faktiska tumörer.

Resultaten i avsnitt 6.3 indikerar att parameteruppskattningen gav varierande resultat beroende på vilken metrik som användes. Felet är som lägst när radiell snittdensitet används för beräkning av accepterade parametrar. Metoden har i regel svårt att approximera p_{Go} och p_{Idle} för den syntetiska datamängden. Dock minskar snittfelet för kumulativ data. Denna observation stämmer också överens med de medelstandardavvikelser som beräknades för *in vitro* data i figur 10. Skillnaden i standardavvikelse för radiell snittdensitet kan tyda på att det finns replikat i *in vitro* data som modellen har svårt att finna likheter till i den syntetiska datamängden. Detta ty tröskelvärdet använt i parameterestimeringen inte är en hård gräns, de 0.5% bästa parametertrippletterna accepteras. Denna metod leder alltså till minsta felet för de parametrar som finns simulerade, vilket kan leda till kluster på parameterutrymmets gränser.

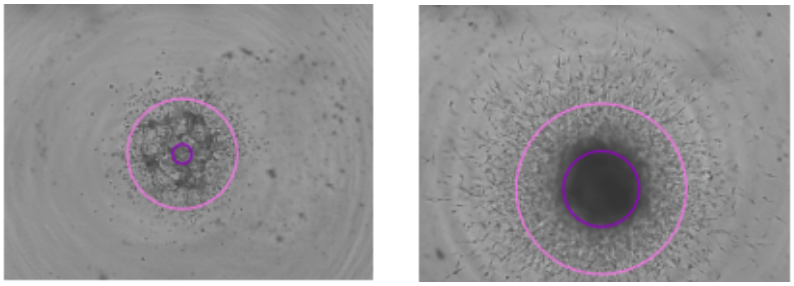
Enligt resultaten i avsnitt 6.3 kan de skattade modellparametrarna potentiellt särskilja och klassificera GBM-undertyper. Undertypen MS är den som kan gå att hitta med parameterestimat, som visas i figur 11, om flera olika metriker används. Med data från fler cellinjer är det därför också troligt att andra subtyper kan klassificeras utifrån parameterestimat.

Subtypen NL är svår att kategorisera, dels på grund av den lilla mängden data, dels på grund av dess uttryck utav neuronala gener 4.1. Däremot vid observation av 11, anser vi att det finns potential med att kunna klustra och därmed kategorisera med hjälp av modellering och simulering. Detta är ett exempel på hur matematisk modellering skulle kunna bidra till medicinsk forskning, och identifiera och kategorisera komplexa strukturer som annars är svåra med ögat. Detta är däremot något som ligger i framtiden då fler datapunkter behövs för dessa celllinjer.

Resultaten i avsnitt 6.4 kan inte användas för att säkerställa modellens exakta förmåga att prediktera patientutfall eftersom korrelationskoefficienterna har stor variation mellan olika mått. Även invasiv radie och radiell densitet, som uppvisar mer exakta skattningar än kärnradiemåttet enligt figur 9, varierar i sina korrelationskoefficienter. Orsaken ligger antagligen främst i den lilla mängden patientdata. Däremot fanns främst en intressant jämförelse att göra gentemot [4]. Ålder har generellt mest korrelation med samtliga parametrar i båda studierna, vilket för vår modell visas i figur 12. På grund av skillnaden i modellmekanismer och -formulering mellan denna studie och [4] är det dock svårt att dra direkta paralleller mellan modellparametrar. Konsekvensen blir att det också är svårt att göra fler jämförelser utan att mer rigoröst koppla parametrarna hos modellerna till varandra.

7.2 Bild- och metrikhantering

Vissa tumörer visar tydligt en tät kärna och en invasiv del medan andra saknar denna distinktion. De tumörer som saknar denna egenskap kan i sin tur delas in i de som är jämnt fyllda och de som är mer "ihåliga". Algoritmen vi använder för att bestämma radier och densitetsprofiler har parametrar som är anpassade för att passa så många utseenden som möjligt på tumörerna. Det finns dock fall där algoritmen uppenbart misslyckas med att bestämma radierna på ett tillfredsställande sätt. Två exempel på det finns i Figur 13.



(a) Kärnradien är till synes godtyckligt bestämd.

(b) Den invasiva radien underskattas vid en för spretig invasiv del.

Figur 13: Exempel på när algoritmen för bildhantering inte fungerar.

På *in vitro* data användes ingen brusreducering för att beräkna fraktaldimensionen eftersom det tar bort viktig information. Detta kompenserades med att kolla på relativa fraktaldimensionen i förhållande till första bilden i varje replikat eftersom bruset ansågs vara konstant genom hela tidsserierna. En alternativ metod för brusreducering hade varit att använda sig av ett högpassfilter som bara behåller delar av bilden med mycket skärpa. Bruset i bilderna är i jämförelse med tumörens celler i oskärpa.

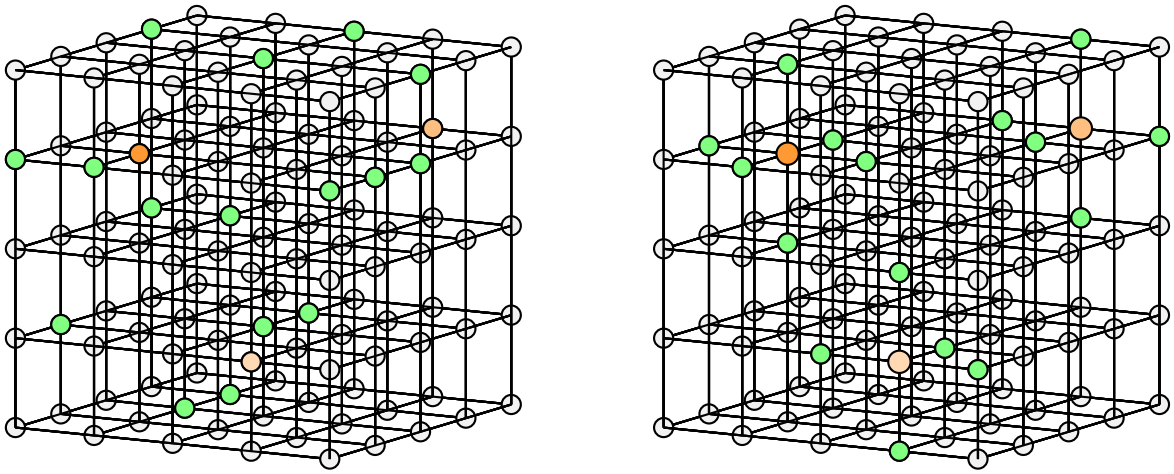
7.3 Förbättringar av modellen

I arbetet valdes en hybridmodell, benämnd P*, där en ursprungligen deterministisk algoritm omvandlades till en stokastisk modell med slumpmässiga parametrar och utfall. Som beskrivits i avsnitt 4.2.2 ger algoritmen god flexibilitet och möjlighet att på ett enkelt sätt integrera ytterligare biologiska egenskaper hos glioblastom. En potentiell vidareutveckling av modellen är att inkludera effekten av näringstillgång, där cellerna strävar efter att röra sig mot områden med hög koncentration av näring och absorbera den när cellerna nått fram.

En sådan implementering kan göras på ett intuitivt sätt genom att placera ut näringskällor vid modellens initieringsfas. Ett exempel på detta visas i figur 14 där de orangea noderna representerar näring och de olika styrkorna på färgen tillhör olika stora näringsinnehåll. Vid initialisering av tumören och det omgivande området kan varje näringsposition i tilldelas ett kostnadsvärde n_i. Därefter kan algoritmens kostnadsfunktion uppdateras enligt:

$$c_i = g_i + \rho_i - \alpha n_i, \quad \alpha > 0,$$

där parametern α reglerar vikten av näringens inflytande. Notera att termen αn_i kan väljas tillräckligt stor för att dominera sannolikheten vid senare beräkningar. Vid varje ny iteration kan cellens nya tillstånd då innefatta att absorbera näring, särskilt om kostnadsfunktionen indikerar att detta är det mest fördelaktiga valet. Eftersom kostnadsfunktionen minimeras, blir det fördelaktigt för celler att röra sig mot näringsrika områden. När en cell befinner sig intill en näringskälla kan den absorbera näring, vilket innebär att värdet på n_i minskar för varje iteration cellen stannar och fortsätter absorbera näring. Denna typ av implementering leder även till att celler tenderar att bilda kluster runt näringskällorna. Celler kan flytta sig till närliggande lediga platser vid en näringskälla eller bilda en kö i väntan på tillgång om dessa platser redan är upptagna.



(a) Visar en tumör, där näring (orangea noder) har placerats.

(b) Visar kluster runt näring efter ett antal iterationer.

Figur 14: Två sätt att initiera näringsnoder i tumör-modellen.

7.4 Paramateruppskattning

Vi har noterat att snittfelen av punktestimaten är för stora givet parameterrymden. Detta i kombination av varierande standardavvikelser för olika parametrar bör alternativa parameterestimeringsmetoder undersökas. Som skrivet i avsnitt 4.3 finns det sekventiella ABC eller MCMC metoder för att minska toleransnivån successivt under beräkningsprocessen. Detta implementerades inte i detta arbete, vilket gör att fördelningen av de erhållna accepterade värdena inte uppnår en lika hög precision som om de nämnda metoderna hade använts. Samtidigt är användandet av 0.5% bästa parametertripplettern för samtliga metriker heller inte en optimal lösning. Detta ty de strängare krav som behövs för begränsade data. Tröskelvärdet skulle alltså kunna förklara felen i figur 8. Samtidigt fanns det ett underliggande syfte med direkt jämförelse av metriker. Förkastnings-ABC har allmänt liknande problematik gällande avtagande konvergens. Men genom att kunna utforska parameterrymden mer för varje tidsseriesteg med exempelvis MCMC. Finns möjligheten till bättre konvergens.

8 Samhälleliga och etiska aspekter

Även om denna studie är av teoretisk och beräkningsmässig karaktär, bedömer vi att den kan ha viktiga implikationer för samhälleliga och etiska aspekter, inte minst om man ser till fortsatta utvecklingen av den här typen av forskning. De cellinjer som används i denna studie kommer från Human Glioblastoma Cell Culture (HGCC), ett biobanksinitiativ där glioblastomceller erhållits från patienter [10]. Enligt HGCC tillhandahålls dessa cellinjer och associerade data som en öppen resurs för forskare och läkemedelsutvecklare, vilket innebär att insamlingen och hanteringen av proverna har skett i enlighet med gällande etiska riktlinjer och lagstiftning.

Trots detta finns etiska aspekter att beakta, särskilt när det gäller tolkning och kommunikation av resultat baserade på denna typ av data och modelleringsarbete. Det är viktigt att tydligt kommunicera begränsningarna hos modellen och data den bygger på. Detta för att undvika att resultaten misstolkas, särskilt i klinisk tillämpning. Som tidigare nämnts är glioblastom en aggressiv cancerform med dålig prognos och forskning som förbättrar förståelsen av tumörtillväxt kan på lång sikt bidra till bättre behandlingsstrategier. Samtidigt medför införandet av modeller i medicinska sammanhang utmaningar, bland annat i form av behovet av omfattande validering och samarbete med klinisk expertis för att säkerställa modellens träffsäkerhet och generaliserbarhet över olika patientgrupper.

Referenser

[1] E. C. Holland, "Glioblastoma multiforme: the terminator", en, *Proc. Natl. Acad. Sci. U. S. A.*, årg. 97, nr 12, s. 6242–6244, juni 2000.

[2] A. M. Stein, T. Demuth, D. Mobley, M. Berens och L. M. Sander, "A mathematical model of glioblastoma tumor spheroid invasion in a three-dimensional in vitro experiment", en, *Biophys. J.*, årg. 92, nr 1, s. 356–365, jan. 2007.

[3] K. Anjum, B. I. Shagufta, S. Q. Abbas m. fl., "Current status and future therapeutic perspectives of glioblastoma multiforme (GBM) therapy: A review", en, *Biomed. Pharmacother.*, årg. 92, s. 681–689, aug. 2017.

[4] A. A. Malik, K. C. Nguyen, J. T. Nardini, C. C. Krona, K. B. Flores och S. Nelander, "Mathematical modeling of multicellular tumor spheroids quantifies inter-patient and intra-tumor heterogeneity", *npj Systems Biology and Applications*, årg. 11, nr 1, s. 20, febr. 2025, ISSN: 2056-7189. DOI: 10.1038/s41540-025-00492-3. URL: https://doi.org/10.1038/s41540-025-00492-3.

[5] R. Angom, N. Nakka och S. Bhattacharya, "Advances in Glioblastoma Therapy: An Update on Current Approaches", *Brain Sciences*, årg. 13, nr 11, s. 1536, 2023. DOI: 10.3390/brainsci13111536.

[6] C. Zhang, M. Jin, J. Zhao, J. Chen och W. Jin, "Organoid models of glioblastoma: advances, applications and challenges", *American Journal of Cancer Research*, årg. 10, nr 8, s. 2242–2257, 2020.

[7] M. Tektonidis, H. Hatzikirou, A. Chauvière, M. Simon, K. Schaller och A. Deutsch, "Identification of intrinsic in vitro cellular mechanisms for glioma invasion", *Journal of Theoretical Biology*, årg. 287, s. 131–147, 2011, ISSN: 0022-5193. DOI: https://doi.org/10.1016/j.jtbi.2011.07.012. URL: https://www.sciencedirect.com/science/article/pii/S0022519311003626.

[8] N. Lanchier, *Stochastisc Modeling*, en. Springer International Publishing, 2017, ISBN: 9783319-500379.

[9] S. J. Han, S. Kwon och K. S. Kim, "Challenges of applying multicellular tumor spheroids in preclinical phase", *Cancer Cell International*, årg. 21, nr 1, s. 152, mars 2021, ISSN: 1475-2867. DOI: 10.1186/s12935-021-01853-8. URL: https://doi.org/10.1186/s12935-021-01853-8.

[10] L. Uhrbom, B. Westermark, K. F. Nilsson, S. Nelander och F. Swartling, *Human Glioblastoma Cell Culture (HGCC) Biobank*, 2025. URL: https://www.hgcc.se/.

[11] A. Giese, M. Loo, N. Tran, D. Haskett, S. Coons och M. Berens, "Dichotomy of astrocytoma migration and proliferation", *IntJCancer*, årg. 67, nr 2, s. 275–282, juli 1996. DOI: 10.1002/(SICI)1097-0215(19960717)67:2<275::AID-IJC20>3.0.CO;2-9.

[12] A. Giese, L. Kluwe, B. Laube, H. Meissner, M. Berens och M. Westphal, "Migration of human glioma cells on myelin", *Neurosurgery*, årg. 38, nr 4, s. 755–764, april 1996. DOI: 10.1227/00006123-199604000-00026.

[13] A. Giese, R. Bjerkvig, M. E. Berens och M. Westphal, "Cost of migration: Invasion of malignant gliomas and implications for treatment", *Journal of Clinical Oncology*, årg. 21, nr 8, s. 1624–1636, 2003. DOI: 10.1200/JCO.2003.05.062.

[14] Y. Xie, T. Bergström, Y. Jiang m. fl., "The human glioblastoma cell culture resource: Validated cell models representing all molecular subtypes", *EBioMedicine*, årg. 2, pages = 1351–1363, okt. 2015, ISSN: 2352-3964. DOI: 10.1016/j.ebiom.2015.08.026. URL: https://doi.org/10.1016/j.ebiom.2015.08.026.

[15] S. Tavaré, D. J. Balding, R. C. Griffiths och P. Donnelly, "Inferring coalescence times from DNA sequence data", *Genetics*, årg. 145, nr 2, s. 505–518, 1997.

[16] J. K. Pritchard, M. T. Seielstad, A. Perez-Lezaun och M. W. Feldman, "Population growth of human Y chromosomes: a study of Y chromosome microsatellites", *Molecular Biology and Evolution*, årg. 16, nr 12, s. 1791–1798, 1999.

[17] S. H. Strogatz, *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*, en. Westview Press, 2014.

[18] M. Simmons, K. Lamborn, M. Takahashi m. fl., "Analysis of complex relationships between age, p53, epidermal growth factor recepto, and survival in glioblastoma patients", *Cancer Research*, årg. 61, nr 3, s. 1122–1128, 2001.

[19] A. Stark, J. van de Bergh, J. Hedderich, H. Mehdorn och A. Nabavi, "Glioblastoma: clinical characteristics, prognostic factors and survival in 492 patients", *Clinical Neurology and Neurosurgery*, årg. 114, nr 7, s. 840–845, 2012.

[20] T. Tykocki och M. Eltayeb, "Ten-year survival in glioblastoma. A systematic review", *JournalofClinicalNeuroscience*, årg. 54, s. 7–13, 2018.

Användning av AI i projektet

AI har använts för olika aspekter av programmeringen. Genom att använda GitHub Copilot kunde vi enkelt och effektivt felsöka. Detta gjordes dock med stor vaksamhet där samtliga delar av koden skrevs först av oss. Detta för att säkerställa att samtliga delar av koden fungerade och följde såväl beräkningarna som algoritmer och modeller på ett korrekt sätt. Vi ansåg att programmeringen var av för stor vikt för resultaten för att låta AI ta ratten. Dessutom blev koden och dess delar stora och komplexa, vilket vi ansåg att LLMs inte kan hantera på ett riskfritt sätt. Med riskaversion som utgångspunkt, används inte AI för `C++` kod. Detta på grund av den stora risken för minnesläckage.

Även vissa delar av själva kodstrukturen AI-genererades först och behandlades vid behov efteråt, men detta gäller främst repetitiva och enkla segment i koden. Slutligen användes av AI för att generera kod för figurframställning, eftersom det ansågs viktigast att kunna fokusera på det grafiska resultatet snarare än syntax i detta fall. För att tydliggöra inkluderar inte AI-koden för figurerna någon databehandling.

A Figurer & Tabeller

A.1 *In vitro* standardavvikelser för parameterestimat - kärnradie

ID	Cellline	Grow (std)	Go (std)	Idle (std)
E1_1	3013	0.0206	0.0654	0.0555
F1_1	3013	0.0206	0.0939	0.0897
F2_1	3013	0.0211	0.0892	0.0843
G1_1	3013	0.0204	0.0868	0.0813
G2_1	3013	0.0214	0.0873	0.0814
G3_1	3123	0.0191	0.1031	0.1017
H1_1	3013	0.0201	0.0789	0.0746
H2_1	3013	0.0210	0.0915	0.0859
B11_1	3028	0.0168	0.0189	0.0174
C6_1	3167	0.0095	0.1628	0.1582
A1_1	3082	0.0090	0.1557	0.1509
A3_1	3230	0.0252	0.1681	0.1519
A4_1	3275	0.0520	0.0510	0.0866
A5_1	3033	0.0074	0.1453	0.1402
A6_1	3086	0.0157	0.2315	0.2216
A7_1	3024	0.0018	0.1042	0.1049
B3_1	3230	0.0068	0.1772	0.1746
B4_1	3275	0.0519	0.0618	0.0839
B5_1	3033	0.0121	0.3005	0.2920
B6_1	3086	0.0292	0.1246	0.1155
B7_1	3024	0.0023	0.0725	0.0730
C1_1	3082	0.0086	0.1558	0.1530
C3_1	3230	0.0110	0.2667	0.2576
C4_1	3275	0.0267	0.0421	0.0315
C5_1	3033	0.0089	0.2358	0.2297
C6_1	3086	0.0063	0.1335	0.1279
C7_1	3024	0.0022	0.0988	0.0996
D1_1	3082	0.0084	0.1693	0.1656
D2_1	3179	0.0244	0.1262	0.1293
D3_1	3230	0.0096	0.1738	0.1667

Sim ID	Cellline	Grow (std)	Go (std)	Idle (std)
D4_1	3275	0.0277	0.0420	0.0279
D5_1	3033	0.0084	0.1808	0.1747
D6_1	3086	0.0122	0.1169	0.1121
D7_1	3024	0.0018	0.1044	0.1052
E1_1	3082	0.0087	0.1613	0.1580
E3_1	3230	0.0110	0.2903	0.2807
E4_1	3275	0.0283	0.0354	0.0246
E5_1	3033	0.0078	0.2072	0.2018
E6_1	3086	0.0074	0.1343	0.1279
E7_1	3024	0.0023	0.0709	0.0713
F1_1	3082	0.0087	0.1457	0.1420
F3_1	3230	0.0111	0.2772	0.2674
F4_1	3275	0.0601	0.0410	0.0559
F5_1	3033	0.0099	0.1679	0.1622
F6_1	3086	0.0068	0.1362	0.1300
F7_1	3024	0.0018	0.1046	0.1054
G1_1	3082	0.0087	0.1450	0.1405
G2_1	3179	0.0101	0.2021	0.1974
G3_1	3230	0.0075	0.2408	0.2348
G4_1	3275	0.0271	0.0380	0.0273
G5_1	3033	0.0083	0.2038	0.1987
G6_1	3086	0.0071	0.1720	0.1696
G7_1	3024	0.0018	0.1038	0.1046
H1_1	3082	0.0084	0.1511	0.1465
H3_1	3230	0.0140	0.1794	0.1704
H4_1	3275	0.0290	0.0755	0.0977
H5_1	3033	0.0139	0.3256	0.3133
H6_1	3086	0.0364	0.1992	0.2225
H7_1	3024	0.0019	0.1052	0.1060

Tabell 2: Standardavvikelser av de accepterade parametrarna från parameterestimeringen av *in vitro* replikaten. För värdena ovan användes metriken **kärnradie** för parameterestimeringen. Varje tidsserie identifieras via kolumnen ID, likaså identifierar cellinje just vilken cellinje replikatet tillhör. Värdena är beräknade för sista tidsseriebilden, $t = 27$.

A.2 *In vitro* standardavvikelser för parameterestimat - invasiv radie

ID	Cellline	Grow (std)	Go (std)	Idle (std)
E1_1	3013	0.0114	0.0336	0.0267
F1_1	3013	0.0115	0.0335	0.0265
F2_1	3013	0.0115	0.0332	0.0262
G1_1	3013	0.0115	0.0333	0.0264
G2_1	3013	0.0115	0.0333	0.0263
G3_1	3123	0.0089	0.0330	0.0292
H1_1	3013	0.0115	0.0333	0.0264
H2_1	3013	0.0114	0.0334	0.0266
B11_1	3028	0.0433	0.1528	0.1195
C6_1	3167	0.0431	0.1671	0.1382
A1_1	3082	0.0115	0.0208	0.0195
A3_1	3230	0.0215	0.0218	0.0188
A4_1	3275	0.0270	0.0366	0.0241
A5_1	3033	0.0114	0.0206	0.0193
A6_1	3086	0.0226	0.0271	0.0177
A7_1	3024	0.0114	0.0204	0.0191
B3_1	3230	0.0198	0.0235	0.0211
B4_1	3275	0.0360	0.0762	0.0509
B5_1	3033	0.0115	0.0208	0.0194
B6_1	3086	0.0233	0.0643	0.0564
B7_1	3024	0.0114	0.0200	0.0187
C1_1	3082	0.0114	0.0205	0.0192
C3_1	3230	0.0224	0.0260	0.0188
C4_1	3275	0.0350	0.0543	0.0320
C5_1	3033	0.0115	0.0208	0.0195
C6_1	3086	0.0274	0.0646	0.0519
C7_1	3024	0.0115	0.0197	0.0185
D1_1	3082	0.0115	0.0205	0.0192
D2_1	3179	0.0151	0.0660	0.0562
D3_1	3230	0.0257	0.0545	0.0406

Sim ID	Cellline	Grow (std)	Go (std)	Idle (std)
D4_1	3275	0.0276	0.0500	0.0394
D5_1	3033	0.0115	0.0208	0.0194
D6_1	3086	0.0308	0.0588	0.0440
D7_1	3024	0.0114	0.0199	0.0187
E1_1	3082	0.0115	0.0201	0.0189
E3_1	3230	0.0217	0.0217	0.0182
E4_1	3275	0.0403	0.1620	0.1350
E5_1	3033	0.0115	0.0208	0.0195
E6_1	3086	0.0308	0.0587	0.0441
E7_1	3024	0.0114	0.0204	0.0191
F1_1	3082	0.0115	0.0208	0.0194
F3_1	3230	0.0165	0.0490	0.0373
F4_1	3275	0.0363	0.1124	0.0871
F5_1	3033	0.0115	0.0208	0.0195
F6_1	3086	0.0239	0.0635	0.0579
F7_1	3024	0.0114	0.0204	0.0191
G1_1	3082	0.0115	0.0206	0.0193
G2_1	3179	0.0167	0.0752	0.0627
G3_1	3230	0.0219	0.0275	0.0192
G4_1	3275	0.0270	0.0366	0.0236
G5_1	3033	0.0115	0.0208	0.0195
G6_1	3086	0.0294	0.0405	0.0249
G7_1	3024	0.0114	0.0204	0.0191
H1_1	3082	0.0115	0.0201	0.0189
H3_1	3230	0.0200	0.0343	0.0245
H4_1	3275	0.0329	0.0495	0.0279
H5_1	3033	0.0115	0.0208	0.0195
H6_1	3086	0.0315	0.0594	0.0444
H7_1	3024	0.0114	0.0204	0.0191

Tabell 3: Standardavvikelser av de accepterade parametrarna från parameterestimeringen av *in-vitro* replikaten. För värdena ovan användes metriken **invasiv radie** för parameterestimeringen. Varje tidsserie identifieras via kolumnen ID, likaså identifierar cellinje just vilken cellinje replikatet tillhör. Värdena är beräknade för sista tidsseriebilden, $t = 27$.

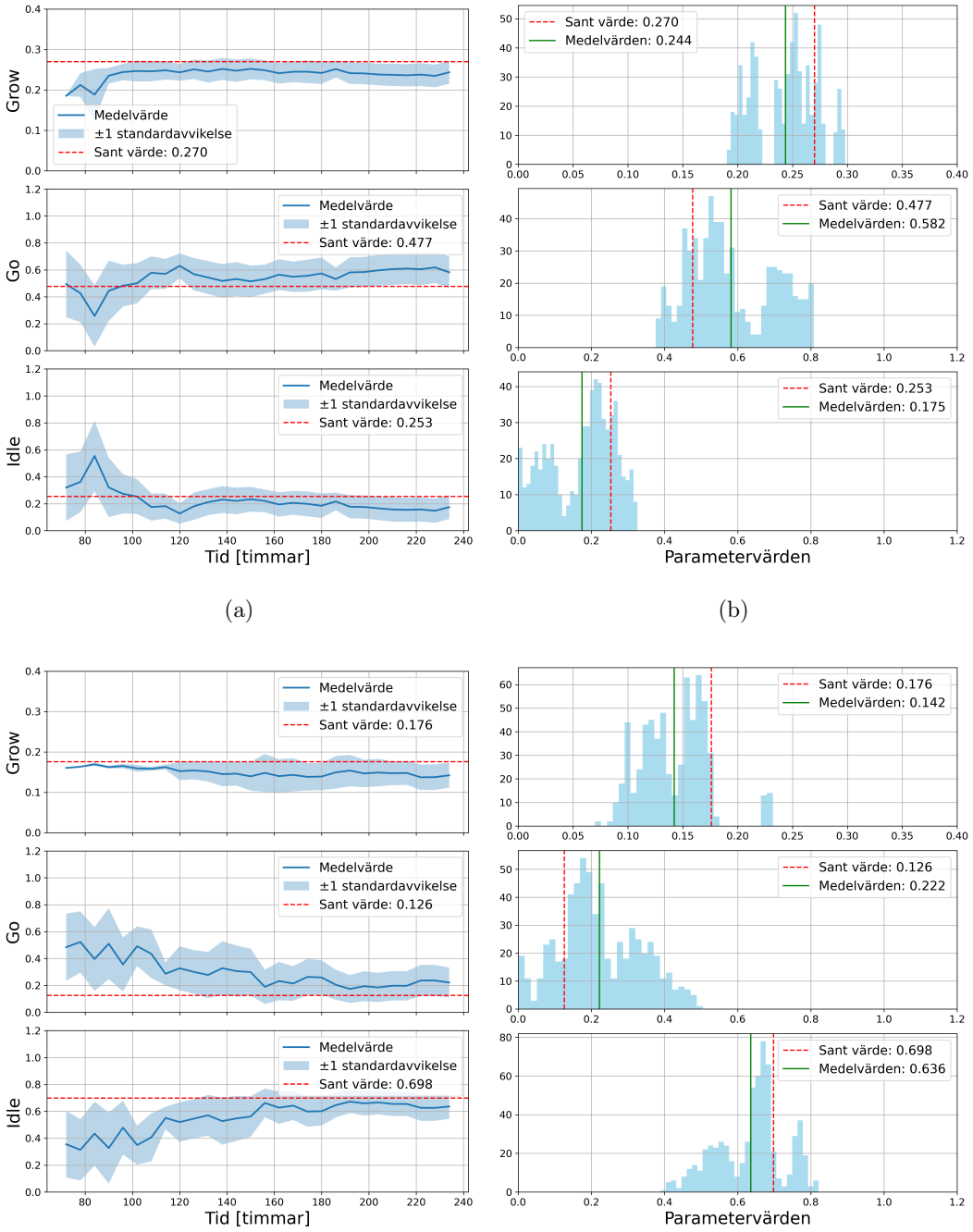
A.3 *In vitro* standardavvikelser för parameterestimat - radiell snittdensitet

ID	Cellline	Grow (std)	Go (std)	Idle (std)
E1_1	3013	0.0110	0.0316	0.0246
F1_1	3013	0.0110	0.0317	0.0247
F2_1	3013	0.0110	0.0317	0.0247
G1_1	3013	0.0110	0.0317	0.0247
G2_1	3013	0.0110	0.0317	0.0247
G3_1	3123	0.0109	0.0318	0.0248
H1_1	3013	0.0111	0.0315	0.0245
H2_1	3013	0.0110	0.0317	0.0247
B11_1	3028	0.0527	0.1006	0.0608
C6_1	3167	0.0138	0.0457	0.0388
A1_1	3082	0.0183	0.0541	0.0492
A3_1	3230	0.0062	0.0260	0.0235
A4_1	3275	0.0372	0.0513	0.0285
A5_1	3033	0.0139	0.0755	0.0683
A6_1	3086	0.0051	0.0254	0.0246
A7_1	3024	0.0094	0.0374	0.0369
B3_1	3230	0.0057	0.0268	0.0247
B4_1	3275	0.0360	0.0488	0.0281
B5_1	3033	0.1071	0.1694	0.0763
B6_1	3086	0.0138	0.0308	0.0222
B7_1	3024	0.0116	0.0584	0.0623
C1_1	3082	0.0996	0.1664	0.0912
C3_1	3230	0.0058	0.0271	0.0250
C4_1	3275	0.0658	0.1120	0.0593
C5_1	3033	0.0040	0.0379	0.0367
C6_1	3086	0.0271	0.0861	0.0639
C7_1	3024	0.0079	0.1505	0.1499
D1_1	3082	0.0046	0.0446	0.0419
D2_1	3179	0.0308	0.0410	0.0258
D3_1	3230	0.0059	0.0253	0.0232

Sim ID	Cellline	Grow (std)	Go (std)	Idle (std)
D4_1	3275	0.0621	0.1063	0.0566
D5_1	3033	0.0035	0.0451	0.0442
D6_1	3086	0.0455	0.1019	0.0663
D7_1	3024	0.0029	0.1628	0.1637
E1_1	3082	0.0104	0.0707	0.0649
E3_1	3230	0.0060	0.0259	0.0237
E4_1	3275	0.0552	0.1032	0.0587
E5_1	3033	0.0038	0.0376	0.0366
E6_1	3086	0.0354	0.0929	0.0665
E7_1	3024	0.0065	0.1388	0.1380
F1_1	3082	0.1055	0.2226	0.1264
F3_1	3230	0.0054	0.0275	0.0255
F4_1	3275	0.0667	0.1067	0.0535
F5_1	3033	0.0035	0.0477	0.0465
F6_1	3086	0.0354	0.0932	0.0662
F7_1	3024	0.0060	0.1490	0.1492
G1_1	3082	0.0032	0.0488	0.0477
G2_1	3179	0.0049	0.0522	0.0529
G3_1	3230	0.0056	0.0268	0.0250
G4_1	3275	0.0522	0.0922	0.0537
G5_1	3033	0.0038	0.0405	0.0392
G6_1	3086	0.0066	0.0219	0.0204
G7_1	3024	0.0104	0.1164	0.1171
H1_1	3082	0.0048	0.0824	0.0783
H3_1	3230	0.0058	0.0274	0.0253
H4_1	3275	0.0187	0.0517	0.0364
H5_1	3033	0.0041	0.0382	0.0369
H6_1	3086	0.0514	0.0921	0.0513
H7_1	3024	0.0104	0.0490	0.0545

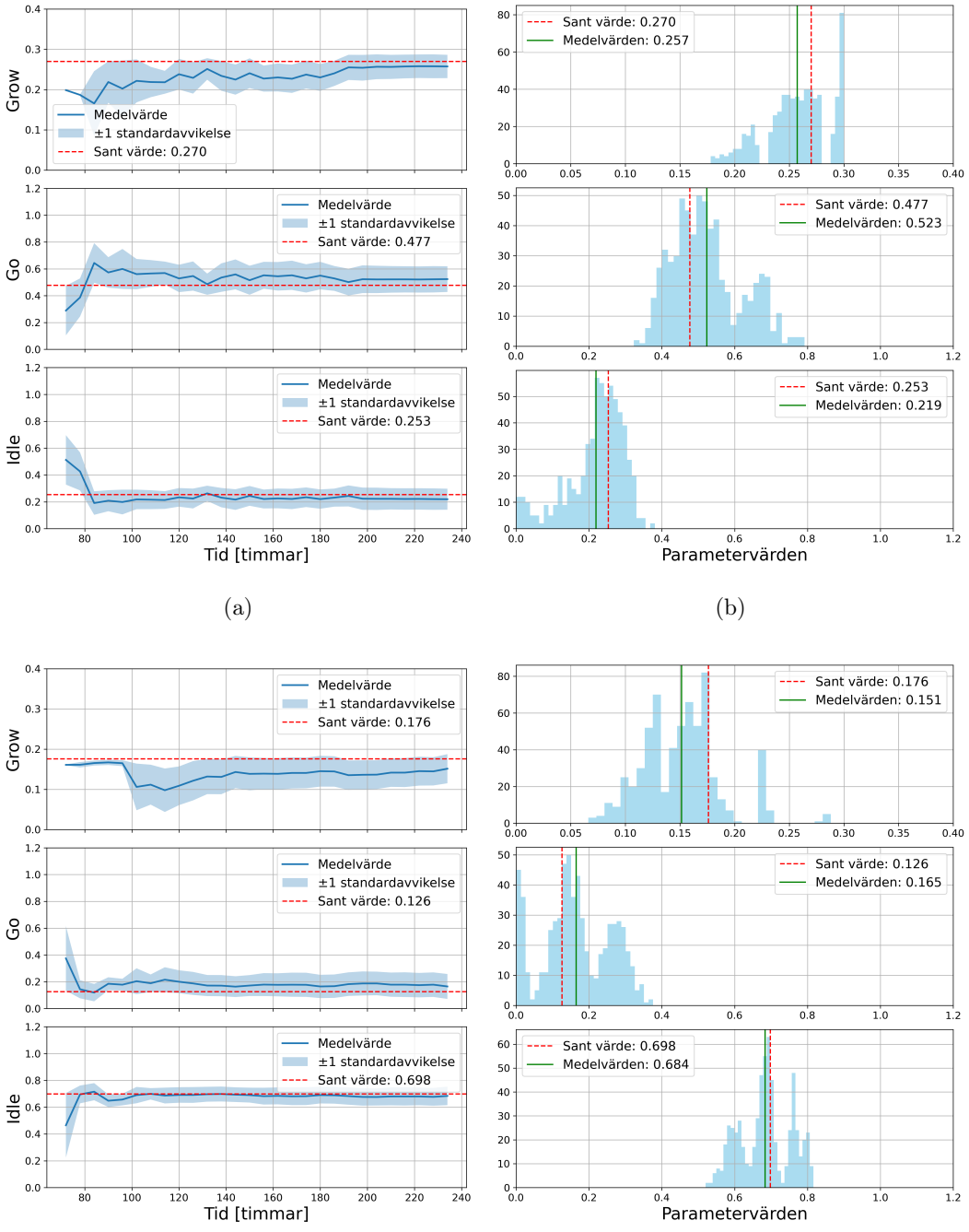
Tabell 4: Standardavvikelser av de accepterade parametrarna från parameterestimeringen av *in vitro* replikaten. För värdena ovan användes metriken **radiell snittdensitet** för parameterestimeringen. Varje tidsserie identifieras via kolumnen ID, likaså identifierar cellinje just vilken cellinje replikatet tillhör. Värdena är beräknade för sista tidsseriebilden, $t = 27$.

A.4 Parameteruppskattning av simulerade replikat - Kärnradie



Figur 15: Ovan härleddes via metriken för **kärnradie** på stokastiskt simulerade tumörer. Till vänster, visas grafer av medelvärdet av de accepterade parametrarna för parametersvep. Detta plottas mot tid för 28 bilder långa tidsserier (motsvarande timme 72 - 234 in-vitro). I rött grafas det sanna värdet på de underliggande parametrarna. Till höger, histogram över de accepterade parameterförslagen för timme 234. Även här är underliggande värden markerat i rött tillsammans med beräknat medelvärde i grönt. Det översta replikatet har underliggande parametrar $(p_{Grow}, p_{Go}, p_{Idle}) = (0.270, 0.477, 0.253)$. Det undre replikatet har istället underliggande parametrar $(p_{Grow}, p_{Go}, p_{Idle}) = (0.176, 0.126, 0.698)$.

A.5 Parameteruppskattning av simulerade replikat - Invasiv radie

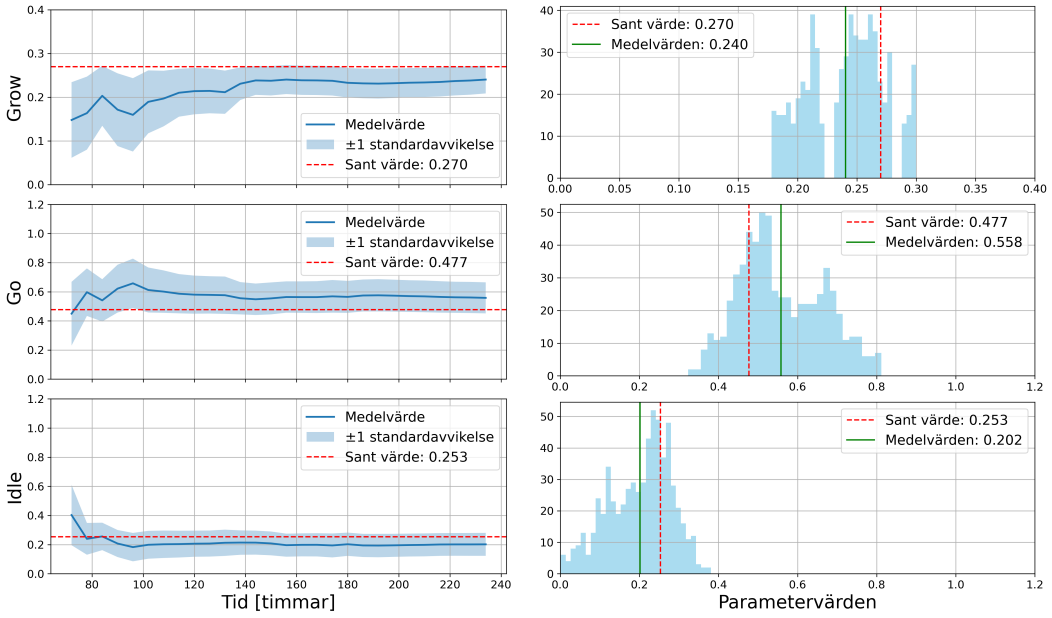


(a) (b)

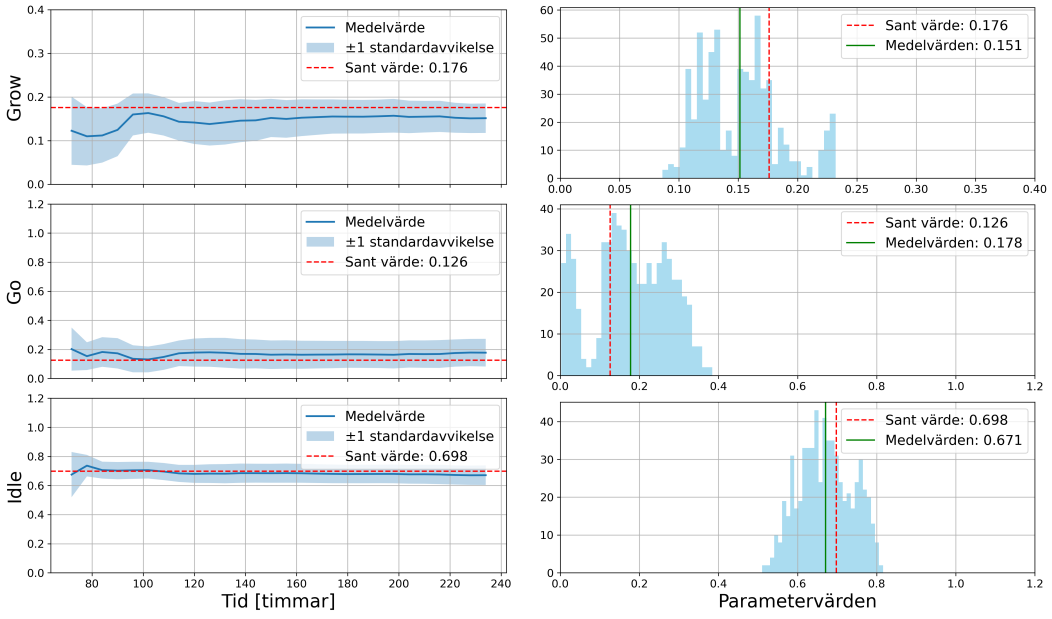
(c) (d)

Figur 16: Ovan härleddes via metriken för **invasiv radie** på stokastiskt simulerade tumörer. Till vänster, visas grafer av medelvärdet av de accepterade parametrarna för parametersvep. Detta plottas mot tid för 28 bilder långa tidsserier (motsvarande timme 72 - 234 in-vitro). I rött grafas det sanna värdet på de underliggande parametrarna. Till höger, histogram över de accepterade parameterförslagen för timme 234. Även här är underliggande värden markerat i rött tillsammans med beräknat medelvärde i grönt. Det översta replikatet har underliggande parametrar $(p_{Grow}, p_{Go}, p_{Idle}) = (0.270, 0.477, 0.253)$. Det undre replikatet har istället underliggande parametrar $(p_{Grow}, p_{Go}, p_{Idle}) = (0.176, 0.126, 0.698)$.

A.6 Parameteruppskattning av simulerade replikat - Fraktal dimension



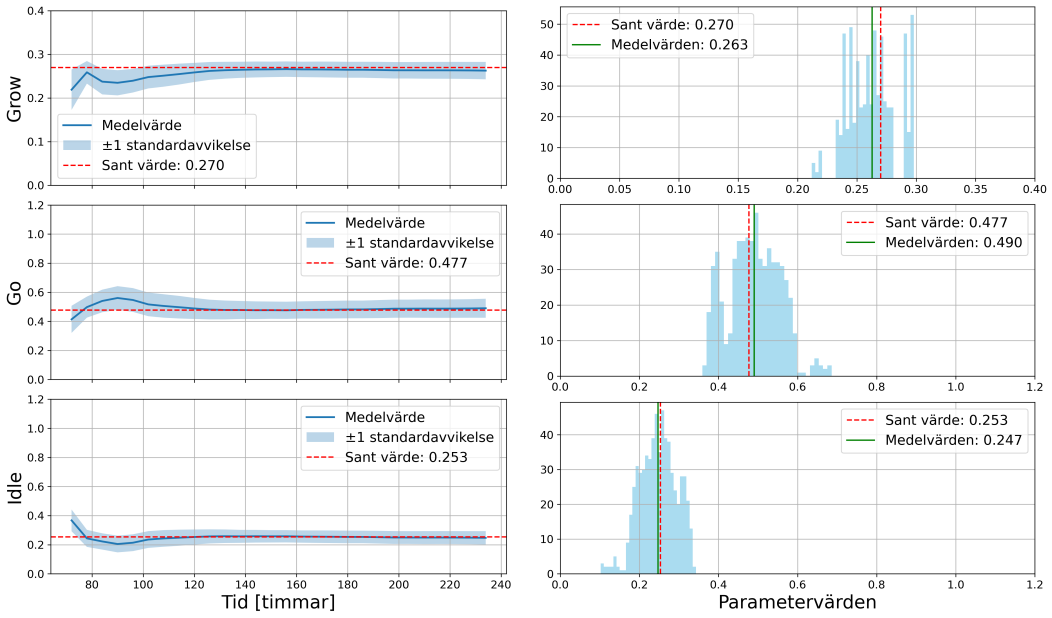
(a) (b)



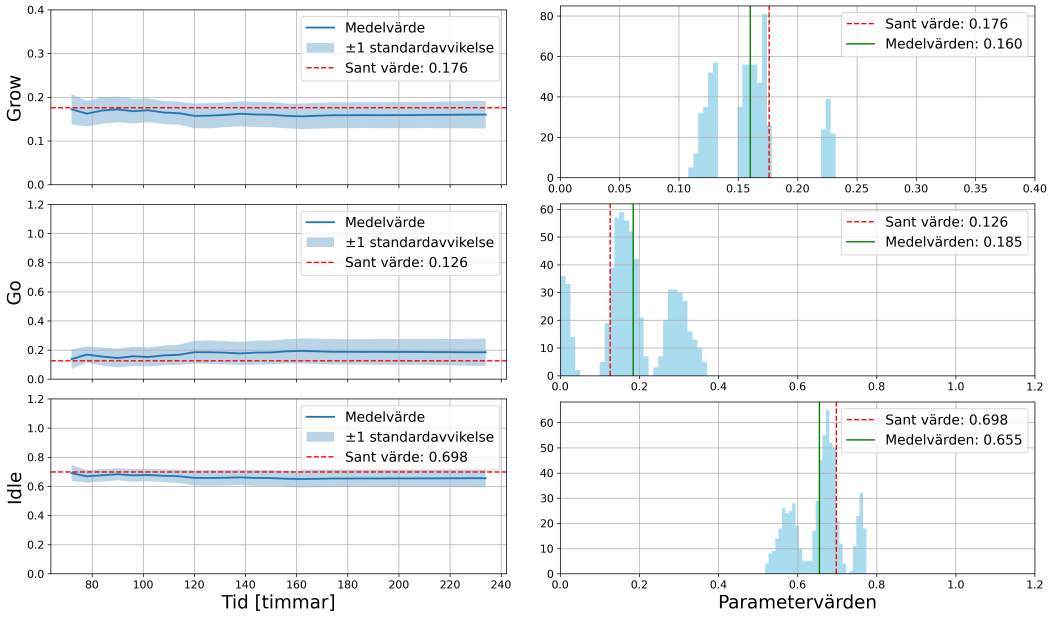
(c) (d)

Figur 17: Ovan härleddes via metriken för **fraktaldimension** på stokastiskt simulerade tumörer. Till vänster, visas grafer av medelvärdet av de accepterade parametrarna för parametersvep. Detta plottas mot tid för 28 bilder långa tidsserier (motsvarande timme 72 - 234 in-vitro). I rött grafas det sanna värdet på de underliggande parametrarna. Till höger, histogram över de accepterade parameterförslagen för timme 234. Även här är underliggande värden markerat i rött tillsammans med beräknat medelvärde i grönt. Det översta replikatet har underliggande parametrar $(p_{Grow}, p_{Go}, p_{Idle}) = (0.270, 0.477, 0.253)$. Det undre replikatet har istället underliggande parametrar $(p_{Grow}, p_{Go}, p_{Idle}) = (0.176, 0.126, 0.698)$.

A.7 Parameteruppskattning av simulerade replikat - Radiell snittdensitet



(a) (b)



(c) (d)

Figur 18: Ovan härleddes via metriken för **radiell snittdensitet** på stokastiskt simulerade tumörer. Till vänster, visas grafer av medelvärdet av de accepterade parametrarna för parametersvep. Detta plottas mot tid för 28 bilder långa tidsserier (motsvarande timme 72 - 234 in-vitro). I rött grafas det sanna värdet på de underliggande parametrarna. Till höger, histogram över de accepterade parameterförslagen för timme 234. Även här är underliggande värden markerat i rött tillsammans med beräknat medelvärde i grönt. Det översta replikatet har underliggande parametrar $(p_{Grow}, p_{Go}, p_{Idle}) = (0.270, 0.477, 0.253)$. Det undre replikatet har istället underliggande parametrar $(p_{Grow}, p_{Go}, p_{Idle}) = (0.176, 0.126, 0.698)$.

B Bildhantering

För implementeringen används följande Python-bibliotek: `numpy` för numeriska beräkningar, `cv2` från `OpenCV` för bildbehandling, `scipy.ndimage` för masscentrumberäkning och `skimage.measure` för formegenskaper.

Bildhanteringsprocessen inleds med att bilden läses och normaliseras, för en mer konsekvent hantering av pixelvärdena. Därefter beräknas medianintensiteten i bilden och denna subtraheras från varje pixelvärde. de inverterade pixelintensiteterna sparas för att i ett senare skede kunna kvantifiera tumörens densitet. För att reducera brus appliceras sedan ett Gaussiskt filter med en specifik standardavvikelse.

Nästa steg innebär att bilden omvandlas till binär med ett särskilt tröskelvärde där pixelvärden över detta får värdet 1 och övriga värdet 0. Därefter identifieras sammanhängande komponenter i bilden och komponenter med färre än 6000 pixlar tas bort. Detta för att ta bort mindre brus. För att hantera större brus som ibland förekommer i bilderna identifieras det objekt med lägst excentricitet, alltså det mest runda objektet, varpå endast detta behålls i bilden.

När den relevanta tumörkomponenten identifieras beräknas dess masscentrum, som används för att ta fram densitetsprofilen senare i denna algoritm men även som input till algoritmen för beräkning av fraktal dimension. För varje pixel beräknas sedan avstånd till centrum och lagras tillsammans med motsvarande intensitet. För att minska brus och skapa en jämnare representation av densiteten utförs en binning-process där pixeldata grupperas i intervaller baserade på avstånd. Genom att ta medelvärden inom dessa intervaller kan vi sammanfatta hur den totala densiteten av tumören ändras med avståndet från tumörens centrum. För att ytterligare jämna ut kurvan beräknas sedan ett glidande medelvärde (Simple Moving Average, SMA) av densiteten. SMA är en metod för att jämna ut variationer i en signal genom att ersätta varje datapunkt med medelvärdet av ett antal närliggande punkter. Det fungerar som ett filter som tar bort brus orsakat av slumpen och istället framhäver övergipande trender. Genom denna utjämning blir det lättare att identifiera tydliga övergångar i tumörens densitet, exempelvis var den invasiva delen övergår i bakgrund.

För att identifiera tumörens yttre gräns (invasiv radie) beräknas första och andra derivatan av SMA-kurvan, alltså den utjämnade densiteten. Den invasiva radien bestäms genom att identifiera den första punkten då en rad kriterier uppnåtts. Det första kriteriet är att första och andra derivatan sammanfaller (har mindre än 0.0005 differens). Det andra kriteriet är att SMA-densiteten i punkten är mindre än ett tröskelvärde (0.5). Det tredje och sista kriteriet är att den ursprungliga densiteten i punkten är mindre än 75% av den maximala densiteten hos tumören. Dessa kriterier och tillhörande gränsvärden har tagits fram genom att iterativt testa olika värden och utvärdera hur väl algoritmen sätter ut radierna på tumörer med olika utseenden (referera till bild här). Kärnradien fastställs genom att undersöka vid vilket avstånd intensiteten når 75% av det maximala densitetsvärdet, även det en empirisk gräns som valts baserat på observationer. I vissa fall (referera till bild med exempel), har tumören ingen tydlig "kärna"varpå det är rimligt att inte sätta ut en kärnradie. I de flesta av dessa fall kommer algoritmen sätta ut en kärnradie som är större än den invasiva radien. Vi kan alltså, i de flesta fall, lösa problemet genom att sätta kärnradien lika med invasiva radien ifall den överstiger den invasiva radien.

För att erhålla de slutgiltiga densitetskurvorna subtraheras densitetsvärdet vid tumörens slut från hela densitetsprofilen. Slutligen returneras de två radierna, en lista med densitetsvärden samt koordinater för masscentrum.

Algorithm 1 Pseudokod för beräkning av masscentrum, radier och densitet

```
procedure GetRadius(image_path)
    Load grayscale image
    Convert pixel values to float
    Compute median of image
    Subtract median from image
    Normalize im values to [0, 1]
    Invert image to get intensity
    Apply Gaussian filter (std = 4)
    Convert to binary with threshold 0.08
    Remove small objects (< 6000 pixels):
    for each component in binary image (except background) do
        if component area < 6000 then
            Remove component from binary image
        end if
    end for
    Find the most circular component:
    Label connected components in the binary image
    Compute region properties
    if multiple components exist then
        Select component with lowest eccentricity
        Keep only this component in binary image
    end if
    Compute center of mass
    Compute distance and intensity for each pixel:
    for each pixel in binary image do
        Compute distance from center of mass
        Store distance-intensity-pair
    end for
    Perform pixel binning (bin size = 4, max distance = 800)
    Compute Simple Moving Average (SMA) of density
    Compute first and second derivative of SMA
    Find first index where derivatives align and density is low:
    if such index exists then
        Store boundary index
    else
        Return (None, None)
    end if
    Define core radius as where density reaches 75% of max
    Adjust density values by subtracting the density value at the boundary
    Compute invasion radius → boundary_index × bin size
    Compute core radius
    if core radius > invasion radius then
        Set core radius = invasion radius
    end if
    return (core_radius, invasion_radius, density, center_of_mass, )
end procedure
```

C Källkod

C.1 Dynamics.cpp

```
#include "Dynamics.h"
#include "Space.h"

#include <cmath>
#include <stdexcept>
#include <queue>
#include <unordered_map>
#include <limits>
#include <iostream>

struct BFSNode {
    double g;  // BFS distance so far
    std::array<int,3> coord;
};

// Just used if we want a priority_queue (not strictly needed)
struct CompareNode {
    bool operator()(const BFSNode& a, const BFSNode& b) {
        return a.g > b.g; // min-first
    }
};

Dynamics::Dynamics(double goProb, double growProb,
                   double idleProb, double deathProb,
                   double moveDistance)
  : go_prob_(goProb),
    grow_prob_(growProb),
    idle_prob_(idleProb),
    death_prob_(deathProb),
    move_distance_(moveDistance)
{
    double sum = go_prob_ + grow_prob_ + idle_prob_ + death_prob_;
    if (std::fabs(sum - 1.0) > 1e-8) {
        throw std::runtime_error("Probabilities must sum to 1.0");
    }
    rng_.seed(std::random_device{}());
    dist01_ = std::uniform_real_distribution<double>(0.0, 1.0);
}

std::string Dynamics::chooseAction()
{
    double r = dist01_(rng_);
    if (r < go_prob_) {
        return "go";
    }
    r -= go_prob_;
    if (r < grow_prob_) {
        return "grow";
    }
    r -= grow_prob_;
    if (r < idle_prob_) {
        return "idle";
    }
    return "die";
}

void Dynamics::applyAction(const std::string& action,
                           const std::array<int,3>& coord,
                           Space& space)
{
    if (action == "go") {
        ruleBasedMove(coord, space);
    } else if (action == "grow") {
        ruleBasedGrow(coord, space);
    } else if (action == "die") {
        dieCell(coord, space);
```

```
67      }
68      // else "idle" => do nothing
69  }
70
71  void Dynamics::dieCell(const std::array<int,3>& coord, Space& space)
72  {
73      space.vacatePoint(coord[0], coord[1], coord[2]);
74  }
75
76  /**
77   * @brief BFS expansions up to 'move_distance_' steps from occupant,
78   *        then pick final location stochastically => exp(-cost).
79   */
80  void Dynamics::ruleBasedMove(const std::array<int,3>& startCoord, Space& space)
81  {
82      if (!space.isOccupied(startCoord[0], startCoord[1], startCoord[2])) {
83          return;
84      }
85      // gather expansions
86      auto expansions = gatherVacantExpansions(startCoord, space);
87      if (expansions.empty()) {
88          return;
89      }
90
91      double maxC = 0.0;
92      for (auto &r : expansions){
93          if (r.cost > maxC) maxC = r.cost;
94      }
95
96      double beta = (maxC > 0.0 ? 1.0 / maxC : 1.0);
97
98      // pick among expansions with probability ~ exp(- cost / normalization factor)
99      double sumW = 0.0;
100     std::vector<double> weights(expansions.size());
101     for (size_t i = 0; i < expansions.size(); i++){
102         double c = expansions[i].cost;
103         // c / 26 --> normalisera
104         double w = std::exp(-c*beta);
105         weights[i] = w;
106         sumW += w;
107     }
108     if (sumW <= 0.0) {
109         return; // fallback
110     }
111
112     double r = dist01_(rng_) * sumW;
113     int chosenIndex = 0;
114     for (size_t i=0; i<expansions.size(); i++){
115         if (r <= weights[i]) {
116             chosenIndex = (int)i;
117             break;
118         }
119         r -= weights[i];
120     }
121     auto chosenCoord = expansions[chosenIndex].coord;
122     // occupant moves
123     space.vacatePoint(startCoord[0], startCoord[1], startCoord[2]);
124     space.occupyPoint(chosenCoord[0], chosenCoord[1], chosenCoord[2]);
125 }
126
127 void Dynamics::ruleBasedGrow(const std::array<int,3>& startCoord, Space& space)
128 {
129     if (!space.isOccupied(startCoord[0], startCoord[1], startCoord[2])) {
130         return;
131     }
132     auto expansions = gatherVacantExpansions(startCoord, space);
133     if (expansions.empty()) {
134         return;
135     }
136
```

```
137     double maxC = 0.0;
138     for (auto &r : expansions){
139         if (r.cost > maxC) maxC = r.cost;
140     }
141 
142     double beta = (maxC > 0.0 ? 1.0 / maxC : 1.0);
143 
144     // pick stochastically
145     double sumW = 0.0;
146     std::vector<double> weights(expansions.size());
147     for (size_t i=0; i<expansions.size(); i++){
148         double c = expansions[i].cost;
149         double w = std::exp(-c*beta);
150         weights[i] = w;
151         sumW += w;
152     }
153     if (sumW<=0.0) {
154         return;
155     }
156     double r = dist01_(rng_) * sumW;
157     int chosenIndex = 0;
158     for (size_t i=0; i<expansions.size(); i++){
159         if (r <= weights[i]){
160             chosenIndex = (int)i;
161             break;
162         }
163         r -= weights[i];
164     }
165     auto chosenCoord = expansions[chosenIndex].coord;
166     // occupant grows => occupy chosenCoord, keep old occupant
167     space.occupyPoint(chosenCoord[0], chosenCoord[1], chosenCoord[2]);
168 }
169 
170 /**
171  * BFS expansions: find all vacant spots within BFS distance <= move_distance_
172  * from 'start'. We'll store cost = BFS distance + local density or similar.
173  */
174 std::vector<Dynamics::BFSResult>
175 Dynamics::gatherVacantExpansions(const std::array<int,3>& start,
176                                  const Space& space) const
177 {
178     std::vector<BFSResult> out;
179     if (!space.isOccupied(start[0], start[1], start[2])) {
180         return out;
181     }
182     int maxSteps = (int)std::floor(move_distance_);
183     if (maxSteps < 1) maxSteps = 1;
184 
185     std::queue<BFSNode> Q;
186 
187     // store best g so far
188     std::unordered_map<long,double> visited;
189 
190     // push start with g=0
191     BFSNode sn{ 0.0, start };
192     Q.push(sn);
193     visited[ coordKey(start) ] = 0.0;
194 
195     while (!Q.empty()) {
196         auto node = Q.front();
197         Q.pop();
198 
199         // neighbors
200         auto nbrs = getNeighbors(node.coord, space);
201         for (auto & nb : nbrs) {
202             // skip occupant
203             if (space.isOccupied(nb[0], nb[1], nb[2])) {
204                 continue;
205             }
206             // BFS distance
```

```
            double stepDist =
                std::sqrt(double((nb[0]-node.coord[0])*(nb[0]-node.coord[0]) +
                                                 (nb[1]-node.coord[1])*(nb[1]-node.coord[1])
                                                     +
                                                 (nb[2]-node.coord[2])*(nb[2]-node.coord[2])));
            double newG = node.g + stepDist;
            if (newG <= (double)maxSteps) {
                long key = coordKey(nb);
                if ( visited.find(key)==visited.end() || newG<visited[key] ) {
                    visited[key] = newG;
                    // record BFS expansions
                    BFSNode newNode{ newG, nb };
                    Q.push(newNode);

                    // compute cost = BFS distance + local density
                    double c = computeCost(newG, nb, space);
                    BFSResult r{ nb, c };
                    out.push_back(r);
                }
            }
        }
    }
    return out;
}

// cost = BFS distance + local density
double Dynamics::computeCost(double gVal, const std::array<int,3>& c,
                             const Space& space) const
{
    double dens = getSurroundingDensity(c, space);

    return gVal + dens;
}

// gather up to 26 neighbors
std::vector<std::array<int,3>> Dynamics::getNeighbors(const std::array<int,3>& c,
                                                      const Space& space) const
{
    std::vector<std::array<int,3>> out;
    out.reserve(26);
    for (int dx=-1; dx<=1; dx++){
        for (int dy=-1; dy<=1; dy++){
            for (int dz=-1; dz<=1; dz++){
                if (dx==0 && dy==0 && dz==0) continue;
                int nx = c[0] + dx;
                int ny = c[1] + dy;
                int nz = c[2] + dz;
                // check if in-bounds
                if (space.getCoordinate(nx,ny,nz) != nullptr) {
                    out.push_back({nx,ny,nz});
                }
            }
        }
    }
    return out;
}

// local density in 3x3x3 region
double Dynamics::getSurroundingDensity(const std::array<int,3>& c,
                                       const Space& space) const
{
    double dens = 0.0;
    for (int dx=-1; dx<=1; dx++){
        for (int dy=-1; dy<=1; dy++){
            for (int dz=-1; dz<=1; dz++){
                if (dx==0 && dy==0 && dz==0) continue;
                int nx = c[0]+dx;
                int ny = c[1]+dy;
                int nz = c[2]+dz;
                if (space.isOccupied(nx,ny,nz)) {
```

```
                    dens += 1.0;
                }
            }
        }
    }
    return dens;
}

long Dynamics::coordKey(const std::array<int,3>& c) const
{
    // simplistic hashing
    long x = (long)(c[0]) & 0xFFFF;
    long y = ((long)(c[1]) & 0xFFFF) << 16;
    long z = ((long)(c[2]) & 0xFFFF) << 32;
    return (x | y | z);
}
```

C.2 Simulation.cpp

```
#include "Simulation.h"
#include "Space.h"
#include "Dynamics.h"

#include <vector>
#include <algorithm> // std::shuffle
#include <numeric>
#include <iostream>
#include <cmath>
#include <ctime>
#include <random>

Simulation::Simulation(Space& space, Dynamics& dynamics, int iterationNr)
    : space_(space),
      dynamics_(dynamics),
      iteration_nr_(iterationNr),
      current_iteration_(0)
{
}

void Simulation::iter()
{
    // Gather all occupied coords
    std::vector<std::array<int,3>> coordsList;
    coordsList.reserve(space_.occupiedData().size());

    auto& occ = space_.occupiedData();
    auto& coords = space_.coordinatesData();

    for (size_t i = 0; i < occ.size(); ++i) {
        if (occ[i] == 1) {
            coordsList.push_back(coords[i]);
        }
    }

    // Shuffle
    std::shuffle(coordsList.begin(), coordsList.end(), rng_);

    int nToProcess = int(std::round(1.0 * coordsList.size()));
    if (nToProcess < 1) {
        // fallback, process all if too few
        nToProcess = int(coordsList.size());
    }
    coordsList.resize(nToProcess);

    // For each chosen coordinate, pick an action and apply
    for (auto& c : coordsList) {
        std::string action = dynamics_.chooseAction();
        dynamics_.applyAction(action, c, space_);
    }
```

```
343
344     current_iteration_++;
345 }
346
347 void Simulation::run()
348 {
349     for (int i = 0; i < iteration_nr_; ++i) {
350         iter();
351         // Debug/feedback: count how many are occupied
352         int occupiedCount = 0;
353         for (auto v : space_.occupiedData()) {
354             if (v == 1) occupiedCount++;
355         }
356         std::cout << "Iteration " << (i+1) << "/" << iteration_nr_
357                   << ": Occupied = " << occupiedCount << std::endl;
358     }
359 }
```