CHALMERS | GÖTEBORGS UNIVERSITET

Parameteridentifikation för matematisk modell beskrivande tumörcellers och makrofagers interaktioner

Identification of parameters describing tumour- macrophages interactions

Examensarbete för kandidatexamen i matematik vid Göteborgs universitet
Kandidatarbete inom civilingenjörsutbildningen vid Chalmers

Ellinor Sorpola Svenningsson
Jonatan Bertolozzi
Karl Olsson-Lalor
Regina Gustavsson

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

Parameteridentifikation för matematisk modell beskrivande tumörcellers och makrofagers interaktioner

Examensarbete för kandidatexamen i matematik inom Matematikprogrammet, inriktning Tillämpad matematik, vid Göteborgs universitet

Ellinor Sorpola Svenningsson

Kandidatarbete i matematik inom civilingenjörsprogrammet Kemiteknik med Fysik vid Chalmers

Karl Olsson-Lalor

Kandidatarbete i matematik inom civilingenjörsprogrammet Bioteknik vid Chalmers

Regina Gustavsson Jonatan Bertolozzi

Handledare: Larisa Beilina

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

Förord

Vi vill börja med att tacka vår handledare Larisa Beilina för sitt gedigna engagemang och det stöd hon har givit oss under hela kandidatarbetet. Under de regelbundna mötena med henne har vi fått flera bra förslag på både källor och utvecklingsmöjligheter i projektet. Larisa har även varit väldigt hjälpsam vid felsökning vilket möjliggjorde ett effektivt arbete. Vi vill dessutom rikta ett tack till biblioteket och Avdelningen för fackspråk och kommunikation för den handledning vi fått.

Under projektets gång har gruppen hållit veckovisa planeringsmöten för att strukturera arbetet och diskutera tidigare samt kommande arbetsuppgifter. Den övergripande planeringen för hela projektet har utgått från den planeringsrapport som skrevs i början av arbetet. Gruppen har kontinuerligt skrivit en dagbok som agerat dokumentation över gemensamt och individuellt arbete. Dessutom har all tid som lagts på projektet loggats individuellt.

Redan tidigt i projektet insågs det att gruppen väldigt effektivt kunde jobba gemensamt, just på grund av gruppstorleken och projektets utformning. Detta blev tydligt under inläsning av material från handledare då det resulterade i omfattande diskussioner vilket även var viktigt för gruppens förståelse av metoden. För att fortsatt kunna diskutera de problem som uppkom under projektets gång valde gruppen att sitta tillsammans under de, av Chalmers tekniska högskola, schemalagda tillfällen med undantag för schemakrockar och sjukdom.

Rapportskrivningen har pågått kontinuerligt under hela arbetet och alla gruppmedlemmar har bidragit till den resulterande texten. Detta gäller även för de delar där enbart några få benämns som huvudförfattare/delförfattare. I dessa avsnitt kan övriga gruppmedlemmar exempelvis ha bidragit med diskussion, omskrivningar eller delar av kod/text. Den huvudsakliga uppdelningen redovisas i följande tabell.

Ansvarsfördelning i rapportskrivandet

Författare	**Huvudansvar i avsnitt**	**Delförfattare i avsnitt**
Ellinor	1.1, 4.1, 7.3, B.1, C.1, C.2, D.3	4.2, 5.2, 7
Jonatan	2, 3.1, 5.1.1, 5.2, 7.2, C.3, D.2, D.3, E	1, 3, 7
Karl	Pp[1], SA[2], 4.2, 5.1.0[3], 5.2, 7.0[3], B.2, D.1	1.2, 7, C.1
Regina	2, 3.2, 6, 7.1, C.2, F	Pp[1], 1.0[3], 7

Ansvarsområden utanför rapporten fördelades enligt:

- Ellinor: Hon hade tillsammans med Karl huvudansvar i framtagandet av derivatorna som användes i Lagrangefunktionen. Hon hade även ansvar över att skriva in alla referenserna till rapporten och studerade samt felsökte tillhandahållen kod tillsammans med Jonatan.
- Jonatan: Han skrev koden för datautjämningen i MATLAB och studerade samt felsökte tillhandahållen kod tillsammans med Ellinor.
- Karl: Han hade tillsammans med Ellinor huvudansvar i framtagandet av derivatorna som användes i Lagrangefunktionen. Han skrev även koden för känslighetsanalysen och grunden för de explicita beräkningarna.
- Regina: Hon hade ansvar för vidareutveckling och felsökning av de explicita beräkningarna. Hon hade även huvudansvar för de figurer och grafer som visas i rapporten.

[1] Populärvetenskaplig presentation.
[2] Sammandrag; abstract.
[3] Inledningar av avsnitt betäcknas med ".0". Exempelvis 5.1.0 innebär avsnitt 5 fram till 5.1.

Populärvetenskaplig presentation

Kan matematik användas för att hjälpa bota cancer?

Cancer är en av de största folkhälsoutmaningarna i världen och har en stor inverkan på samhället, både ekonomiskt och livskvalitetsmässigt för de drabbade. Till följd av att människor lever längre, våra levnadsvanor samt förbättrad screeningteknik har antalet bekräftade fall ökat, och då det inte finns något botemedel görs det stora satsningar på att förbättra förutsättningarna för de utsatta.

På individnivå kan cancerns utvecklingsförlopp variera kraftigt från person till person även om det handlar om samma cancertyp. I en del fall kan tumörcellernas tillväxtförlopp bromsas in genom olika behandlingar som strålbehandling, immunterapi eller operationer. Det sker hela tiden mängder av ny forskning inom området för att hitta mer precisa och pålitliga metoder för att utveckla behandlingsstrategier och förutspå tumörtillväxt. Idag baseras ofta uppskattningarna av förloppet på klinisk erfarenhet och statistik, och även om prognostisering generellt har förbättrats med åren råder det fortfarande en stor variation i dess tillförlitlighet [10][11].

För att studera de variationer som finns i cancertillväxten mellan olika individer har en mängd matematiska modeller tagits fram [9] [6]. Dessa modeller framställer en bild av tillväxten av cancer- och immunförsvars-celler genom att med ekvationer beskriva hur de påverkar varandra. Dessa ekvationer består av olika termer vilka representerar specifika fenomen som sker i samspelet mellan de olika cellerna, exempelvis att immunförsvarsceller dödar cancerceller. Vissa av frekvenserna som dessa fenomen sker i har tagits fram experimentellt av biologer. Resterande frekvenser tros variera mellan individer vilket kan vara orsaken till att cancerns utvecklingsförlopp är svårt att förutse.

I stället för att experimentellt ta fram dessa frekvenser syftar detta projekt på att göra det matematiskt genom att analysera en redan framtagen modell. Genom att studera en del av tumörcellens tillväxtförlopp kan matematik användas för att visa vilka värden dessa frekvenser måste ha haft för att just det tillväxtförloppet skulle ske. När dessa frekvenser identifierats är förhoppningen att de i kombination med modellen ska kunna användas för att visa hur tumören kommer fortsätta utvecklas framåt i tiden. Denna metod hoppas då kunna ge mer tillförlitliga uppskattningar av förloppet än de som används i nuläget.

Tillvägagångssättet att hitta dessa frekvenser som främst används i detta projekt är något som kallas explicit beräkning. I de fall där man vet alla utom en frekvens kan denna metod användas för att beräkna denna. För att göra detta skrivs ekvationerna i modellen om på sådant sätt att den okända frekvensen enkelt kan beräknas numeriskt. För att bestämma flera frekvenser samtidigt kan något som heter konjugerade gradientmetoden användas vilket är något som påbörjats men kräver mer forskning. Denna metod går ut på att först gissa ett värde på frekvenserna och sedan optimera det. Genom att optimera gissningen flera gånger närmar sig de faktiska frekvensvärdena de frekvenserna som bäst lämpar sig för att beskriva den aktuella tillväxten av tumör- och immunförsvars-cellerna.

Kan då dessa frekvenser bestämmas? Ja, med en del begränsningar. Tillväxtförloppen studerades över ett tidsintervall på 20 dagar. Att beräkna en frekvens fungerar bra för de senare delarna av tidsintervallet, från dag 6 ungefär, men ger väldigt opålitliga resultat i början. Anledningen till detta är något som kallas avrundningsfel, ett fel som sker på grund av att beräkningsprogrammet inte klarar av tillräckligt små tal. Detta innebär att beräkningsprogrammet avrundar dessa tal tidigare och oftare än vad den borde, vilket kan leda till stora fel. På grund av detta fel i beräkningen av endast en frekvens förväntas liknande problem uppstå när flera frekvenser ska bestämmas samtidigt, det vill säga att de första dagarna inte ger pålitliga resultat. För att verifiera detta och eventuellt lösa de problem som uppstått krävs ytterligare utredning och forskning.

Sammandrag

Det finns sedan tidigare en mängd matematiska modeller som beskriver tumörcellers tillväxt och interaktioner med makrofager i kroppen. I projektet studerades en av dessa vilken beskriver tillväxtförloppet av malignt melanom i möss. Modellen innehåller ett antal parametrar varav några är oidentifierade. Huvudsyftet med projektet var att utveckla de redskap som krävs för att identifiera parametervärdena med målsättningen att kunna använda modellen för att förutspå tumörtillväxten i en individ.

För att först undersöka modellen och dess egenskaper, genomfördes en känslighetsanalys och en steady state-analys. Känslighetsanalysen visade hur de okända parametrarna påverkar tumörens samt makrofagernas tillväxt; steady state-analysen visade hur parametrarna påverkar modellens asymptotiska beteenden. I syfte att bestämma modellens oidentifierade parametrar beskrevs metoder för att göra detta explicit samt genom att använda en algoritm för att optimera en initialestimering. De explicita beräkningarna, som gjordes för en parameter i taget, studeras numerisk vilket resulterade i stora avvikelser början av tumörtillväxten.

Resultatet från de explicita beräkningarna ledde till hypotesen att samma avvikelser skulle uppstå även vid användning av optimeringsalgoritmen.

Abstract

There are a number of mathematical models that describe tumour cell growth and interactions with macrophages in the body. The project studies one of these already existing models, (3.1), describing the development of malignant melanoma in mice using parameters, some of which are unidentified. The aim of the project is to develop the tools required to identify these with the end goal that the model can be used to predict tumour growth in people.

Firstly, to study the model and its properties, sensitivity- and steady state-analyses were carried out. The sensitivity analysis showed how the unidentified parameters affect the outcome of tumour and macrophage growth; the steady state-analysis showed how the parameters affect the model's asymptotic behaviour. With the purpose to identify the parameters methods were described to either explicitly calculate these, or use an algorithm to optimize an initial estimate of them. The explicit calculations, which can only be done for one parameter at a time, were studied numerically which resulted in large deviations from the correct values at the early stages of tumour growth.

The results from the numerical studies led to the hypothesis that the same deviations would arise in use of the optimization algorithm.

Innehåll

1 Inledning

Cancer är en av de största folkhälsoutmaningarna som drabbat samhället vilket innebär att framsteg och ny forskning kan få betydande implikationer. En av de stora utmaningarna med cancer är att det finns stora variationer i hur den uttrycker sig och tumörcellernas tillväxt, vilket gör den väldigt svårbehandlad. Anledningarna till dessa variationer är att cancer är ett samlingsnamn för en mängd olika sjukdomar. Dessa definieras alla av olika sjukdomsmekanismer som varierar beroende på vilken vävnad tumören växer i. Detta medför att behandlingsplanen måste anpassas till varje enskild patient baserat på faktorer som cancertyp, sjukdomsstadium, patientens allmäntillstånd och personliga preferenser. Detta understryker behovet av ökad förståelse kring tumörtillväxt.

I och med framgångar inom immunologi, har flera matematiska modeller beskrivande interaktioner mellan immunförsvar och cancertumörer tagits fram [16]. Modellerna gör det möjligt att bland annat skapa nya empiriskt testbara hypoteser och kan hjälpa till att förutsäga tumörcellernas utveckling. Detta hade således kunnat agera som stöd vid diagnostisering och utvärdering av sjukdomsbild vid val av behandling. Fungerande modeller innebär även nya möjligheter för att utforska strategier för cancerbehandling vilka tidigare dömts vara för dyra, svårapplicerade eller riskfyllda. Till exempel går det på olika nivåer att ratificera valet av behandlingsmetod beroende på hur akut patientens tillstånd är och hur den förutspådda tillväxtbilden ser ut.

Projektet bygger på en redan befintlig tillväxtmodell för malignt melanom [6]. Modellen beskriver kinetiken för tumörtillväxt och formuleras som ett system av ordinära differentialekvationer, ODE, vilket innehåller flera parametrar med fysiologisk innebörd. Vissa av dessa parametrar har experimentellt framtagna värden, medan andra endast har en uppskattad storleksordning.

1.1 Syfte

Syftet med arbetet är att utveckla en metod för att identifiera parametrar i en förenklad modell beskrivande interaktionen mellan immunförsvaret och hudcancertumörer. Målet är även att påbörja implementering och utvärdering av metoden för den givna modellen.

1.1.1 Problemformulering

Den givna modellen producerar tillväxtkurvor för två olika typer av immunceller och för tumörcellerna givet värden på ett antal parametrar. Vissa av dessa parametrar är konstanta i tiden och har redan framtagna värden medan resterande enbart har uppskattade intervall. Det är de senare som önskas identifieras. Detta kan göras för en parameter i taget, om de andra antas ha kända värden, genom att göra en explicit beräkning av den oidentifierade parametern. För flera oidentifierade parametrar kan man utveckla en optimeringsalgoritm med hjälp av Tikhonovs reguleringsfunktional som ger de önskade parametervärdena.

Identifiering av parametrar kräver bra initialestimeringar. Valet av startvärden underlättas av en känslighetsanalys och en steady state-analys. Känslighetsanalysen kan användas för att hitta parametervärden som ger mindre skillnader i tillväxtkurvorna medan steady state-analysen kan användas för att hitta punkter där tillväxten sker långsamt. Steady state-analysen ger även en bild av systemets asymptotiska beteenden.

1.2 Samhälliga och etiska aspekter

De approximerade värdena för modellens olika parametrar bygger till stor del på data som är insamlad från experiment på möss [14]. Projektets användning av datan motiveras av att denna förhoppningsvis ska leda till optimering av cancerbehandling vilket i sin tur leder till mindre smärta för patienter samt färre dödsfall. Användningen av data insamlad från djurförsök är nödvändigt vid framtagande av modellen för att upprätthålla pålitligheten [31]. Detta i kombination med att det inte utförts några djurförsök i projektet anses tillräckligt som motivering.

2 Biologisk bakgrund

Biologiska system kan definieras som komplexa nätverk av sammankopplade processer och interaktioner mellan organismer, celler samt molekyler. Systemen är oftast dynamiska och påverkas av ett flertal faktorer. I följande avsnitt beskrivs de mest relevanta processerna och begreppen för att förstå modellen i projektet.

2.1 Tumörceller

Av en mängd olika anledningar kan det ske mutationer i cellernas DNA vilka i vissa fall kan leda till bildandet av tumörceller. Effekten av dessa blir att cellens mekanismer för reglering av celldelning och tillväxt slås ut, vilket resulterar i en obegränsad celltillväxt. Med tiden kan tumörtillväxt leda till att tumörcellerna invaderar och skadar intilliggande vävnader. Vidare kan tumörceller, genom metastasering, orsaka skada i avlägsna delar av kroppen via transport genom att via lymf- och blodkärl kolonisera frisk vävnad.

De immunogena egenskaperna hos en tumörcell påverkas av flera faktorer, såsom omfattningen av DNA-mutationer, de specifika förhållandena i tumörens mikromiljö samt tumörcellernas interaktioner med kroppens immunceller. Tumörceller kan klassificeras ytterligare beroende på deras immunogenicitet, det vill säga i vilken grad de utlöser immunsvar eller ej. Tumörcellerna delas därmed upp i två klasser: de som lätt känns igen av immunförsvaret och de som undgår detektion [29]. Därför är det fortfarande en utmaning att utveckla målinriktade terapier för att bekämpa tumörceller med låg immunogenicitet.

2.1.1 Hudcancerceller

Under de senaste årtiondena har antalet diagnostiserade med hudcancer ökat och det är idag en av de vanligaste cancerformerna [15]. Det finns flera olika sorters hudcancer vilka brukar delas in i två huvudgrupper: icke-melanocytisk hudcancer och malignt melanom. Malignt melanom uppstår då tumörcellerna bildas i hudens melanocyter, pigmentsproducerande celler, medan icke-melanocytisk hudcancer bildas i andra sorters hudceller. Icke-melanocytisk hudcancer är vanligare men malignt melanom är ofta mer aggressivt. Malignt melanom visar sig ofta som pigmentfläckar i huden och det är ofta tillräckligt att operera bort dessa hudförändringar [13]. I en del fall där detta är otillräckligt är det nödvändigt att använda andra behandlingar, exempelvis strålning.

2.2 Makrofager

Immunförsvaret kan delas in i två delar: det adaptiva (specifika) immunförsvaret och det medfödda (ospecifika) immunförsvaret [27]. Det adaptiva immunförsvaret består bland annat av T-celler och B-celler, vilka försvarar kroppen med hjälp av specifika antikroppar vilket gör det väldigt precist, till skillnad från det medfödda immunförsvaret. Vid infektion agerar det medfödda immunförsvaret som kroppens första försvar, detta inkluderar bland annat neutrofiler och makrofager. Makrofager är immunförsvarsceller vars viktigaste funktion är att skydda kroppen mot okända molekyler och patogener, främmande smittoämnen [12].

M1- och M2-makrofagerna är två olika sorters celler i det medfödda immunförsvaret vilka har motsatta roller kopplat till tumörtillväxten [26]. Genom att reglera hur immuncellerna svarar på närvaron av T-hjälparcellerna Th1 och Th2 hindrar M1 spridningen av patogena celler medan M2 agerar främjande. Förekomsten av T-hjälparcellerna reglerar i sin tur även makrofagernas aktivitet där Th1 inaktiverar M2 och Th2 inaktiverar M1.

2.2.1 T-hjälparceller

Th1 och Th2 är så kallade T-hjälparceller vilket är en grupp T-celler, eller T lymfocyter som de även kallas [2]. På T lymfocyternas membran finns olika receptorprotein som känner igen specifika smittoämnen. Speciellt för gruppen T-celler som Th1 och Th2 tillhör, är att de alla har samma sorts receptorprotein [8]. Th1 och Th2 producerar olika cytokiner, signalmolekyler, som reglerar immunförsvarets funktion. M1 makrofagerna förstärker Th1-cytokinernas aktiverande påverkan medan Th2-cytokinerna, som istället gynnas av M2, skapar en antiinflammatorisk effekt.

M1-makrofager utsöndrar cytokiner som förstärker aktiveringen av Th1-celler, som i sin tur aktiverar M1-makrofagerna. Denna positiva återkoppling förstärker immunsvaret mot patogener.

2.2.2 Polarisering och repolarisering

I den miljön som omger en tumör, känd som tumörens mikromiljö, återfinns som bekant både M1- och M2-makrofager. Polarisering beskriver processen genom vilken makrofager, även tumörassocierade makrofager, erhåller en distinkt funktionell fenotyp som svar på stimuli från mikromiljön. Alltså specialiseras cellernas funktion och beteende [24]. För de tumörassocierade makrofagerna resulterar polarisering i huvudfenotyperna: M1 och M2. Dessa fenotyp uppvisar som nämnts i avsnitt 2.2 olika egenskaper och har motsatta roller i mikromiljön [23]. Notera att polarisering inte är en binär process med distinkta utfall utan bör istället betraktas som ett kontinuum med kontinuerliga tillstånd som makrofagerna kan anta baserat på stimuli och signaler i deras omgivning [23].

Repolarisering beskriver istället processen då den funktionella fenotypen hos en cell förändras från ett tillstånd till ett annat. I sammanhanget för tumör associerade makrofager innebär repolarisering att fenotypen skiftar från M1 till M2 eller vice versa. Repolariseringshastigheten påverkas av en rad olika faktorer och kan variera över tid och även i olika delar av tumören.

3 Modellbeskrivning

Modellen som används i projektet beskriver interaktionen mellan tumörceller samt M1- och M2-makrofager. Denna är tagen från [6] vilken baseras på modellen från [9] med anpassning för vissa avgränsningar.

3.1 Modellavgränsningar

Avgränsningarna i [6] behandlar bland annat de ekvationer i [9] som modellerar kinetiken för Th1- och Th2-celler, vilka inte inkluderas. Modellen tar inte heller hänsyn till skillnaden mellan olika sorters tumörceller eller det faktum att de reagerar olika på cytokiner. Eftersom det inte görs någon urskiljning mellan tumörceller med olika immunogenicitet kan även uttrycken för mutation mellan de olika celltyperna bortses från.

Värt att nämna är även att polarisering är en komplicerad process och är därav svår att simulera. I [9] har detta därav förenklats så att representationen blir binär. Repolariseringen har sedan förenklats ytterligare i [6] så att bara en variant tas hänsyn till.

Utöver dessa förenklingar antas i detta projekt att alla okända parametrar som önskas identifieras har konstanta värden. Detta till skillnad från att de skulle vara funktioner av tiden som de beskrivs i den ursprungliga formuleringen av problemet.

Alla förenklingar av det biologiska systemet syftar till att omformulera ekvationssystemet för att på så vis underlätta framtagandet av parametervärden.

3.2 System av differentialekvationer och parameterbeskrivning

Med anpassning för modellavgränsningarna ges följande system av differentialekvationer

$$\frac{dx_T}{dt} = rx_T\left(1 - \frac{x_T}{\beta_T}\right) - d_{M1}x_{M1}x_T + d_{M2}x_{M2}x_T, \tag{3.1a}$$

$$\frac{dx_{M1}}{dt} = a_{t1}x_Tx_{M1}\left(1 - \frac{x_{M1} + x_{M2}}{\beta_M}\right) - \delta_{M1}x_{M1} - k_{12}x_{M1}x_T, \tag{3.1b}$$

$$\frac{dx_{M2}}{dt} = a_{t2}x_Tx_{M2}\left(1 - \frac{x_{M1} + x_{M2}}{\beta_M}\right) - \delta_{M2}x_{M2} + k_{12}x_{M1}x_T. \tag{3.1c}$$

Variablerna x_T, x_{M1} och x_{M2} betecknar densiteten av tumörceller, M1 makrofager respektive M2 makrofager, vars tillväxt beskrivs av (3.1). Parametrarna i (3.1) beskrivs i tabell 1.

Tabell 1: Definition av parametrar.

Parameter	Beskrivning	Parametervärden	Ref.
r	Tumörens spridningshastighet	0,93 $\frac{1}{\text{dag}}$	[14]
β_T	Bärkapacitet för tumörer	$3 \cdot 10^9$ celler	[18] [20]
d_{M1}	Hastigheten M1 makrofager dödar tumörceller	$10^{-11} - 10^{-7} \frac{1}{\text{dagar} \cdot \text{celler}}$	[6]
d_{M2}	Hastigheten M2 makrofager bidrar till tumörtillväxt	$10^{-12} - 10^{-8} \frac{1}{\text{dagar} \cdot \text{celler}}$	[6]
a_{t1}	Medelhastigheten för aktivering och rekrytering av M1 makrofager till tumören	$10^{-10} - 10^{-6} \frac{1}{\text{dagar} \cdot \text{celler}}$	[6]
a_{t2}	Medelhastigheten för aktivering och spridning av M2 makrofager vid tumören	$10^{-12} - 10^{-8} \frac{1}{\text{dagar} \cdot \text{celler}}$	[6]
β_M	Bärkapacitet för makrofager	$9 \cdot 10^8$ celler	[20]
δ_{M1}	Dödshastighet för M1 makrofager	0,173 $\frac{1}{\text{dag}}$	[17] [21]
δ_{M2}	Dödshastighet för M2 makrofager	0,173 $\frac{1}{\text{dag}}$	[17] [21]
k_{12}	Hastigheten M1 makrofager repolariseras till M2 makrofager i närvaro av cytokiner och andra tillväxtfaktorer producerade av tumörcellerna	$10^{-12} - 10^{-7} \frac{1}{\text{dagar} \cdot \text{celler}}$	[6]

Relationerna i (3.1) har tagits fram efter vissa antaganden som förklaras kort nedan.

- För att ta hänsyn till den minskade tumörtillväxten vid stora tumörcellspopulationer i (3.1a) så antas den ha en logistisk tillväxthastighet, r, upp till bärkapaciteten för tumörer, β_T. Tumörcellerna kommer även elimineras av M1 makrofagerna med en hastighet d_{M1} samtidigt som M2 makrofagerna inducerar tillväxt med en hastighet d_{M2}.
- Ekvation (3.1b) beskriver istället tillväxtdynamiken av M1 makrofager vilka, vid tumören, aktiveras och ansamlas med hastigheten a_{t1} upp till makrofagernas bärkapacitet, β_M. Cellernas halveringstid är $\frac{1}{\delta_{M1}}$ och de repolariseras till M2 makrofager med en hastighet k_{12}.
- Tillväxtdynamiken av M2 makrofagerna beskrivs av (3.1c), där a_{t2} betecknar aktivering och ansamling av M2 makrofager vid tumören. Tillsammans med M1 makrofagerna antas de ha delat bidrag till det totala makrofagbeståndet. $\frac{1}{\delta_{M2}}$ betecknar M2 makrofagernas halveringstid.

De oidentifierade parametrarna samlas i parametervektorn $\boldsymbol{\alpha} = \left(d_{M1}, d_{M2}, a_{t1}, a_{t2}, k_{12}\right)$ med intervall enligt tabell 2.

Tabell 2: Okända parametrar samt deras uppskattade parameterintervall.

Parameter	d_{M1}	d_{M2}	a_{t1}	a_{t2}	k_{12}
Parameterintervall	10^{-11} - 10^{-7}	10^{-12} - 10^{-8}	10^{-10} - 10^{-6}	10^{-12} - 10^{-8}	10^{-12} - 10^{-7}
Mellanvärde	10^{-9}	10^{-10}	10^{-8}	10^{-10}	$5 \cdot 10^{-10}$

4 Modellanalys

Eftersom det saknas analytiska lösningar ekvationerna i modellen är det lämpligt att undersöka problemet för att analysera hur val av parametrar och initialvärden påverkar lösningen för olika tidsintervall.

4.1 Analys av steady states

För att få en uppfattning om lösningskurvornas beteenden längre fram i tiden studerar man steady state, en punkt där tidsderivatorna för $\boldsymbol{x}(t)$ är noll.

Det finns flera egenskaper steady state kan ha, bland annat kan de vara instabila, stabila eller asymptotiskt stabila. Låt $f(\boldsymbol{x}) = \dot{\boldsymbol{x}} := \frac{d\boldsymbol{x}}{dt}$ och låt $f : G \to \mathbb{R}^N$ där f är kontinuerlig och G är en öppen icke-tom mängd i $\mathbb{R}^N$. För definitionen av stabil och asymptotiskt stabil följer nedan modifierade varianter av motsvarande definitioner (5.1 och 5.14) ur [22].

Definition 4.1. *[22] Ett steady state $\boldsymbol{x}^*$ sägs vara stabilt om det, för alla $\epsilon > 0$, finns ett $\delta > 0$ sådant att, för varje maximal lösning $\boldsymbol{x} : I \to G$ till $f(\boldsymbol{x})$ med $\boldsymbol{x}(0) = \xi$, $0 \in I$ och $||\boldsymbol{x}(0) - \boldsymbol{x}^*|| \leq \delta$ gäller att*

$$||\boldsymbol{x}(t) - \boldsymbol{x}^*|| \leq \epsilon \quad \forall t \in I \cap \mathbb{R}_+.$$

Om ett steady state inte är stabilt sägs det vara instabilt.

Definition 4.2. *[22] Ett steady state $\boldsymbol{x}^*$ sägs vara attraherande om det finns ett $\delta > 0$ sådant att, för varje $\boldsymbol{\xi} \in G$ med $||\boldsymbol{\xi} - \boldsymbol{x}^*|| \leq \delta$, gäller att lösningen $\boldsymbol{x}(t) = \varphi(t, \boldsymbol{\xi})$ till $f(\boldsymbol{x})$ med $\boldsymbol{x}(0) = \boldsymbol{\xi}$ existerar på $\mathbb{R}_+$ och $\varphi(t, \boldsymbol{\xi}) \to \boldsymbol{x}^*$ då $t \to \infty$. Vi säger att $\boldsymbol{x}^*$ är ett asymptotiskt stabilt steady state om det både är stabilt och attraherande.*

Definition 4.2 innebär att en lösningskurva som startar i en punkt tillräckligt nära ett asymptotiskt stabilt steady state, kommer att närma sig detta steady state.

De fall av steady state som identifierats till (3.1) presenteras i [6] men har korrigerats och samlats i följande sats.

Sats 4.1. *Alla möjliga steady state till* (3.1) *med $\boldsymbol{x}(t) \geq 0$ och $\boldsymbol{\alpha} > 0$ kan skrivas på någon av följande former:*

1. $\boldsymbol{x}^* = (0, 0, 0)$,
2. $\boldsymbol{x}^* = (\beta_T, 0, 0)$,
3. $\boldsymbol{x}^* = \left(x_T^*, 0, \frac{r}{d_{M2}}\left(\frac{x_T^*}{\beta_T} - 1\right)\right)$,
   där $x_T^* = \frac{d_{M2}\beta_T\beta_M}{2a_{t2}r}\left[\left(a_{t2} + \frac{a_{t2}r}{d_{M2}\beta_M}\right) \pm \sqrt{\left(a_{t2} + \frac{a_{t2}r}{d_{M2}\beta_M}\right)^2 - \frac{4a_{t2}r\delta_{M2}}{\beta_T\beta_M d_{M2}}}\right]$,
4. $\boldsymbol{x}^* = (x_T^*, x_{M1}^*, x_{M2}^*)$,
   där sambandet mellan x_T^*, x_{M1}^* och x_{M2}^* beskrivs av följande uttryck:

$$x_{M1}^* = \frac{r}{d_{M1}}\left(1 - \frac{x_T^*}{\beta_T}\right) + \frac{d_{M2}}{d_{M1}}x_{M2}^*,$$

$$a_{t2}x_{M2}^*\left(\frac{\delta_{M1}}{a_{t1}} + \frac{a_{t2}}{a_{t1}}x_T^*\right) - \delta_{M2}x_{M2}^* + a_{t2}x_T^*\left(\frac{d_{M2}x_{M2}^*}{d_{M1}} + \frac{r}{d_{M1}}\left(1 - \frac{x_T^*}{\beta_T}\right)\right) = 0.$$

För bevis se appendix B.1.

För att avgöra om ett steady state är instabilt används följande modifierade version av sats 5.31 från [22].

Sats 4.2. *[22] Låt f vara differentierbar i punkten $0 \in G$ och låt $f(0) = 0$. Om matrisen*

$$A := (Df)(0) = ((\partial_j f_i)(0))_{1 \leq i,j \leq N}$$

har ett egenvärde med positiv realdel så är 0 ett instabilt steady state.

Med hjälp av sats 4.2 kan man komma fram till följande sats om stabiliteten hos de två första steady state.

Sats 4.3. *De första två steady state som identifierats i sats 4.1 har klassificerats som följande:*

1. $\boldsymbol{x}^* = (0, 0, 0)$ *är instabil.*

2. $\boldsymbol{x}^* = (\beta_T, 0, 0)$ *är instabil.*

För bevis se appendix B.1. På grund av komplexiteteten av steady state 3 och 4 i sats 4.1 görs ingen motsvarande analys av deras stabilitet.

Det vi kan säga om steady state $\boldsymbol{x}^* = (0, 0, 0)$ är att alla lösningar som startar i punkten $\boldsymbol{x}_0 = (x_{T0}, x_{M10}, x_{M20})$ tillräckligt nära $\boldsymbol{x}^*$ kommer röra sig bort från $\boldsymbol{x}^*$ om $x_{T0} > 0$ eftersom tidsderivatan av x_T då blir positiv i (3.1). Om $x_{T0} = 0$ kommer lösningskurvan gå mot $\boldsymbol{x}^*$ eftersom tidsderivatorna av x_{M1} och x_{M2} då blir negativa.

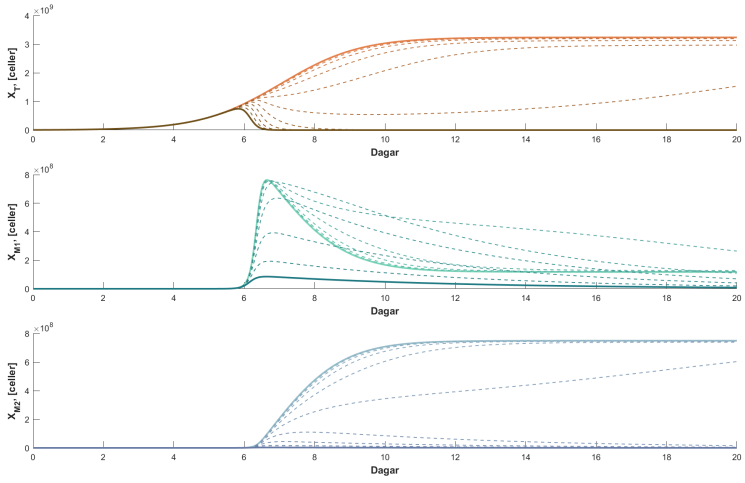
Om $\boldsymbol{x}^* = (\beta_T, 0, 0)$ kan vi säga att lösningskurvan som startar i punkten $x_0 = (x_{T0}, 0, 0)$ där $x_{T0} > 0$, kommer att gå mot $\boldsymbol{x}^*$. Detta kan vi se eftersom tidsderivatorna av x_{M1} och x_{M2} i (3.1) blir noll samtidigt som tidsderivatan av x_T är positiv tills det att lösningskurvan närmar sig punkten $\boldsymbol{x}^*$ där derivatan är noll.

Det finns en möjlighet att punkter på någon av formerna 3 eller 4 beskrivna i sats 4.1 är asymptotiskt stabila steady state och att lösningskurvor därmed rör sig mot någon sådan punkt. Detta undersöks inte närmare i denna rapport.

4.2 Modellens känslighet till variation av parametrar

För att visualisera parametrarnas inverkan på den numeriska lösningen av modellen togs ett flertal lösningskurvor fram för olika värden på de oidentifierade parametrarna.

Figurerna 4.1, 4.2, 4.3, 4.4 och 4.5 fås genom att variera enbart en parameter i taget och låta de resterande parametrarna, med konstanta värden, anta deras respektive mellanvärde enligt tabell 2. Resultatet visar känsligheten för variation av endast en parameter och kan ge insikt i parametrarnas signifikans i modellen. För färggradienterna som används i figurerna representerar de mörkare tonerna högre parametervärden gentemot de ljusa och de heldragna linjerna motsvarar parameterintervallets ändvärden.

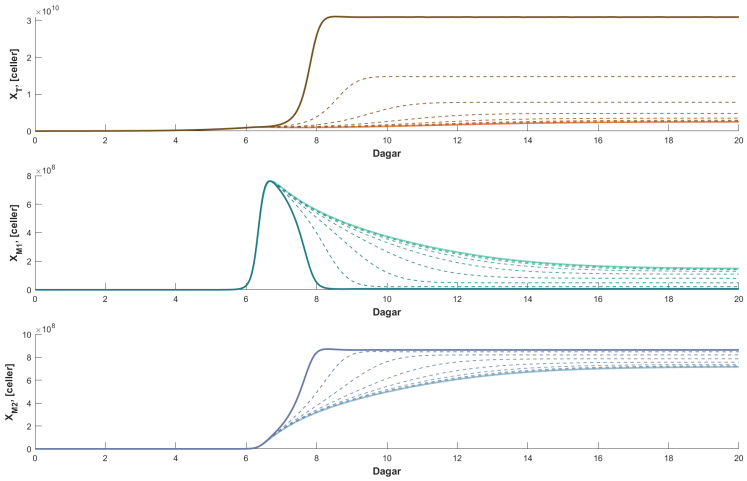


Figur 4.1: Variation av parameter d_{M1}. Linjernas färg är beroende av parametervärdet på d_{M1} där de ljusare har lägre värden och de mörkare har högre värden. Valda parametervärden är $1,00 \cdot 10^{-11}$; $2,00 \cdot 10^{-11}$; $5,00 \cdot 10^{-11}$; $1,20 \cdot 10^{-10}$; $2,80 \cdot 10^{-10}$; $6,60 \cdot 10^{-10}$; $1,52 \cdot 10^{-9}$; $3,51 \cdot 10^{-9}$; $8,11 \cdot 10^{-9}$; $1,87 \cdot 10^{-8}$; $4,33 \cdot 10^{-8}$ och $1,00 \cdot 10^{-7}$.

Från tabell 1 ges det att d_{M1} är hastigheten M1 makrofager dödar tumörceller. Enligt (3.1a) är dess inverkan på tumördensitetens förändringshastighet linjärt beroende av M1 makrofagernas densitet, det vill säga antalet makrofagceller.

Effekten av låga värden på parametern syns tydligt i figur 4.1 där även vid höga densiteter av M1 makrofager sprider sig tumörcellerna snabbare än de dödas av. Detta gör att även fast densiteten

av M1 makrofager når ett tidigt maximum påverkar detta i stort sett inte förändringshastigheten av tumörcellerna. Tumörerna ges då möjlighet att fortsätta sprida sig vilket ökar antalet M1 makrofager som repolariseras och därmed minskar deras densitet. Repolariserandet av M1 makrofager till M2 makrofager gör att mot slutet av det studerade tidsintervallet blir densiteten av M1 makrofagerna låg och M2 makrofagerna hög. Med högre parametervärden behövs endast ett fåtal M1 makrofager för att markant påverka densiteten av tumörcellerna. Detta gör att så fort densiteten av M1 makrofagerna börjar öka dör tumörcellerna vilket hindrar densiteterna av M1- och M2 makrofagerna från att fortsätta öka.



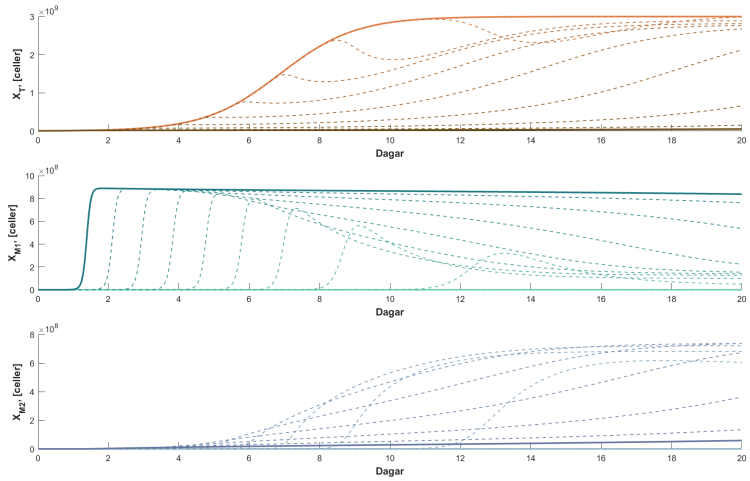
Figur 4.2: Variation av parameter d_{M2}. Linjernas färg är beroende av parametervärdet på d_{M2} där de ljusare har lägre värden och de mörkare har högre värden. Valda parametervärden är $1,00{\cdot}10^{-12}$; $2,00{\cdot}10^{-12}$; $5,00{\cdot}10^{-12}$; $1,20{\cdot}10^{-11}$; $2,80{\cdot}10^{-11}$; $6,60{\cdot}10^{-11}$; $1,52{\cdot}10^{-10}$; $3,51{\cdot}10^{-10}$; $8,11{\cdot}10^{-10}$; $1,87 \cdot 10^{-9}$: $4,33 \cdot 10^{-9}$ och $1,00 \cdot 10^{-8}$.

I tabell 1 står det att d_{M2} är hastigheten M2 makrofager bidrar till tumörtillväxten. Enligt (3.1a) är dess inverkan på tumördensitetens förändringshastighet linjärt beroende av M2 makrofagernas densitet.

För höga värden på d_{M2} bidrar densiteten av M2 mycket till tumördensitetens förändringshastighet, vilket innebär att när M2 makrofagernas densitet ökar gör även förändringshastigheten för tumördensiteten det. Detta gör i sin tur att bärkapaciteten snabbt uppnås för tumören och även då för makrofagerna, se figur 4.2. När bärkapaciteten är nådd kommer densiteten av M1 makrofager att minska på grund av repolarisation. För tillräckligt höga värden på d_{M2} kommer tumördensiteten överskrida bärkapaciteten på grund av det stora bidrag den får från M2 makrofagerna. Låga val av parametern gör att tumördensitetens förändringshastighet knappt påverkas av M2 markofagerna. Detta ger att tumören växer långsammare och inte nödvändigtvis når lika höga värden. Dess inverkan på makrofagernas förändringshastigheter blir då mycket lägre. Många av de ovanstående situationerna leder till att systemet uppnår steady state.

Parametern a_{t1} är medelhastigheten för aktivering och rekrytering av M1 makrofager till tumören, se tabell 1.

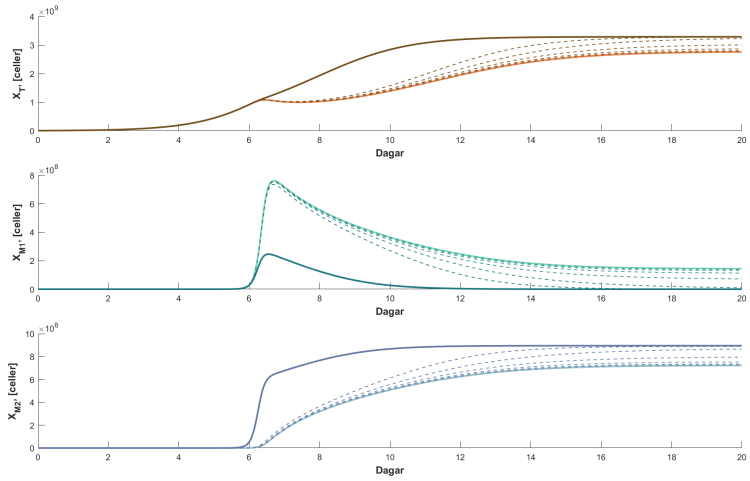
För höga värden av parametern ökar densiteten av M1 makrofager väldigt tidigt, se figur 4.3. Då förändringshastigheten av tumördensiteten innehåller en term med ett negativt linjärt beroende till M1 makrofagernas densitet, se (3.1a), gör detta att tumördensiteten begränsas. Vid tillräckligt höga värden kommer M1 makrofagerna uppnå hela den gemensamma bärkapaciteten vilket begränsar M2 makrofagernas möjlighet att öka, vilket inte kompenseras för tillräckligt med repolariseringen. Lägre värden av a_{t1} ger tumördensiteten och M2 makrofagernas densitet möjlighet att öka. M1 makrofagernas densitet ökar nu inte snabbt nog för att kompensera för repolariseringen. För tillräckligt låga värden på parametern tillåts tumören växa näst intill obehindrat. Då enbart rekrytering, det vill säga utan repolarisering, av M2 makrofager är långsammare än dess dödshastighet kommer densiteten av M2 makrofager begränsas.



Figur 4.3: Variation av parameter a_{t1}. Linjernas färg är beroende av parametervärdet på a_{t1} där de ljusare har lägre värden och de mörkare har högre värden. Valda parametervärden är $1.00 \cdot 10^{-10}$, $2.00 \cdot 10^{-10}$, $5.00 \cdot 10^{-10}$, $1.20 \cdot 10^{-9}$, $2.80 \cdot 10^{-9}$, $6.60 \cdot 10^{-9}$, $1.52 \cdot 10^{-8}$, $3.51 \cdot 10^{-8}$, $8.11 \cdot 10^{-8}$, $1.87 \cdot 10^{-7}$, $4.33 \cdot 10^{-7}$ och $1.00 \cdot 10^{-6}$.

Tabell 1 ger att a_{t2}, vars variation illustreras i figur 4.4, är medelhastigheten för aktivering och spridning av M2 makrofager vid tumören.

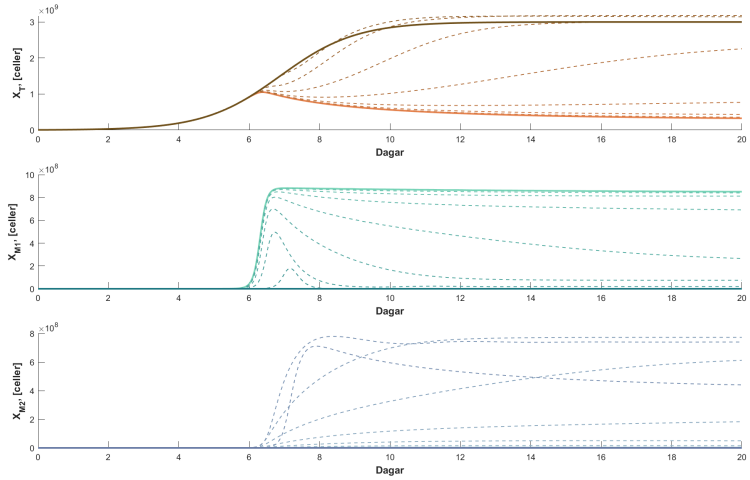
För höga värden på a_{t2} fyller M2 makrofagerna snabbt bärkapaciteten för makrofager vilket begränsar ökningen av M1 makrofager. Notera dock att majoriteten av M2 makrofagernas tillväxt kommer från repolarisering av M1 makrofager, vilket är varför ökning inte börjar lika tidigt som för M1 makrofager med höga värden på a_{t1}. Den lägre densiteten av M1 makrofager och högre densiteten av M2 makrofager gör att tumördensiteten växer snabbare. Låga värden på a_{t2} ger istället att densiteten av M1 makrofager kan anta större värden och nå ett maximum innan repolarisering leder till en minskning. Här är repolarisering nästan det enda som bidrar till ökning av M2 makrofagernas densitet.



Figur 4.4: Variation av parameter a_{t2}. Linjernas färg är beroende av parametervärdet på a_{t2} där de ljusare har lägre värden och de mörkare har högre värden. Valda parametervärden är $1,00 \cdot 10^{-12}$; $2,00 \cdot 10^{-12}$; $5,00 \cdot 10^{-12}$; $1,20 \cdot 10^{-11}$; $2,80 \cdot 10^{-11}$; $6,60 \cdot 10^{-11}$; $1,52 \cdot 10^{-10}$; $3,51 \cdot 10^{-10}$; $8,11 \cdot 10^{-10}$; $1,87 \cdot 10^{-9}$; $4,33 \cdot 10^{-9}$ och $1,00 \cdot 10^{-8}$.

För låga värden på k_{12}, repolariseringshastigheten, sker ingen repolarisering vilket gör att densiteten av M1 makrofager ökar utan påverkan av repolariseringen, se figur 4.5. M2 ökar väldigt långsamt då repolarisering är den faktor som gör att de växer snabbare än de dör. Detta leder till att tumörtillväxten hindras av M1 makrofager samtidigt som den får minimalt med hjälp från M2 makrofager.

Höga värden på k_{12} innebär att densiteten av M1 makrofagerna minskar snabbt efter att de nått sitt maximum samtidigt som densiteten av M2 makrofagerna ökar. För tillräckligt höga värden repolariseras M1 makrofagerna nästan direkt efter att de aktiverats eller rekryterats. Detta gör att densiteten av M1 makrofagerna aldrig tillåts öka. Den låga densiteten av M1 makrofagerna gör att repolariseringen inte räcker som kompensation för att M2 makrofagernas dödshastighet är högre än deras rekryteringshastighet. Avsaknaden av M1 makrofager gör även att tumören tillåts växa näst intill obehindrat. När tumördenisteten blir tillräckligt stor ökar antalet M2 makrofager, dock väldigt långsamt. Detta görs tydligt från (3.1c) där förhållandet mellan densiteterna av tumören och M2 makrofagerna, a_{t2} samt δ_{M2} visas.



Figur 4.5: Variation av parameter k_{12}. Linjernas färg är beroende av parametervärdet på k_{12} där de ljusare har lägre värden och de mörkare har högre värden. Valda parametervärden är $1,00 \cdot 10^{-11}$; $2,00 \cdot 10^{-11}$; $5,00 \cdot 10^{-11}$; $1,20 \cdot 10^{-10}$; $2,80 \cdot 10^{-10}$; $6,60 \cdot 10^{-10}$; $1,52 \cdot 10^{-9}$; $3,51 \cdot 10^{-9}$; $8,11 \cdot 10^{-9}$; $1,87 \cdot 10^{-8}$; $4,33 \cdot 10^{-8}$ och $1,00 \cdot 10^{-7}$

5 Parameteridentifikationsproblem, PIP

Beroende på hur många parametrar som är oidentifierade kan olika metoder användas. I fallet då enbart en parameter är oidentifierad kan denna beräknas explicit. I resterande fall kan en optimeringsalgoritm användas.

5.1 Explicit beräkning av en oidentifierad parameter

För att explicit beräkna värdet på en oidentifierad parameter används (3.1) för att ta fram parametervärdet vid diskreta tidpunkter. Detta görs genom att diskretisera tidsderivatan och bryta ut den oidentifierade parametern från någon av (3.1a), (3.1b) eller (3.1c)

Låt värdena på alla parametrar utom en samt $\boldsymbol{x}(t) = \big(x_T(t), x_{M1}(t), x_{M2}(t)\big)$ vara kända för en viss tidspartition T_h med h diskreta punkter och steglängd τ. Då kan den oidentifierade parametern beräknas explicit vid tidpunkten t_i, $i = 1, 2, \ldots, h$, med en av de följande ekvationerna

$$d_{M1}(t_i) \approx \frac{-\frac{x_T(t_{i+1})-x_T(t_{i-1})}{2\tau} + rx_T(t_i)\cdot(1-\frac{x_T(t_i)}{\beta_T}) + d_{M2}x_{M2}(t_i)x_T(t_i)}{x_{M1}(t_i)x_T(t_i)}, \tag{5.1a}$$

$$d_{M2}(t_i) \approx \frac{\frac{x_T(t_{i+1})-x_T(t_{i-1})}{2\tau} - rx_T(t_i)\cdot(1-\frac{x_T(t_i)}{\beta_T}) + d_{M1}x_{M1}(t_i)x_T(t_i)}{x_{M2}(t_i)x_T(t_i)}, \tag{5.1b}$$

$$a_{t1}(t_i) \approx \frac{\frac{x_{M1}(t_{i+1})-x_{M1}(t_{i-1})}{2\tau} + \delta_{M1}x_{M1}(t_i) + k_{12}x_{M1}(t_i)x_T(t_i)}{x_T(t_i)x_{M1}(t_i)\cdot(1-\frac{x_{M1}(t_i)+x_{M2}(t_i)}{\beta_M})}, \tag{5.1c}$$

$$a_{t2}(t_i) \approx \frac{\frac{x_{M2}(t_{i+1})-x_{M2}(t_{i-1})}{2\tau} + \delta_{M2}x_{M2}(t_i) - k_{12}x_{M1}(t_i)x_T(t_i)}{x_T(t_i)x_{M2}(t_i)\cdot(1-\frac{x_{M1}(t_i)+x_{M2}(t_i)}{\beta_M})}, \tag{5.1d}$$

$$k_{12}(t_i) \approx \frac{-\frac{x_{M1}(t_{i+1})-x_{M1}(t_{i-1})}{2\tau} + a_{t1}x_T(t_i)x_{M1}(t_i)\cdot(1-\frac{x_{M1}(t_i)+x_{M2}(t_i)}{\beta_M}) - \delta_{M1}x_{M1}(t_i)}{x_{M1}(t_i)x_T(t_i)}. \tag{5.1e}$$

Då alla parametrar antas vara konstanta över tid förväntas den explicit beräknade parametern anta samma värde vid varje diskret tidpunkt. På grund av övergången från en analytisk till en numerisk lösning kan det ske olika sorters beräkningsfel, vilka visas som avvikelser från det förväntade värdet.

5.1.1 Diskretiserings- och avrundningsfel

Vid numerisk representation av mycket små element ansätts ett tal med begränsad precision och decimalupplösning. Därför introduceras finita differenser vilket är derivatans diskreta motsvarighet. Sådana differenser härleds från funktionens Taylorutveckling. Första och andra ordningens diskretisering av $f'(x)$, där $f(x)$ är en godtyckligt funktion, med steglängd τ är

$$f'(x) \approx \frac{f(x+\tau)-f(x)}{\tau}, \tag{5.2}$$

respektive

$$f'(x) \approx \frac{f(x+\tau)-f(x-\tau)}{2\tau}. \tag{5.3}$$

Se appendix B.3 för härledning. I just dessa uttryck används framåtdifferens vilket är en typ av finit differens.

Avvikelser i den diskreta representationen av den kontinuerliga funktionen, som den trunkerade taylorutvecklingen, orsakar diskretiseringsfel, se avsnitt B.3. Medan begränsad talrepresentation istället orsakar avrundningsfel. Avrundningsfelet uppstår då en beräkning involverar mer värdesiffror än vad som korrekt kan representeras av datorn eller den numeriska metoden. Vidare är det möjligt att det totala avrundingsfelet kan amplifieras i varje beräkningssteg. Åtgärder för att begränsa felet inkluderar bland annat att höja precisionen och använda numeriska metoder avsedda för att minimera avrundningsfelet, till exempel Kahan summering [28]. Notera att huruvida felutbredningen fortlöper är fallspecifikt och påverkas i hög grad av den beräkningsalgoritm som används.

Tillsammans utgör summan av trunkerings- och avrundningsfelen det totala numeriska felet. Sammantaget i (5.2) och (5.3), är det för stora värden på τ som diskretiseringsfelet dominerar samtidigt som avrundningsfelet kan försummas. Medan då steglängden τ minskar krymper diskretiseringsfelet och istället framträder avrundningsfelet som dominerande. Initialt kan det framstå uppenbart att steglängden, τ, i en ideal situation bör minimeras. Men i praktiken gäller det att balansera diskretiseringsfelet mot avrundningsfelet. Som nämnts finns det flera olika metoder för att reducera dessa fel. Avrundningsfel kan i viss mån begränsas genom att förenkla ekvationerna innan de skrivs in i beräkningsprogrammet för att minimera antalet operationer. Trunkeringsfelet kan reduceras genom att använda adaptiva steglängder eller högre ordningens finita differenser.

En metod för att reducera avrundningsfel är att förenkla ekvationerna och på så vis minska antalet operationer. Förenkling av (5.1) för respektive parameter är följande

$$d_{M1}(t_i) \approx \frac{r(1-\frac{x_T}{\beta_T})}{x_{M1}(t_i)} - \frac{x_T(t_{i+1}) - x_T(t_{i-1})}{2\tau x_{M1}(t_i)x_T(t_i)} + \frac{d_{M2}x_{M2}(t_i)}{x_{M1}(t_i)}, \tag{5.4a}$$

$$d_{M2}(t_i) \approx \frac{x_T(t_{i+1}) - x_T(t_{i-1})}{2\tau x_{M2}(t_i)x_T(t_i)} - \frac{r(1-\frac{x_T(t_i)}{\beta_T})}{x_{M2(t_i)}} + \frac{d_{M1}x_{M1}(t_i)}{x_{M2}(t_i)}, \tag{5.4b}$$

$$\begin{aligned} a_{t1}(t_i) \approx & \frac{x_{M1}(t_{i+1}) - x_{M1}(t_{i-1})}{2\tau x_T(t_i)x_{M1}(t_i)(1-\frac{x_{M1}(t_i)+x_{M2}(t_i)}{\beta_M})} + \frac{\delta_{M1}}{x_T(t_i)(1-\frac{x_{M1}(t_i)+x_{M2}(t_i)}{\beta_M})} \\ & + \frac{k_{12}}{1-\frac{x_{M1}(t_i)+x_{M2}(t_i)}{\beta_M}}, \end{aligned} \tag{5.4c}$$

$$\begin{aligned} a_{t2}(t_i) \approx & \frac{x_{M2}(t_{i+1}) - x_{M2}(t_{i-1})}{2\tau x_T(t_i)x_{M2}(t_i)(1-\frac{x_{M1}(t_i)+x_{M2}(t_i)}{\beta_M})} + \frac{\delta_{M2}}{x_T(t_i)(1-\frac{x_{M1}(t_i)+x_{M2}(t_i)}{\beta_M})} \\ & - \frac{k_{12}}{1-\frac{x_{M1}(t_i)+x_{M2}(t_i)}{\beta_M}}, \end{aligned} \tag{5.4d}$$

$$k_{12}(t_i) \approx -\frac{x_{M1}(t_{i+1}) - x_{M1}(t_{i-1})}{2\tau x_{M1}(t_i)x_T(t_i)} + a_{t1}(1 - \frac{x_{M1}(t_i) + x_{M2}(t_i)}{\beta_M}) - \frac{\delta_{M1}}{x_T(t_i)}. \tag{5.4e}$$

Som tidigare nämnt kan felet reduceras genom användning av adaptiva steglängder. En sådan metod är Chebyshev noder.

5.1.2 Chebyshev noder

Genom att ansätta Chebyshev noder kan steglängden mellan interpolationspunkterna av funktionen anpassas för att minimera avvikelserna [30].

På ett godtyckligt tidsintervall $[\alpha, \beta]$ blir Chebyshev noderna, enligt [5],

$$t_i = -\frac{\beta-\alpha}{2}\cos\left(\frac{(2k-1)\pi}{2n}\right) + \frac{\alpha+\beta}{2}, \qquad k = 1, 2, ..., n, \tag{5.5}$$

där n är antalet noder.

5.2 Approximation av flera oidentifierade parametrar

I de fall flera parametrar är oidentifierade används en optimeringsalgoritm vilken är del av en metod beskriven i [6] som modifierats för projektets ändamål. För att kunna optimera parametervärdena krävs en initialestimering. Denna görs genom att analysera de givna diskreta mätvärdena för tumördensiteten och makrofagdensiteterna, $\mathbf{g} = \left(\mathrm{g}_1, \mathrm{g}_2, \mathrm{g}_3\right)$. Genom att optimera initialestimeringen ges värden på $\boldsymbol{\alpha}$ vilka gör att (3.1) bäst beskriver datapunkterna $\mathbf{g}$.

För att mäta avvikelsen mellan modellösningen, $\boldsymbol{x}$, och datapunkterna, $\mathbf{g}$, används Tikhonovs regulariseringsfunktional.

5.2.1 Tikhonovs regulariseringsfunktional

Tikhonovs regulariseringsfunktional, eller Tikhonovfunktionalen, är en funktion som kan användas vid optimering av parametrar i en annan funktion.

Definition 5.1. *Låt $\boldsymbol{x}(t) = (x_1(t), x_2(t), \ldots, x_m(t))$ vara lösningen till första ordningens differentialekvationer*

$$\begin{aligned} &\frac{\partial \boldsymbol{x}}{\partial t} = f(\boldsymbol{x}(t, \boldsymbol{\alpha}(t)),\ t \in [0, T], \\ &\boldsymbol{x}(0) = \boldsymbol{x^0}. \end{aligned}$$

Låt även $\gamma_i \in [0, 1]$ vara en regulariseringsparameter och $\mathbf{g^c}(t) = \left(\mathrm{g}_1^c(t), \mathrm{g}_2^c(t), \ldots, \mathrm{g}_m^c(t)\right)$ vara kontinuerliga approximationer av givna datapunkter, $\boldsymbol{g}$. Då skrivs Tikhonov funktionalen enligt:

$$J(\boldsymbol{\alpha}) = \frac{1}{2}\int_0^T \sum_{i=1}^{m} \left(x_i(t) - \mathrm{g}_i^c(t)\right)^2 dt + \frac{1}{2}\sum_{i=1}^{n}\int_0^T \gamma_i \left(\alpha_i(t) - \alpha_i^0\right)^2 dt \tag{5.6}$$

där $\boldsymbol{\alpha} = \big(\alpha_1(t), \alpha_2(t), \ldots, \alpha_n(t)\big)$ *är oidentifierade parametrar och* $\alpha_i^0 = \alpha_i(0)$.

Ekvation (5.6) är tagen från [6] med viss modifikation. För att bestämma $\mathbf{g^c}(t)$ kan minsta kvadratmetoden användas, se avsnitt D.

Vi formulerar minimeringsproblemet

$$\min_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}), \tag{5.7}$$

som kan användas för att identifiera vilka värden på parametrarna som bäst återskapar datapunkterna, då funktionalen beskriver felet mellan den beräknade lösningen, $\boldsymbol{x}$, och datapunkterna, $\mathbf{g}$.

För att lösa minimeringsproblemet behöver stationära punkter med avseende på $\boldsymbol{\alpha}(t)$ identifieras, sådana att

$$J'(\boldsymbol{\alpha})(\bar{\boldsymbol{\alpha}}) = 0 \tag{5.8}$$

där $J'(\boldsymbol{\alpha})(\bar{\boldsymbol{\alpha}})$ är Fréchetderivatan av $J(\boldsymbol{\alpha})$ i punkten $\bar{\boldsymbol{\alpha}}$. För definition av Fréchetderivatan, se appendix A.1.

Lokalt är Fréchetderivatan av Tikhonov funktionalen (5.6) starkt konvex [4] vilket innebär att

$$(J'(x) - J'(y), x - y) \geq \kappa \|x - y\|^2, \ \kappa = konstant > 0. \tag{5.9}$$

Följaktligen kan det förutsättas att det $\boldsymbol{\alpha}$ som löser (5.7) även löser parameteridentifikationsproblemet. För att lösa (5.7) används Lagrangemetoden.

5.2.2 Lagrangemetoden

Lagrangemetoden används för att hitta lokala extrempunkter med bestämda bivillkor. Metoden syftar till att omformulera problemet så att det kan lösas genom derivering. Omformuleringen baseras på att funktionens gradient kan skrivas om som en linjärkombination av bivillkorets gradient i en gemensam punkt [1].

Sats 5.1. *[1] Antag att funktionerna f och g har kontinuerliga första ordningens partiella derivator nära punkten P_0 på kurvan C med ekvation $g(x, y) = 0$. Antag även att, när funktionen är begränsad till punkter på C, så har funktionen $f(x, y)$ ett lokalt extremvärde i P_0. Slutligen antag att*

label=() *P_0 inte är en ändpunkt på C, samt*

lbbel=() *$\nabla g(P_0) \neq 0$.*

Då existerar det ett tal, λ_0, sådant att (x_0, y_0, λ_0) är en stationär punkt för Lagrangefunktionen

$$L(x, y, \lambda) = f(x, y) + \lambda g(x, y),$$

där λ är Lagrangemultplikatorn.

För bevis se [1].

Applicering av sats 5.1 ger att en Lagrangefunktion kan konstrueras enligt följande

$$L(\boldsymbol{v}) = J(\boldsymbol{\alpha}) + \sum_{i=1}^{m} \int_0^T \lambda_i(t) \cdot (\dot{x}_i(t) - f_i(t))dt, \tag{5.10}$$

där $\boldsymbol{\lambda} = \big(\lambda_1, \lambda_2, \ldots, \lambda_m\big)$ är Lagrangemultiplikatorn, $f(\boldsymbol{x}(\boldsymbol{\alpha}(t)) = \frac{\partial x}{\partial t}, t \in [0, T]$ och $\boldsymbol{v} = \big(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{\alpha}\big)$.

Genom att anta att varje komponent i $\boldsymbol{v} = \big(\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{\alpha}\big)$ kan varieras oberoende, sådan att

$$L'(\boldsymbol{v})(\bar{\boldsymbol{v}}) = L'_{\boldsymbol{\lambda}}(\boldsymbol{v})(\bar{\boldsymbol{\lambda}}) + L'_{\boldsymbol{x}}(\boldsymbol{v})(\bar{\boldsymbol{x}}) + L'_{\boldsymbol{\alpha}}(\boldsymbol{v})(\bar{\boldsymbol{\alpha}}) = 0 \tag{5.11}$$

uppfylls, kan beräkningarna förenklas och fortfarande ge samma resultat [7].

För Lagrangefunktionen, $L : U \subseteq \mathbb{R}^{2m+n} \to \mathbb{R}$, där m är antalet ekvationer i modellen och n är antalet oidentifierade parametrar, antas det att det att U är en öppen mängd enligt definitionen nedan

Definition 5.2. *[6] Låt $\boldsymbol{x}(t) = (x_1(t), x_2(t), \ldots, x_m(t))$ vara lösningen till första ordningens differentialekvationer*

$$\begin{aligned} &\frac{\partial \boldsymbol{x}}{\partial t} = f(\boldsymbol{x}(\boldsymbol{\alpha}(t)),\ t \in \Omega_T = [0, T] \\ &\boldsymbol{x}(0) = \boldsymbol{x^0}. \end{aligned}$$

Låt även $\boldsymbol{\lambda}(t)$ vara Lagrangemultiplikatorn $\boldsymbol{\lambda}(t) = (\lambda_1(t), \lambda_2(t), \ldots, \lambda_m(t))$ och $\boldsymbol{v} = (\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{\alpha})$. Då definieras mängden U enligt:

$$\begin{aligned} H_x^1(\Omega_T) &= \{\boldsymbol{x} \in H^1(\Omega_T) : \boldsymbol{x}(0) = \boldsymbol{x^0}\} \\ H_\lambda^1(\Omega_T) &= \{\boldsymbol{\lambda} \in H^1(\Omega_T) : \boldsymbol{\lambda}(T) = 0\} \\ U &= H_x^1(\Omega_T) \times H_\lambda^1(\Omega_T) \times C(\Omega_T) \end{aligned}$$

där alla funktioner är realvärda.

Definition för $H^1(\Omega_T)$ ges i appendix A.2.

5.2.3 Lagrangefunktionens partialderivator

Låt $\boldsymbol{v} = (\boldsymbol{\lambda}, \boldsymbol{x}, \boldsymbol{\alpha})$ och antag att parametrarna $\boldsymbol{\lambda}$, $\boldsymbol{x}$ och $\boldsymbol{\alpha}$ kan variera oberoende av varandra. Då kan vi med hjälp av följande formeln

$$L'(\boldsymbol{v})(\bar{\boldsymbol{v}}) = \sum_{i=1}^{2m+n} \bar{v}_i \frac{\partial L}{\partial v_i}(\boldsymbol{v}), \tag{5.12}$$

ta fram uttryck för de partiella derivatorna till Lagrangefunktionen som vi sätter till 0. De partiella derivatorna är samlade i följande tre satser.

Sats 5.2. *De partiella derivatorna av L med avseende på $\boldsymbol{\lambda}$ är*

$$\begin{aligned} 0 =& L'_{\lambda_1}(\boldsymbol{v})(\bar{\lambda}_1) \\ =& \int_0^T \left(\frac{dx_T}{dt} - r x_T \left(1 - \frac{x_T}{\beta_T}\right) + d_{M1} x_{M1} x_T - d_{M2} x_{M2} x_T \right) \bar{\lambda_1} dt, \\ 0 =& L'_{\lambda_2}(\boldsymbol{v})(\bar{\lambda}_2) \\ =& \int_0^T \left(\frac{dx_{M1}}{dt} - a_{t1} x_T x_{M1} \left(1 - \frac{x_{M1} + x_{M2}}{\beta_M}\right) + \delta_{M1} x_{M1} + k_{12} x_{M1} x_T \right) \bar{\lambda_2} dt, \\ 0 =& L'_{\lambda_3}(\boldsymbol{v})(\bar{\lambda}_3) \\ =& \int_0^T \left(\frac{dx_{M2}}{dt} - a_{t2} x_T x_{M2} \left(1 - \frac{x_{M1} + x_{M2}}{\beta_M}\right) + \delta_{M2} x_{M2} - k_{12} x_{M1} x_T \right) \bar{\lambda_3} dt, \end{aligned}$$

$\forall \bar{\boldsymbol{\lambda}} = \left(\bar{\lambda}_1, \bar{\lambda}_2, \bar{\lambda}_3\right) \in H_\lambda^1(\Omega_T)$.

Sats 5.3. *De partiella derivatorna av L med avseende på $\boldsymbol{x}$ är*

$$
\begin{aligned}
0 =& L'_{x_T}(\boldsymbol{v})(\bar{x}_T) \\
=& \int_0^T (x_T - g_1^c)\bar{x}_T dt + \int_0^T (-\frac{d\lambda_1}{dt} - \lambda_1 r(1 - \frac{2x_T}{\beta_T}) + \lambda_1 d_{M1} x_{M1} - \lambda_1 d_{M2} x_{M2} \\
& - \lambda_2 a_{t1} x_{M1}(1 - \frac{x_{M1} + x_{M2}}{\beta_M}) + \lambda_2 k_{12} x_{M1} - \lambda_3 a_{t2} x_{M2}(1 - \frac{x_{M1} + x_{M2}}{\beta_M}) \\
& - \lambda_3 k_{12} x_{M1})\bar{x}_T dt, \\
0 =& L'_{x_{M1}}(\boldsymbol{v})(\bar{x}_{M1}) \\
=& \int_0^T (x_{M1} - g_2^c)\bar{x}_{M1} dt + \int_0^T (\lambda_1 d_{M1} x_T - \frac{d\lambda_2}{dt} - \lambda_2 a_{t1} x_T(1 - \frac{2x_{M1} + x_{M2}}{\beta_M}) + \lambda_2 \delta_{M1} \\
& + \lambda_2 k_{12} x_T + \frac{\lambda_3 a_{t2} x_T x_{M2}}{\beta_M} - \lambda_3 k_{12} x_T)\bar{x}_{M1} dt, \\
0 =& L'_{x_{M2}}(\boldsymbol{v})(\bar{x}_{M2}) \\
=& \int_0^T (x_{M2} - g_3^c)\bar{x}_{M2} dt + \int_0^T (-\lambda_1 d_{M2} x_T + \frac{\lambda_2 a_{t1} x_T x_{M1}}{\beta_M} - \frac{d\lambda_3}{dt} \\
& - \lambda_3 a_{t2} x_T(1 - \frac{x_{M1} + 2x_{M2}}{\beta_M}) + \lambda_3 \delta_{M2})\bar{x}_{M2} dt,
\end{aligned}
$$

$\forall \bar{\boldsymbol{x}} = (\bar{x}_T, \bar{x}_{M1}, \bar{x}_{M2}) \in H_x^1(\Omega_T)$.

Sats 5.4. *De partiella derivatorna av L med avseende på $\boldsymbol{\alpha}$ är*

$$
\begin{aligned}
0 =& L'_{d_{M1}}(\boldsymbol{v})(\bar{d}_{m1}) \\
=& \gamma_1 \int_0^T (d_{M1} - d_{M1}^0)\bar{d}_{m1} dt + \int_0^T \lambda_1 x_{M1} x_T \bar{d}_{m1} dt \\
0 =& L'_{d_{M2}}(\boldsymbol{v})(\bar{d}_{m2}) \\
=& \gamma_2 \int_0^T (d_{M2} - d_{M2}^0)\bar{d}_{m2} dt - \int_0^T \lambda_1 x_{M2} x_T \bar{d}_{m2} dt \\
0 =& L'_{a_{t1}}(\boldsymbol{v})(\bar{a}_{t1}) \\
=& \gamma_3 \int_0^T (a_{t1} - a_{t1}^0)\bar{a}_{t1} dt - \int_0^T \lambda_2 x_{M1} x_T (1 - \frac{x_{M1} + x_{M2}}{\beta_M})\bar{a}_{t1} dt \\
0 =& L'_{a_{t2}}(\boldsymbol{v})(\bar{a}_{t2}) \\
=& \gamma_4 \int_0^T (a_{t2} - a_{t2}^0)\bar{a}_{t2} dt - \int_0^T \lambda_3 x_{M2} x_T (1 - \frac{x_{M1} + x_{M2}}{\beta_M})\bar{a}_{t2} dt \\
0 =& L'_{k_{12}}(\boldsymbol{v})(\bar{k}_{12}) \\
=& \gamma_5 \int_0^T (k_{12} - k_{12}^0)\bar{k}_{12} dt + \int_0^T x_{M1} x_T (\lambda_2 - \lambda_3)\bar{k}_{12} dt
\end{aligned}
$$

$\forall \bar{\boldsymbol{\alpha}} = (\bar{d}_{m1}, \bar{d}_{m2}, \bar{a}_{t1}, \bar{a}_{t2}, \bar{k}_{12}) \in C(\Omega_T)$.

För bevis av sats 5.2, 5.3 och 5.4 se appendix B.2. Vidare kommer dessa ekvationer att användas i en lösningsalgoritm, specifikt konjugerade gradientmetoden, för iterativ uppdatering av parametrarna som avses i de partiella derivatorna, $\boldsymbol{\alpha} = (d_{M1}, d_{M2}, a_{t1}, a_{t2}, k_{12})$.

5.2.4 Bakåtproblemet

För att beräkna de partiella derivatorna av L för olika värden av $\boldsymbol{v}$ ser vi enligt sats 5.2, 5.3 och 5.4 att $\boldsymbol{\lambda}$ behöver bestämmas. För att göra detta utnyttjas det att derivatorna i sats 5.3 är de svaga formuleringarna av följande system av differentialekvationer [6]

$$\begin{aligned}\frac{d\lambda_1}{dt} &= -\lambda_1 r(1-\frac{2x_T}{\beta_T}) + \lambda_1 d_{M1}x_{M1} - \lambda_1 d_{M2}x_{M2} - \lambda_2 a_{t1}x_{M1}(1-\frac{x_{M1}+x_{M2}}{\beta_M}) \\ &\quad + \lambda_2 k_{12}x_{M1} - \lambda_3 a_{t2}x_{M2}(1-\frac{x_{M1}+x_{M2}}{\beta_M}) - \lambda_3 k_{12}x_{M1} + x_T - g_1^c, \end{aligned} \tag{5.13a}$$

$$\begin{aligned}\frac{d\lambda_2}{dt} &= -\lambda_2 a_{t1}x_T(1-\frac{2x_{M1}+x_{M2}}{\beta_M}) + \lambda_2\delta_{M1} + \frac{\lambda_3 a_{t2}x_T x_{M2}}{\beta_M} - \lambda_3 k_{12}x_T + \lambda_1 d_{M1}x_T \\ &\quad + \lambda_2 k_{12}x_T + x_{M1} - g_2^c, \end{aligned} \tag{5.13b}$$

$$\frac{d\lambda_3}{dt} = \lambda_3\delta_{M2} - \lambda_3 a_{t2}x_T(1-\frac{x_{M1}+2x_{M2}}{\beta_M}) - \lambda_1 d_{M2}x_T + \frac{\lambda_2 a_{t1}x_T x_{M1}}{\beta_M} + x_{M2} - g_3^c, \tag{5.13c}$$

vilka kan användas för att beräkna $\boldsymbol{\lambda}$.

5.2.5 Konjugerade gradientmetoden

För att slutligen optimera $\boldsymbol{\alpha}$ används den konjugerade gradientmetoden. Låt gradienten $\mathbf{K}^{(s)}(t_i)$ vid tidpunkt t_i och optimeringsiteration s definieras enligt följande

$$\mathbf{K}^{(s)}(t_i) = \left(L'^{(s)}_{\boldsymbol{\alpha}_1}(t_i),\ L'^{(s)}_{\boldsymbol{\alpha}_2}(t_i),\ \ldots,\ L'^{(s)}_{\boldsymbol{\alpha}_n}(t_i)\right). \tag{5.14}$$

Genom att för $\boldsymbol{\alpha}^{(s)} = \left(\alpha_1^{(s)},\ \alpha_2^{(s)},\ \ldots,\ \alpha_n^{(s)}\right)$ lösa $\boldsymbol{x}^{(s)}$ och $\boldsymbol{\lambda}^{(s)}$ kan nästkommande iteration av $\boldsymbol{\alpha}$ bestämmas enligt

$$\boldsymbol{\alpha}^{(s+1)}(t_i) = \boldsymbol{\alpha}^{(s)}(t_i) + \boldsymbol{\beta}^{(s)} \cdot \boldsymbol{\delta}^{(s)}(t_i). \tag{5.15}$$

Vektorn $\boldsymbol{\beta}^{(s)}$ är steglängden i gradientmetoden och bestäms enligt

$$\boldsymbol{\beta}^{(s)} = -\frac{(\mathbf{K}^{(s)}, \boldsymbol{\delta}^{(s)})}{\boldsymbol{\gamma}^{(s)} \cdot (\boldsymbol{\delta}^{(s)}, \boldsymbol{\delta}^{(s)})}, \tag{5.16}$$

där $(\cdot,\cdot)$ är inre produkten i L^2 rummet. Vektorn $\boldsymbol{\gamma}^{(s)} = \left(\gamma_1^{(s)},\ \gamma_2^{(s)},\ \ldots,\ \gamma_n^{(s)}\right)$ är regulariseringsparametrarna och bestäms enligt följande iterativa regel

$$\boldsymbol{\gamma}^{(s)} = \frac{\boldsymbol{\gamma}^{(0)}}{(s+1)^\eta}, \tag{5.17}$$

med $\boldsymbol{\gamma}^{(0)} \ll \boldsymbol{\alpha}^{(0)}$ och $\eta \in (0,1)$. Observera att dessa är samma regulariseringsparametrar som används i (5.6). Slutligen ges $\boldsymbol{\delta}^{(s)}(t_i)$ av

$$\boldsymbol{\delta}^{(s)}(t_i) = -\mathbf{K}^{(s)}(t_i) + \sigma^{(s)} \cdot \boldsymbol{\delta}^{(s-1)}(t_i), \tag{5.18}$$

där $\sigma^{(s)}$ bestäms enligt

$$\sigma^{(s)} = \frac{\|\mathbf{K}^{(s)}(t_i)\|^2}{\|\mathbf{K}^{(s-1)}(t_i)\|^2}. \tag{5.19}$$

För första iterationen är $\boldsymbol{\delta}^{(0)}(t_i) = -\mathbf{K}^{(0)}(t_i)$. Denna typ av iteration upprepas tills något av följande är uppfyllt:

- $\|\mathbf{K}^{(s)}(t_i)\| < \theta$ där $\theta \in (0,1)$ är en vald tolerans,
- $\|\mathbf{K}^{(s)}(t_i)\|$ börjar växa markant,
- $\|\boldsymbol{\alpha}^{(s)}\|$ närmar sig något $\|\boldsymbol{\alpha}_{opt}\|$.

Med en tillräckligt bra initialestimering kommer $\boldsymbol{\alpha}^{(s)}$ att närma sig lösningen till minimeringsproblemet (5.7).

6 Resultat från explicit beräkning av en parameter

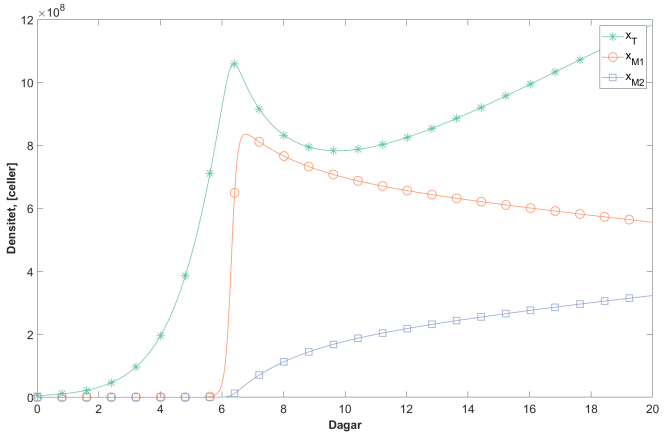
Det finns flera olika metoder för att lösa differentialekvationer. I projektet testades två av MATLABs egna lösare, ode45 och ode23s. Lösaren ode45 fungerar för icke-styva problem medan ode23s är anpassad för att lösa styva ekvationssystem. Ett system anses styvt om det innehåller flera olika tidsskalor, det vill säga, systemet innehåller snabbt och långsamt varierande parametrar samtidigt. Detta innebär att lösningen kan vara mycket känslig för små förändringar i ingångsparametrarna. Även Newtons metod har prövats för de relevanta numeriska beräkningarna. För beskrivning av denna metod samt diskussion kring resultaten vid användning av metoden, se appendix C. Det gjordes en analys av de olika lösarna samt Newtons metod och det konstaterades att ode23s presterade bäst för alla parametrar. För jämförelse av olika lösare, se appendix E.

I avsnitt 5.1 beskrevs det hur explicit beräkning var möjligt i fallet då alla parametrar utom en är känd. Valda parametervärden samt startvärden för densiteterna som användes vid explicita beräkningar visas i tabell 3. Alla explicita beräkningar utfördes numeriskt över tidsintervallet 0 till 20 dagar.

Tabell 3: Startvärden för parametrar samt densiteter vid explicita beräkningar.

Parameter/Densitetet	d_{M1}	d_{M2}	a_{t1}	a_{t2}	k_{12}	x_T	x_{M1}	x_{M2}
Initialvärde	10^{-9}	10^{-10}	10^{-8}	10^{-10}	10^{-10}	$5 \cdot 10^6$	10^3	10^3

Figur 6.1 visar densiteterna vilka beräknats med initialvärden enligt tabell 3.



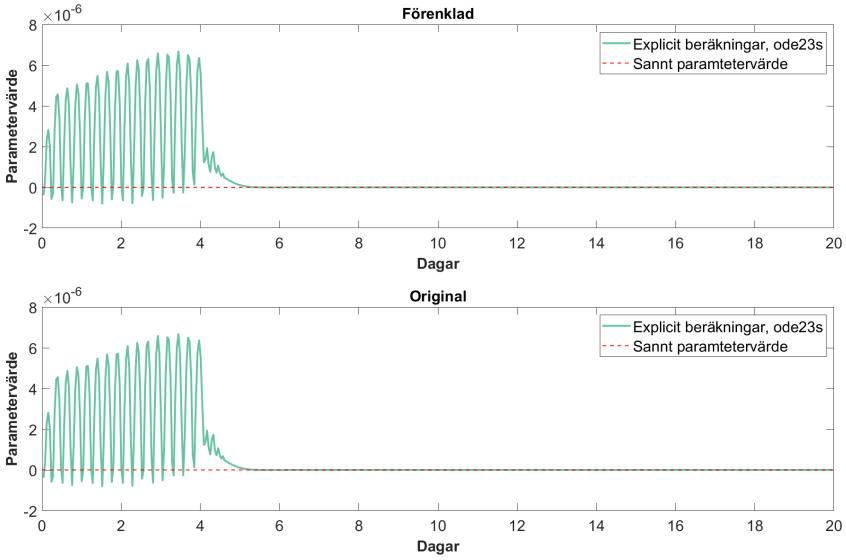
Figur 6.1: Graf av beräknade densiteterna $\boldsymbol{x}$ som används i de explicita beräkningarna.

Då det antas att parametrarna är konstanta över tid förväntas det att beräkningarna enligt (5.1) resulterar i en horisontell linje vid parameterns sanna värde. För alla fem parametrar visar graferna istället en relativt stor varians kring det sanna parametervärdet i början av tidsintervallet innan det stabiliseras, för exempel se figur 6.2 och 6.3. I texten visas endast resultat för ett fåtal parametrar, för resterande parametrar se appendix E.

För att reducera felen tillämpades flera olika metoder: förenkling av ekvationerna, ansättning av Chebyshev noder samt testning av olika startvärden på densiteterna, $\boldsymbol{x}$.

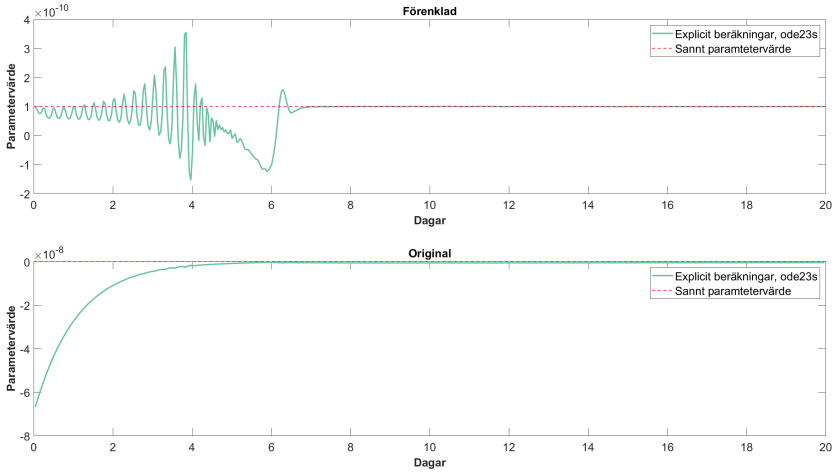
6.1 Jämförelse mellan original ekvationer och förenklade ekvationer

Inledningsvis utfördes enbart förenklingen enligt (5.4) vars resultat visas, i jämförelse med resultatet för originalekvationen i figur 6.2 och 6.3, för d_{M2} respektive k_{12}.



Figur 6.2: Graf för explicita beräkningar av d_{M2}. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $9.93 \cdot 10^{-11}$ för både den förenklade modellen och original modellen.

Graferna för parameter d_{M2} uppvisar fluktuationer i början av tidsintervallet för båda varianterna av ekvationen. Efter dag fyra börjar båda graferna stabiliseras ut vid det sanna parametervärdet, 10^{-10}.



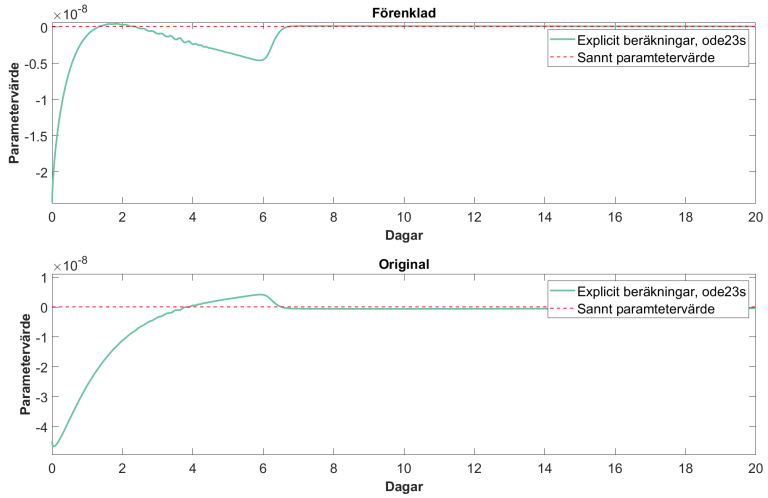
Figur 6.3: Graf för explicita beräkningar av k_{12}. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $9.99 \cdot 10^{-11}$ för den förenklade modellen och $-4.67 \cdot 10^{-10}$ för original modellen.

Till skillnad från parametern d_{M2} observeras en större skillnad mellan båda varianterna av ekvationen för parameter k_{12}, se figur 6.3. Den förenklade ekvationen genererar en graf med betydande variation i början av intervallet innan den stagnerar, vid det sanna värdet 10^{-10}. Grafen till originalekvationen är mindre dynamisk och börjar vid dag sex stagnera kring $-4 \cdot 10^{-10}$, vilket är utanför parameterintervallet. Observera att figuren kan vara missvisande då skalorna på axlarna för de två graferna skiljer sig markant.

För parameter d_{M1}, d_{M2} samt a_{t1} är graferna för originalekvationen samt den förenklade nästintill identiska, medan representationen av parameter a_{t2} och k_{12} blev något försämrad respektive förbättrad.

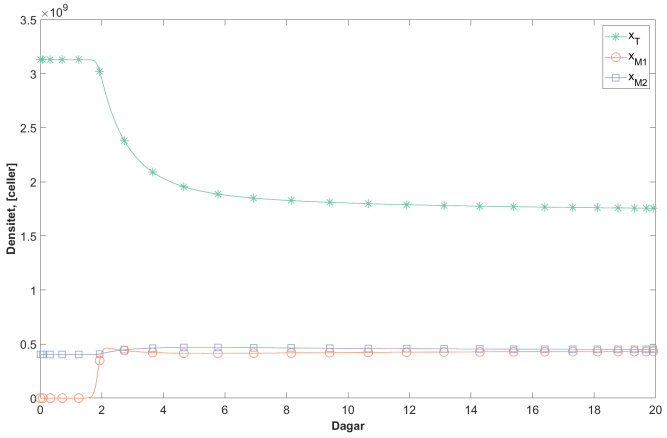
6.2 Explicit beräkning med Chebyshev noder

För att reducera felen ansattes 500 Chebyshev noder, vilket illustreras i figur 6.4 och figur 6.6.



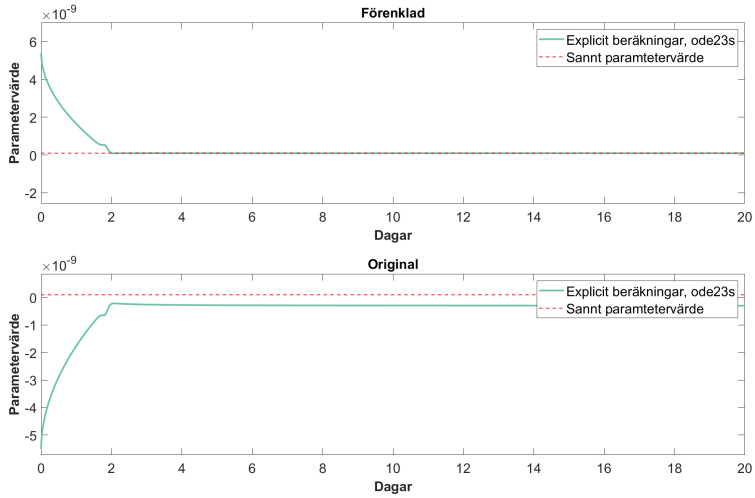
Figur 6.4: Graf för explicita beräkningar av k_{12} utförda med Chebyshev noder med startvärden enligt tabell 3. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $1.03 \cdot 10^{-10}$ för den förenklade modellen och $-4.49 \cdot 10^{-10}$ för original modellen.

Ansättning av Chebyshev noder, i figur 6.4, resulterade i mindre fluktuation för de två graferna vilka är relativt lika. Genom att utföra samma explicita beräkningar med initialvärden nära ett steady state, enligt $x_T = 3.130 \cdot 10^9$, $x_{M1} = 1.000 \cdot 10^{-5}$ och $x_{M2} = 4.025 \cdot 10^8$, ges följande graf för densiteterna.



Figur 6.5: Graf av beräknade densiteterna $\boldsymbol{x}$ med startvärden nära steady state.

Notera att det steady state som valdes tros vara asymptotiskt stabilt. Detta bör dock bevisas analytiskt.



Figur 6.6: Graf för explicita beräkningar av k_{12} utförda med Chebyshev noder nära steady state. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $9.98 \cdot 10^{-11}$ för den förenklade modellen och $-2.95 \cdot 10^{-10}$ för original modellen.

I figur 6.6 visar graferna en jämn kurva som närmar sig det sanna parametervärdet vid dag 2. Resultaten visar att avvikelserna de första två dagarna är lägre med initialvärden nära steady state. I jämförelse mellan båda varianterna av ekvationen är de fortfarande relativt identiska för d_{M1}, d_{M2} och a_{t1} medan det finns en skillnad mellan dessa för a_{t2} och k_{12}.

7 Diskussion

Att med matematiska modeller beskriva tumörers tillväxt och med specifika mätvärden förutspå hur de kommer att växa är av stort intresse utifrån ett samhällsperspektiv. Trots att modeller sen tidigare tagits fram för att beskriva tillväxtförloppet hos cancerceller finns det ingen implementerad metod för det syftet.

I detta projekt utvecklades metoder för att bestämma de parametrar vilka representerar variationerna i tillväxtförloppen hos olika individer. Först analyserades den valda modellen för att visa vilken påverkan de oidentifierade parametrarna hade, både på kort och lång sikt. Detta utfördes med hjälp av en steady state-analys och en känslighetsanalys. Efter modellanalysen introducerades en metod för att explicit beräkna en oidentifierad parameter och det redogjordes även för en metod vilken förväntas kunna identifiera flera parametrar samtidigt. För att undersöka lämpligheten att använda den valda modellen för att identifiera flera parametrar lades extra fokus på numeriska studier för explicit beräkning av en parameter.

Resultaten av de explicita beräkningarna visade i de tidigare delarna av tidsintervallet stora avvikelser från parametrarnas sanna värden. Dessa avvikelser minskar dock mot mitten och slutet av intervallet.

7.1 Tolkning av resultat

Analys av resultaten från den explicita beräkningen, vilka alla hade avvikelser i början av tidsintervallet, ledde till hypotesen att dessa avvikelser beror på avrundningsfel och diskretiseringsfel. I jämförelsen mellan de ursprungligen introducerade ekvationerna samt dess förenklade versioner visas det att skillnaden för parametrarna d_{M1}, d_{M2} och a_{t1} knappt var märkbar. För parametrarna a_{t2} och k_{12} var däremot skillnaden betydligt mer signifikant. Samma tendenser finns även i graferna där Chebyshev noder implementerats, både med initialvärden enligt tabell 3 för densiteterna och startvärden nära ett steady state.

Införandet av Chebyshev noder resulterade inte i någon signifikant förbättring i de explicita

beräkningarna. Däremot när densiteternas initialvärden ansattes nära ett steady state resulterade detta i förbättrade parameteruppskattningar i början av tidsintervallet. Dessa förbättringar skulle möjligtvis kunna vara kopplade till att densiteterna förändras långsammare i närheten av valt steady state. En annan möjlig förklaring är att storleken på de alternativa startvärdena på x kan minska felen. Det behöver dock undersökas vidare vad de förbättrade resultaten beror på.

Det finns flera metoder som kan utvärderas för att förbättra resultaten i de explicita beräkningarna. En av dessa innefattar elimination av de termer i (5.1) och (5.4) som orsakar stora avvikelser på mindre delintervall.

7.2 Utvärdering av projektets begränsningar

I det tidiga skedet av intervallet antar ibland x_T, x_{M1} och x_{M2} väldigt små värden. Detta visades problematiskt eftersom parametrarna har en storleksordning kring 10^{-9} vilket ger upphov till fel eller avvikelser från den förväntade lösningen, se avsnitt 5.1.1. I avsnitt 6 uppenbaras återigen problematiken med små parametervärden, då den explicita parameterrekonstruktionen fluktuerar initialt, se exempelvis figur 6.2 och 6.3.

Med hänsyn till de små storleksordningarna och att MATLAB som standard tillämpar en precision på 16 decimaler [25] går det inte att utesluta att precisionsbegräsningen lätt skulle kunna överskridas. Detta hade bidragit till ett större avrundningsfel och således även det totala numeriska felet. För att hantera denna fråga kan det vara lämpligt att överväga algoritmer och programvara med förbättrad precision.

En alternativ metod för att hantera avrundningsproblemet innefattar användning av olika skalningstekniker, vilka syftar till att omvandla skalorna på parameterintervallen till mer hanterbara storlekar. Val av skalningsmetod beror i hög grad på modellformuleringen och omskalning av systemet kräver därav djupare förståelse av modellen. Detta för att undvika förlust av fysikalisk och biologisk koppling.

Det är tydligt att det finns utmaningar kopplade till den nuvarande metoden, i synnerhet när det kommer till skalning och beräkningsprecision. Därför är det viktigt att betona att förbättringsmöjligheter för metoder som presenteras kräver bearbetning och vidare forskning. Om man tillåter sig att bortse från dessa utmaningar och istället fokuserar på att utveckla verktyg för fortsatt arbete, så finns det betydande möjligheter till framsteg.

7.3 Framtida forskning och hypoteser

Med explicit beräkning av en parameter implementerad är ett möjligt nästa steg att identifiera flera parametrar samtidigt, förslagsvis med metoden beskriven i avsnitt 5.2. På grund av avvikelserna i de explicita beräkningarna är en rimlig hypotes att liknande fel även kommer att uppstå i optimeringsalgoritmen. Ett möjligt tillvägagångssätt för att kringgå problemet är att implementera algoritmen på ett tidsintervall där de explicita beräkningarna resulterar i tillräckligt små avvikelser. Detta skulle innebära en del förändringar i bland annat Tikhonovfunktionalen och Lagrangemetoden då de behöver omformuleras för att passa in på det nya tidsintervallet. Vidare är det relevant att undersöka om parameterrekonstruktion är möjlig även då brus introducerats i datan, för att simulera verkliga mätvärden.

Förhoppningen är att modellen, (3.1), i framtiden ska kunna ge en bättre förståelse av tumörtillväxt vilket i längden kan gynna cancerbehandling. För att kunna använda dessa metoder i syfte att beskriva tumörtillväxten i människor är det nödvändigt att vidare studera och utveckla modellen.

Referenser

[1] Adams RA, Essex C. Calculus : a complete course. 9 uppl. North York: Pearson Canada Inc; 2018. s. 85.

[2] Alberts B, Johnson A, Lewis J, Morgan D, Raff M, Roberts K, et al. Molecular Biology of the Cell. 6 uppl. New York: Garland Science; 2015.

[3] Asadzadeh M. An Introduction to the Finite Element Method for Differential Equations. Hoboken: John Wiley & Sons Inc; 2020.

[4] Beilina L, Klibanov MV, Kokurin MY. Adaptivity with relaxation for ill-posed problems and global convergence for a coefficient inverse problem. J Math Sci. 2010; 167: 279–325. doi: 10.1007/s10958-010-9921-1

[5] Beilina L. Numerisk Analys, MMG410, Lecture 14 [internet]. Göteborg:;2019 [citerad 6 maj 2023]. Hämtad från: `http://www.math.chalmers.se/Math/Grundutb/GU/MMG410/V19/Lectures/`

[6] Beilina L, Eftimie R. Parameter identification for a mathematical model describing tumor-macrophages interactions. Under förberedelse.

[7] Beilina L, Klibanov MV. Approximate Global Convergence and Adaptivity for Coefficient Inverse Problems [internet]. Berlin: Springer Science+Business Media; 2012. [citerad 2 maj 2023]. Hämtad från: `https://link.springer.com/book/10.1007/978-1-4419-7805-9`

[8] Berger A. Science commentary: Th1 and Th2 responses: what are they? BMJ. 2000 Aug 12; 321(7258): s.424. doi: 10.1136/bmj.321.7258.424

[9] den Breems NY, Eftimie R. The re-polarisation of M2 and M1 macrophages and its role on cancer outcomes. J Theor Biol. 2016 Feb 7; 390: 23–39. doi: 10.1016/j.jtbi.2015.10.034

[10] Cancerfonden. Palliativ vård i livets slutskede [internet]. [okänt år][citerad 28 mars 2023]. Hämtad från: `https://www.cancerfonden.se/om-cancer/obotlig-cancer/vard-livets-slut`

[11] Cancerfonden. Prognos vid cancer [internet]. [okänt år][citerad 28 mars 2023]. Hämtad från: `https://www.cancerfonden.se/om-cancer/statistik/prognos-om-overlevnad-vid-cancer`

[12] Chakraborty S, Rahman T. The difficulties in cancer treatment. Ecancermedicalscience. 2012 Nov 14; 6:ed16. doi: 10.3332/ecancer.2012.ed16. PMID: 24883085; PMCID: PMC4024849.

[13] Chummun S, McLean NR. The management of malignant skin cancers. Surgery (Oxf). 2017 Sep; 35(9): 519–24. doi: 10.1016/j.mpsur.2017.06.013

[14] Danciu C, Falamas A, Dehelean C, Soica C, Radeke H, Barbu-Tudoran L, et al. A characterization of four B16 murine melanoma cell sublines molecular fingerprint and proliferation behavior. Cancer Cell Int. 2013; 13(1): artikel nr. 75. doi: 10.1186/1475-2867-13-75

[15] Dwivedi A, Tripathi A, Ray RS, Singh AK. Skin Cancer: Pathogenesis and Diagnosis. Springer Singapore; 2021. doi: 10.1007/978-981-16-0364-8

[16] Eftimie R, Gillard JJ, Cantrell DA. Mathematical Models for Immunology: Current State of the Art and Future Research Directions. Bull Math Biol. 2016; 78: 2091–134. doi: 10.1007/s11538-016-0214-9

[17] Fidler IJ, Jessup JM, Kleinerman ES, Fogler WE, Mazumder A. Circumvention of neoplastic heterogeneity by systemically activated macrophages. I: Mastromarino, AJ. Biology and Treatment of Colorectal Cancer Metastasis. Developments in Oncology, vol 42. Boston: Springer; 1986. doi: 10.1007/978-1-4613-2301-3_25

[18] Friberg S, Mattson S. On the growth rates of human malignant tumors: implications for

medical decision making. J Surg Oncol. 1997 Aug;65(4):284-97. doi: 10.1002/(sici)1096-9098(199708)65:4<284::aid-jso11>3.0.co;2-2. PMID: 9274795.

[19] Golub GH, Van Loan CF. Matrix computations. 3 uppl. Baltimore: The Johns Hopkins University Press; 1996.

[20] Hussein MR. Tumour-associated macrophages and melanoma tumourigenesis: integrating the complexity. Int J Exp Pathol. 2006 Jun;87(3):163-76. doi: 10.1111/j.1365-2613.2006.00478.x. PMID: 16709225; PMCID: PMC2517364.

[21] Italiani P, Boraschi D. From monocytes to M1/M2 macrophages: Phenotypical vs. functional differentiation. Front Immunol. 2014; 5: 514. doi: 10.3389/fimmu.2014.00514

[22] Logemann H, Ryan EP. Ordinary Differential Equations, Analysis, Qualitative Theory and Control. London: Springer London; 2014. doi: 10.1007/978-1-4471-6398-5

[23] Mantovani A, Locati M. Tumor-Associated Macrophages as a Paradigm of Macrophage Plasticity, Diversity, and Polarization. Arterioscler Thromb Vasc Biol. 2013;33(7):1478-83. doi: 10.1161/ATVBAHA.113.300168

[24] Mantovani A, Sica A, Locati M. Macrophage Polarization Comes of Age. Immunity. 2005; 23(4): 344-6. doi: 10.1016/J.IMMUNI.2005.10.001

[25] MathWorks. digits [internet]. 2023 [citerad 4 maj 2023]. Hämtad från: `https://se.mathworks.com/help/symbolic/digits.html`

[26] Mills CD. M1 and M2 Macrophages: Oracles of Health and Disease. Crit Rev Immunol. 2012;32(6):463-88. doi: 10.1615/critrevimmunol.v32.i6.10. PMID: 23428224.

[27] Parkin J, Cohen B. An overview of the immune system. Lancet. 2001; 357(9279): 1777–89. doi: 10.1016/S0140-6736(00)04904-7

[28] Robey RW, Robey JM, Aulwes R. In search of numerical consistency in parallel programming. Parallel Comput. 2011 Apr-May; 37(4-5): 217-29. doi: 10.1016/J.PARCO.2011.02.009

[29] Schumacher TN, Schreiber RD. Neoantigens in cancer immunotherapy. Science. 2015; 348(6230): 69-74. doi: 10.1126/science.aaa4971

[30] Stewart GW. Afternotes on Numerical Analysis. Philadelphia: Society for Industrial and Applied Mathematics; 1996.

[31] Tang C. In vitro vs. In vivo: Is One Better? [internet]. Toronto: University Health Network; [okänt år] [citerad 18 april 2023]. Hämtad från: `https://www.uhnresearch.ca/news/vitro-vs-vivo-one-better`

A Appendix 1 – Definitioner

A.1 Fréchetderivatan

Följande definition av Frechétderivatan är tagen från appendix A.3 i [22].

Definition A.1 (Fréchetderivatan)**.** *[22]*

Låt $X \subset \mathbb{R}^N$ vara en öppen icke-tom mängd. En funktion $f : X \to \mathbb{R}^M$ är differentierbar i punkten $x \in X$ om det finns en reell $M \times N$ matris, som vi betecknar med $(Df(x))(x) \in \mathbb{R}^{M\times N}$, sådan att

$$\lim_{z\to 0} \frac{||f(x+z) - f(x) - ((Df)(x))z||}{||z||} = 0.$$

A.2 Definition av rum

A.2.1 Sobolev-rum

Definition A.2. *[3] Låt Ω vara en öppen delmängd av $\mathbb{R}^n$. För icke-negativa heltal k och $p \in [1, \infty]$, definierar vi Sobolev rummet av ordning k enligt*

$$W_p^k(\Omega) \coloneqq \{u \in L_p(\Omega) : D^\alpha \in L_p(\Omega), |\alpha| \leq k\}. \tag{A.1}$$

De korresponderande Sobolev-normerna är

$$\|u\|_{W_p^k(\Omega)} \coloneqq \left(\sum_{|\alpha|\leq k} \|D^\alpha u\|^p_{L_p(\omega)} \right)^{1/p}, \ 1 \leq p < \infty, \tag{A.2}$$

och

$$\|u\|_{W_\infty^k(\Omega)} \coloneqq \sum_{|\alpha|\leq k} \|D^\alpha u\|_{L_\infty(\Omega)}. \tag{A.3}$$

Vi definierar också seminormerna enligt

$$|u|_{W_p^k(\Omega)} \coloneqq \left(\sum_{|\alpha|=k} \|D^\alpha u\|^p_{L_p(\omega)} \right)^{1/p}, \ 1 \leq p < \infty. \tag{A.4}$$

Därmed, för $1 \leq p < \infty$, får vi skriva

$$\|u\|_{W_\infty^k(\Omega)} \coloneqq \left(\sum_{j=0}^{k} |u|^p_{W_p^j(\Omega)} \right). \tag{A.5}$$

Utöver det, när $p = \infty$, får vi

$$|u|_{W_\infty^k(\Omega)} \coloneqq \sum_{|\alpha|\leq k} \|D^\alpha u\|_{L_\infty(\Omega)}, \tag{A.6}$$

vilket ger

$$\|u\|_{W_\infty^k(\Omega)} \coloneqq \sum_{j=0}^{k} |u|_{W_\infty^j(\Omega)} \tag{A.7}$$

Anmärkning A.1. *För $k = 0$ är $|\cdot|_{W_p^k(\Omega)}$ den vanliga L_p-normen. Benämningen seminorm används endast när $k \geq 1$*

A.2.2 Hilbert-rum

Specialfallen av W_p^k där $p = 2$ och $k = 1, 2$ kallas Hilbertrum och skrivs $H^k(\Omega),\ \Omega \subset \mathbb{R}^n$ [3]. För $k = 1$ fås

$$H^1(\Omega) \coloneqq \left\{ u \in L_2(\Omega) \middle| \frac{\partial u}{\partial x_j} \in L_2(\Omega), j = 1, \ldots, n \right\}, \tag{A.8a}$$

$$\|u\|_{H^1(\Omega)} \coloneqq \left(\|u\|^2_{L_2(\Omega)} + \sum_{j=1}^{n} \left\| \frac{\partial u}{\partial x_j} \right\|^2_{L_2(\Omega)} \right)^{1/2}, \tag{A.8b}$$

$$|u|_{H^1(\Omega)} \coloneqq \left(\sum_{j=1}^{n} \left\| \frac{\partial u}{\partial x_j} \right\|^2_{L_2(\Omega)} \right)^{1/2}, \tag{A.8c}$$

och för $k = 2$

$$H^2(\Omega) \coloneqq \left\{ u \in L_2(\Omega) \middle| \frac{\partial u}{\partial x_j} \in L_2(\Omega), j = 1, \ldots, n \text{ och } \frac{\partial^2 u}{\partial x_i x_j} \in L_2(\Omega),\ i, j = 1, \ldots, n \right\}, \tag{A.9a}$$

$$\|u\|_{H^2(\Omega)} \coloneqq \left(\|u\|^2_{L_2(\Omega)} + \sum_{j=1}^{n} \left\| \frac{\partial u}{\partial x_j} \right\|^2_{L_2(\Omega)} + \sum_{i,j=1}^{n} \left\| \frac{\partial^2 u}{\partial x_i x_j} \right\|^2_{L_2(\Omega)} \right)^{1/2}, \tag{A.9b}$$

$$|u|_{H^2(\Omega)} \coloneqq \left(\sum_{j=1}^{n} \left\| \frac{\partial u}{\partial x_j} \right\|^2_{L_2(\Omega)} \right)^{1/2}. \tag{A.9c}$$

B Appendix 2 – Bevis

B.1 Analys av steady states

Bevis av sats 4.1. Antag att $\boldsymbol{x}(t) \geq \mathbf{0}$, $\boldsymbol{\alpha} > \mathbf{0}$ och att $r, \beta_T, \beta_M, \delta_{M1}, \delta_{M2} > 0$. Låt $f(\boldsymbol{x}) = \frac{dx}{dt}$, det vill säga

$$f_1(\boldsymbol{x}) = rx_T\left(1 - \frac{x_T}{\beta_T}\right) - d_{M1}x_{M1}x_T + d_{M2}x_{M2}x_T, \tag{B.1a}$$

$$f_2(\boldsymbol{x}) = a_{t1}x_Tx_{M1}\left(1 - \frac{x_{M1} + x_{M2}}{\beta_M}\right) - \delta_{M1}x_{M1} - k_{12}x_{M1}x_T, \tag{B.1b}$$

$$f_3(\boldsymbol{x}) = a_{t2}x_Tx_{M2}\left(1 - \frac{x_{M1} + x_{M2}}{\beta_M}\right) - \delta_{M2}x_{M2} + k_{12}x_{M1}x_T. \tag{B.1c}$$

Enligt definition är punkten $\boldsymbol{x}^*$ ett steady state om och endast om $f(\boldsymbol{x}^*) = 0$.

Vi börjar med att hitta alla steady state under antagandet att $x_T^* = 0$ vilket ger att $f(\boldsymbol{x}) = 0$ förenklas till

$$\begin{cases} f_1(\boldsymbol{x}^*) = 0, & \text{(B.2)} \\ f_2(\boldsymbol{x}^*) = -\delta_{M1}x_{M1}^*, & \text{(B.3)} \\ f_3(\boldsymbol{x}^*) = -\delta_{M2}x_{M2}^*. & \text{(B.4)} \end{cases}$$

Eftersom $\delta_{M1}, \delta_{M2} > 0$ måste därmed $x_{M1}^* = x_{M2}^* = 0$ för att $\boldsymbol{x}^*$ ska vara ett steady state. Detta medför att det enda steady state för $x_T^* = 0$ är $\boldsymbol{x}^* = (0, 0, 0)$.

Antar vi istället att $x_T^* > 0$ och $x_{M1}^* = x_{M2}^* = 0$, förenklas $f(\boldsymbol{x}) = 0$ till

$$\begin{cases} f_1(\boldsymbol{x}^*) = rx_T^*\left(1 - \frac{x_T^*}{\beta_T}\right), & \text{(B.5)} \\ f_2(\boldsymbol{x}^*) = 0, & \text{(B.6)} \\ f_3(\boldsymbol{x}^*) = 0. & \text{(B.7)} \end{cases}$$

Eftersom $r, x_T^*, \beta_T > 0$ uppfylls endast kravet att $f(\boldsymbol{x}^*) = 0$ om $x_T^* = \beta_T$. Alltså är $\boldsymbol{x}^* = (\beta_T, 0, 0)$ det enda möjliga steady state för de gällande antagandena.

Vi antar nu att $x_T^*, x_{M2}^* > 0$ och $x_{M1}^* = 0$ vilket ger att förenklingen av $f(\boldsymbol{x}^*) = 0$ blir

$$\begin{cases} f_1(\boldsymbol{x}^*) = rx_T^*\left(1 - \frac{x_T^*}{\beta_T}\right) + d_{M2}x_{M2}^*x_T^* = 0, & \text{(B.8)} \\ f_2(\boldsymbol{x}^*) = 0, & \text{(B.9)} \\ f_3(\boldsymbol{x}^*) = a_{t2}x_T^*x_{M2}^*\left(1 - \frac{x_{M2}^*}{\beta_M}\right) - \delta_{M2}x_{M2}^* = 0. & \text{(B.10)} \end{cases}$$

Genom att bryta ut x_{M2}^* ur (B.8) ges att

$$x_{M2}^* = \frac{-rx_T^*(1 - \frac{x_T^*}{\beta_T})}{d_{M2}x_T^*} = \frac{r}{d_{M2}}(\frac{x_T^*}{\beta_T} - 1). \tag{B.11}$$

Detta kan sedan användas för att skriva om (B.10) som

$$a_{t2}x_T^*\left(1 - \frac{\frac{r}{d_{M2}}(\frac{x_T^*}{\beta_T} - 1)}{\beta_M}\right) - \delta_{M2} = 0, \tag{B.12}$$

vilket i sin tur kan förenklas till andragradsekvationen

$$(x_T^*)^2 - \frac{d_{M2}\beta_T\beta_M}{r}\left(1 + \frac{r}{d_{M2}\beta_M}\right)x_T^* + \frac{\delta_{M2}d_{M2}\beta_T\beta_M}{a_{t2}r} = 0. \tag{B.13}$$

Lösningarna till (B.13) är

$$x_T^* = \frac{d_{M2}\beta_T\beta_M(1+\frac{r}{d_{M2}\beta_M})}{2r} \pm \sqrt{\left(\frac{d_{M2}\beta_T\beta_M(1+\frac{r}{d_{M2}\beta_M})}{2r}\right)^2 - \frac{\delta_{M2}d_{M2}\beta_T\beta_M}{a_{t2}r}}, \tag{B.14}$$

vilket kan skrivas om till

$$x_T^* = \frac{d_{M2}\beta_T\beta_M}{2a_{t2}r}\left[\left(a_{t2}+\frac{a_{t2}r}{d_{M2}\beta_M}\right) \pm \sqrt{\left(a_{t2}+\frac{a_{t2}r}{d_{M2}\beta_M}\right)^2 - \frac{4a_{t2}r\delta_{M2}}{\beta_T\beta_M d_{M2}}}\right]. \tag{B.15}$$

Vi har alltså hittat ett tredje och fjärde steady state $x^* = \left(x_T^*, 0, \frac{r}{d_{M2}}\left(\frac{x_T^*}{\beta_T}-1\right)\right)$ där x_T^* beskrivs av (B.15). Dessa steady state existerar endast om $x_T^* \in \mathbb{R}$ och $x_T^* > 0$.

Nu antar vi att $x_T^*, x_{M1}^* > 0$ och $x_{M2}^* = 0$. Förenklingen av $f(\boldsymbol{x}^*) = 0$ ger då ekvationssystemet

$$\begin{cases} f_1(\boldsymbol{x}^*) = rx_T^*\left(1-\frac{x_T^*}{\beta_T}\right) - d_{M1}x_{M1}^*x_T^* = 0, & \text{(B.16)} \\ f_2(\boldsymbol{x}^*) = a_{t1}x_T^*x_{M1}^*(1-\frac{x_{M1}^*}{\beta_M}) - \delta_{M1}x_{M1}^* - k_{12}x_{M1}^*x_T^* = 0, & \text{(B.17)} \\ f_3(\boldsymbol{x}^*) = k_{12}x_{M1}^*x_T^* = 0. & \text{(B.18)} \end{cases}$$

Enligt antagande är $k_{12}, x_{M1}^*, x_T^* > 0$ vilket gör att (B.18) ej går att uppfylla. Under dessa antaganden finns därför inga steady state.

Till sist antar vi att $x_T^*, x_{M1}^*, x_{M2}^* > 0$ vilket ger att $f(\boldsymbol{x}^*) = 0$ förblir oförändrat, det vill säga

$$\begin{cases} rx_T^*\left(1-\frac{x_T^*}{\beta_T}\right) - d_{M1}x_{M1}^*x_T^* + d_{M2}x_{M2}^*x_T^* = 0, & \text{(B.19)} \\ a_{t1}x_T^*x_{M1}^*\left(1-\frac{x_{M1}^*+x_{M2}^*}{\beta_M}\right) - \delta_{M1}x_{M1}^* - k_{12}x_{M1}^*x_T^* = 0, & \text{(B.20)} \\ a_{t2}x_T^*x_{M2}^*\left(1-\frac{x_{M1}^*+x_{M2}^*}{\beta_M}\right) - \delta_{M2}x_{M2}^* + k_{12}x_{M1}^*x_T^* = 0. & \text{(B.21)} \end{cases}$$

Vi kan bryta ut x_{M1}^* ur (B.19) vilket ger oss

$$x_{M1}^* = \frac{r}{d_{M1}}\left(1-\frac{x_T^*}{\beta_T}\right) + \frac{d_{M2}}{d_{M1}}x_{M2}^*. \tag{B.22}$$

Vi kan skriva om (B.20) som

$$x_T^*\left(1-\frac{x_{M1}^*+x_{M2}^*}{\beta_M}\right) = \frac{\delta_{M1}}{a_{t1}} + \frac{k_{12}x_T^*}{a_{t1}}. \tag{B.23}$$

Om vi sedan sätter in (B.23) i (B.21) får vi

$$a_{t2}x_{M2}^*\left(\frac{\delta_{M1}}{a_{t1}} + \frac{k_{12}}{a_{t1}}x_T^*\right) - \delta_{M2}x_{M2}^* + k_{12}x_{M1}^*x_T^* = 0. \tag{B.24}$$

Insättning av (B.22) i (B.24) ger oss ekvationen

$$a_{t2}x_{M2}^*\left(\frac{\delta_{M1}}{a_{t1}} + \frac{k_{12}}{a_{t1}}x_T^*\right) - \delta_{M2}x_{M2}^* + k_{12}x_T^*\left(\frac{d_{M2}x_{M2}^*}{d_{M1}} + \frac{r}{d_{M1}}\left(1-\frac{x_T^*}{\beta_T}\right)\right) = 0. \tag{B.25}$$

Detta innebär att vi har steady state, $x^* = (x_T^*, x_{M1}^*, x_{M2}^*)$, givna implicit av (B.22) och (B.25). □

Bevis av sats 4.3. Låt $f = \dot{x}$, det vill säga

$$\begin{cases} f_1 = rx_T\left(1-\frac{x_T}{\beta_T}\right) - d_{M1}x_{M1}x_T + d_{M2}x_{M2}x_T, & \text{(B.26)} \\ f_2 = a_{t1}x_Tx_{M1}\left(1-\frac{x_{M1}+x_{M2}}{\beta_M}\right) - \delta_{M1}x_{M1} - k_{12}x_{M1}x_T, & \text{(B.27)} \\ f_3 = a_{t2}x_Tx_{M2}\left(1-\frac{x_{M1}+x_{M2}}{\beta_M}\right) - \delta_{M2}x_{M2} + k_{12}x_{M1}x_T. & \text{(B.28)} \end{cases}$$

Notera att f är differentierbar i punkterna $\boldsymbol{x}_1^* = (0,0,0)$ och $\boldsymbol{x}_2^* = (\beta_T, 0, 0)$ samt att $f(\boldsymbol{x}_1^*) = f(\boldsymbol{x}_2^*) = 0$. Så vi kan använda oss av sats 4.2.

Vi börjar med att ta fram matrisen A för $\boldsymbol{x}_1^* = (0,0,0)$. För det behöver vi de partiella derivatorna av f med avseende på x_T, x_{M1} och x_{M2}. Derivatorna är följande:

- $\frac{\partial f_1}{\partial x_T}(\boldsymbol{x}) = r(1 - \frac{2x_T}{\beta_T}) - d_{M1}x_{M1} + d_{M2}x_{M2}$,
- $\frac{\partial f_1}{\partial x_{M1}}(\boldsymbol{x}) = -d_{M1}x_T$,
- $\frac{\partial f_1}{\partial x_{M2}}(\boldsymbol{x}) = d_{M2}x_T$,
- $\frac{\partial f_2}{\partial x_T}(\boldsymbol{x}) = a_{t1}x_{M1}(1 - \frac{x_{M1}+x_{M2}}{\beta_M}) - k_{12}x_{M1}$,
- $\frac{\partial f_2}{\partial x_{M1}}(\boldsymbol{x}) = a_{t1}x_T(1 - \frac{2x_{M1}+x_{M2}}{\beta_M}) - \delta_{M1} - k_{12}x_T$,
- $\frac{\partial f_2}{\partial x_{M2}}(\boldsymbol{x}) = -\frac{a_{t1}x_Tx_{M1}}{\beta_M}$,
- $\frac{\partial f_3}{\partial x_T}(\boldsymbol{x}) = a_{t2}x_{M2}(1 - \frac{x_{M1}+x_{M2}}{\beta_M}) + k_{12}x_{M1}$,
- $\frac{\partial f_3}{\partial x_{M1}}(\boldsymbol{x}) = -\frac{a_{t2}x_Tx_{M2}}{\beta_M} + k_{12}x_T$,
- $\frac{\partial f_3}{\partial x_{M2}}(\boldsymbol{x}) = a_{t2}x_T(1 - \frac{x_{M1}+2x_{M2}}{\beta_M}) - \delta_{M2}$.

Dessa derivator samt insättning av punkten $\boldsymbol{x}_1^*$ ger oss matrisen

$$A = \begin{bmatrix} r & 0 & 0 \\ 0 & -\delta_{M1} & 0 \\ 0 & 0 & -\delta_{M2} \end{bmatrix},$$

som har egenvärdena $r, -\delta_{M1}$ och $-\delta_{M2}$. Eftersom $r > 0$ enligt antagande så säger sats 4.2 att punkten $\boldsymbol{x}_1^*$ är ett instabilt steady state.

Genom att förskjuta funktionen f från $f(\boldsymbol{x})$ till $f(\boldsymbol{x} - \boldsymbol{x}_2^*)$ kan vi ta fram motsvarande matris A för punkten $\boldsymbol{x}_2^*$. Vi kallar den nya förskjutna funktionen för $\tilde{f}$ vilken ser ut på följande sätt.

$$\begin{cases} \tilde{f}_1 = r(x_T - \beta_T)\left(1 - \dfrac{x_T - \beta_T}{\beta_T}\right) - d_{M1}x_{M1}(x_T - \beta_T) + d_{M2}x_{M2}(x_T - \beta_T), & \text{(B.29)} \\ \tilde{f}_2 = a_{t1}(x_T - \beta_T)x_{M1}\left(1 - \dfrac{x_{M1} + x_{M2}}{\beta_M}\right) - \delta_{M1}x_{M1} - k_{12}x_{M1}(x_T - \beta_T), & \text{(B.30)} \\ \tilde{f}_3 = a_{t2}(x_T - \beta_T)x_{M2}\left(1 - \dfrac{x_{M1} + x_{M2}}{\beta_M}\right) - \delta_{M2}x_{M2} + k_{12}x_{M1}(x_T - \beta_T). & \text{(B.31)} \end{cases}$$

De nya derivatorna med avseende på x_T, x_{M1} och x_{M2} är följande:

- $\frac{\partial \tilde{f}_1}{\partial x_T}(\boldsymbol{x}) = r - \frac{2r(x_T-\beta_T)}{\beta_T} - d_{M1}x_{M1} + d_{M2}x_{M2}$,
- $\frac{\partial \tilde{f}_1}{\partial x_{M1}}(\boldsymbol{x}) = -d_{M1}(x_T - \beta_T)$,
- $\frac{\partial \tilde{f}_1}{\partial x_{M2}}(\boldsymbol{x}) = d_{M2}(x_T - \beta_T)$,
- $\frac{\partial \tilde{f}_2}{\partial x_T}(\boldsymbol{x}) = a_{t1}x_{M1}(1 - \frac{x_{M1}+x_{M2}}{\beta_M}) - k_{12}x_{M1}$,
- $\frac{\partial \tilde{f}_2}{\partial x_{M1}}(\boldsymbol{x}) = a_{t1}(x_T - \beta_T)(1 - \frac{2x_{M1}+x_{M2}}{\beta_M}) - \delta_{M1} - k_{12}(x_T - \beta_T)$,
- $\frac{\partial \tilde{f}_2}{\partial x_{M2}}(\boldsymbol{x}) = -\frac{a_{t1}(x_T-\beta_T)x_{M1}}{\beta_M}$,
- $\frac{\partial \tilde{f}_3}{\partial x_T}(\boldsymbol{x}) = a_{t2}x_{M2}(1 - \frac{x_{M1}+x_{M2}}{\beta_M}) + k_{12}x_{M1}$,
- $\frac{\partial \tilde{f}_3}{\partial x_{M1}}(\boldsymbol{x}) = -\frac{a_{t2}(x_T-\beta_T)x_{M2}}{\beta_M} + k_{12}(x_T - \beta_T)$,

- $\frac{\partial \tilde{f}_3}{\partial x_{M2}}(\boldsymbol{x}) = a_{t2}(x_T - \beta_T)(1 - \frac{x_{M1}+2x_{M2}}{\beta_M}) - \delta_{M2}$.

Dessa derivator samt insättning av punkten $(0,0,0)$ ger oss matrisen

$$A = \begin{bmatrix} 3r & d_{M1}\beta_T & -d_{M2}\beta_T \\ 0 & (k_{12} - a_{t1})\beta_T - \delta_{M1} & 0 \\ 0 & -\beta_T k_{12} & -a_{t2}\beta_T - \delta_{M2} \end{bmatrix}.$$

Notera att vi sätter in punkten $(0,0,0)$ istället för $\boldsymbol{x}_2^*$ på grund av att vi använder den förskjutna funktionen $\tilde{f}$.

Vi beräknar egenvärdena, λ, till A med den karaktäristiska ekvationen

$$det(A - \lambda I) = 0,$$

vilket i vårt fall blir

$$\begin{aligned} det(A - \lambda I) &= det \begin{pmatrix} 3r - \lambda & d_{M1}\beta_T & -d_{M2}\beta_T \\ 0 & (k_{12} - a_{t1})\beta_T - \delta_{M1} - \lambda & 0 \\ 0 & -\beta_T k_{12} & -a_{t2}\beta_T - \delta_{M2} - \lambda \end{pmatrix} = \\ &= (3r - \lambda) det \begin{pmatrix} (k_{12} - a_{t1})\beta_T - \delta_{M1} - \lambda & 0 \\ -\beta_T k_{12} & -a_{t2}\beta_T - \delta_{M2} - \lambda \end{pmatrix} = \\ &= (3r - \lambda)((k_{12} - a_{t1})\beta_T - \delta_{M1} - \lambda)(-a_{t2}\beta_T - \delta_{M2} - \lambda) = 0. \end{aligned}$$

Lösningarna till den karaktäristiska ekvationen är

$$\begin{cases} \lambda_1 = 3r \\ \lambda_2 = \beta_T(k_{12} - a_{t1}) - \delta_{M1} \\ \lambda_3 = -a_{t2}\beta_T - \delta_{M2}. \end{cases}$$

Dessa lösningar är även våra egenvärden. Eftersom $r > 0$ enligt antagande, så är $\lambda_1 = 3r > 0$. Enligt sats 4.2 är då punkten $\boldsymbol{x}_2^* = (\beta_T, 0, 0)$ ett instabilt steady state.

Därav är satsen bevisad. □

B.2 Derivator

Bevis av sats 5.2. Under antagandet att $\boldsymbol{\lambda}, \boldsymbol{x}$ och $\boldsymbol{\alpha}$ kan varieras oberoende av varandra blir de partiella Fréchetderivatorna av L med avseende på $\boldsymbol{\lambda}$ följande.

$$\begin{aligned} L'_{\lambda_1}(\boldsymbol{v})(\overline{\lambda}_1) &= \frac{\partial}{\partial \lambda_1}\left(\int_0^T \lambda_1(t)(\dot{x_1}(t) - f_1(t))dt)\overline{\lambda}_1\right) = \left(\int_0^T \frac{dx_T}{dt} - f_1 dt\right)\overline{\lambda}_1 = \\ &= \int_0^T (\frac{dx_T}{dt} - r x_T(1 - \frac{x_T}{\beta_T}) + d_{M1}x_{M1}x_T - d_{M2}x_{M2}x_T)\overline{\lambda}_1 dt. \end{aligned} \tag{B.32a}$$

$$\begin{aligned} L'_{\lambda_2}(\boldsymbol{v})(\overline{\lambda}_2) &= \frac{\partial}{\partial \lambda_2}\left(\int_0^T \lambda_2(t)(\dot{x_2}(t) - f_2(t))dt\right)\overline{\lambda}_2 = \left(\int_0^T \frac{dx_{M1}}{dt} - f_2 dt\right)\overline{\lambda}_2 = \\ &= \int_0^T (\frac{dx_{M1}}{dt} - a_{t1}x_T x_{M1}(1 - \frac{x_{M1} + x_{M2}}{\beta_M}) + \delta_{M1}x_{M1} + k_{12}x_{M1}x_T)\overline{\lambda}_2 dt. \end{aligned} \tag{B.32b}$$

$$\begin{aligned} L'_{\lambda_3}(\boldsymbol{v})(\overline{\lambda}_3) &= \frac{\partial}{\partial \lambda_3}\left(\int_0^T \lambda_3(t)(\dot{x_3}(t) - f_3(t))dt\right)\overline{\lambda}_3 = \left(\int_0^T \frac{dx_{M2}}{dt} - f_3 dt\right)\overline{\lambda}_3 = \\ &= \int_0^T (\frac{dx_{M2}}{dt} - a_{t2}x_T x_{M2}(1 - \frac{x_{M1} + x_{M2}}{\beta_M}) + \delta_{M2}x_{M2} - k_{12}x_{M1}x_T)\overline{\lambda}_3 dt. \end{aligned} \tag{B.32c}$$

Notera att alla termer inte skrivs med i första steget. Alla termer som inte innehåller λ_1, λ_2 respektive λ_3 kommer bli 0 när vi deriverar med avseende på motsvarande parameter.

Genom att sätta de olika derivatorna till noll är sats 5.2 bevisad.

□

Bevis av sats 5.3. Under antagandet att $\boldsymbol{\lambda}, \boldsymbol{x}$ och $\boldsymbol{\alpha}$ kan varieras oberoende av varandra kan vi beräkna de partiella Fréchetderivatorna av L med avseende på x_T på följande sätt.

$$L'_{x_T}(\boldsymbol{v})(\overline{x}_T) = \frac{\partial}{\partial x_T}\left(\frac{1}{2}\int_0^T (x_1(t) - g_1^c(t))^2 dt + \sum_{i=1}^{3}\int_0^T \lambda_i(t)(\dot{x_i}(t) - f_i(t))dt\right)\bar{x}_T. \tag{B.33}$$

Vi delar upp i fyra mindre beräkningar, en för varje term i funktionen som deriveras.

- $\frac{\partial}{\partial x_T}\ (\frac{1}{2}\int_0^T (x_1(t) - g_1^c(t))^2 dt) = \frac{1}{2}\frac{\partial}{\partial x_T}(\int_0^T (x_T - g_1^c)^2 dt) = \frac{1}{2}(\int_0^T 2(x_T - g_1^c)dt =$
  $= \int_0^T (x_T - g_1^c)dt,$
- $\frac{\partial}{\partial x_T}\ (\int_0^T \lambda_1(t)(\dot{x_1}(t) - f_1(t))dt) = \frac{\partial}{\partial x_T}(\int_0^T \lambda_1 \frac{dx_T}{dt}dt) - \frac{\partial}{\partial x_T}(\int_0^T \lambda_1 f_1 dt) =$
  $= \{\text{Partiell integration}\} = \frac{\partial}{\partial x_T}([\lambda_1 x_T]_0^T - \int_0^T \frac{d\lambda_1}{dt}x_T dt) - \frac{\partial}{\partial x_T}(\int_0^T \lambda_1(r x_T(1 - \frac{x_T}{\beta_T})$
  $- d_{M1}x_{M1}x_T + d_{M2}x_{M2}x_T)dt) = \int_0^T -\frac{d\lambda_1}{dt} - \lambda_1 r(1 - \frac{2x_T}{\beta_T}) + \lambda_1 d_{M1}x_{M1} - \lambda_1 d_{M2}x_{M2}dt,$
- $\frac{\partial}{\partial x_T}\ (\int_0^T \lambda_2(t)(\dot{x_2}(t) - f_2(t))dt) = \frac{\partial}{\partial x_T}(\int_0^T -\lambda_2 f_2 dt) = \frac{\partial}{\partial x_T}(-\int_0^T \lambda_2(a_{t1}x_T x_{M1}(1 - \frac{x_{M1}+x_{M2}}{\beta_M})$
  $- \delta_{M1}x_{M1} - k_{12}x_{M1}x_T)dt) = \int_0^T -\lambda_2 a_{t1}x_{M1}(1 - \frac{x_{M1}+x_{M2}}{\beta_M}) + \lambda_2 k_{12}x_{M1}dt,$
- $\frac{\partial}{\partial x_T}\ (\int_0^T \lambda_3(t)(\dot{x_3}(t) - f_3(t))dt) = \frac{\partial}{\partial x_T}(\int_0^T -\lambda_3 f_3 dt) = \frac{\partial}{\partial x_T}(-\int_0^T \lambda_3(a_{t2}x_T x_{M2}(1 - \frac{x_{M1}+x_{M2}}{\beta_M})$
  $- \delta_{M2}x_{M2} + k_{12}x_{M1}x_T)dt) = \int_0^T -\lambda_3 a_{t2}x_{M2}(1 - \frac{x_{M1}+x_{M2}}{\beta_M}) - \lambda_3 k_{12}x_{M1}dt.$

Det ger oss följande uttryck för derivatan:

$$L'_{x_T}(\boldsymbol{v})(\overline{x}_T) = \int_0^T (x_T - g_1^c)\overline{x}_T dt + \int_0^T \Bigg(-\frac{d\lambda_1}{dt} - \lambda_1 r(1 - \frac{2x_T}{\beta_T}) + \lambda_1 d_{M1}x_{M1} - \lambda_1 d_{M2}x_{M2}$$
$$- \lambda_2 a_{t1}x_{M1}(1 - \frac{x_{M1}+x_{M2}}{\beta_M}) + \lambda_2 k_{12}x_{M1} - \lambda_3 a_{t2}x_{M2}(1 - \frac{x_{M1}+x_{M2}}{\beta_M}) - \lambda_3 k_{12}x_{M1})\Bigg)\overline{x}_T dt.$$

De partiella Fréchetderivatorna av L med avseende på x_{M1} beräknas på följande sätt.

$$L'_{x_{M1}}(\boldsymbol{v})(\overline{x}_{M1}) = \frac{\partial}{\partial x_{M1}}\left(\frac{1}{2}\int_0^T (x_2(t) - g_2^c(t))^2 dt + \sum_{i=1}^{3}\int_0^T \lambda_i(t)(\dot{x_i}(t) - f_i(t))dt\right)\overline{x}_{M1}. \tag{B.34}$$

Vi delar upp i fyra delberäkningar, en för varje term.

- $\frac{\partial}{\partial x_{M1}}\ (\frac{1}{2}\int_0^T (x_2(t) - g_2^c(t))^2 dt) = \frac{1}{2}\frac{\partial}{\partial x_{M1}}(\int_0^T (x_{M1} - g_2^c)^2 dt) = \frac{1}{2}(\int_0^T 2(x_{M1} - g_2^c)dt) =$
  $= \int_0^T (x_{M1} - g_2^c)dt,$
- $\frac{\partial}{\partial x_{M1}}\ (\int_0^T \lambda_1(t)(\dot{x_1}(t) - f_1(t))dt) = \frac{\partial}{\partial x_{M1}}(\int_0^T -\lambda_1 f_1 dt) = \frac{\partial}{\partial x_{M1}}(-\int_0^T \lambda_1(r x_T(1 - \frac{x_T}{\beta_T}) - d_{M1}x_{M1}x_T$
  $+ d_{M2}x_{M2}x_T)dt) = \int_0^T \lambda_1 d_{M1}x_T dt,$
- $\frac{\partial}{\partial x_{M1}}\ (\int_0^T \lambda_2(t)(\dot{x_2}(t) - f_2(t))dt) = \frac{\partial}{\partial x_{M1}}(\int_0^T \lambda_2 \frac{\partial x_{M1}}{\partial t}dt) - \frac{\partial}{\partial x_{M1}}(\int_0^T \lambda_2 f_2 dt) = \{\text{Partiell integration}\}$
  $= \frac{\partial}{\partial x_{M1}}([\lambda_2 x_{M1}]_0^T - \int_0^T \frac{d\lambda_2}{dt}x_{M1}dt) - \frac{\partial}{\partial x_{M1}}(\int_0^T \lambda_2(a_{t1}x_T x_{M1}(1 - \frac{x_{M1}+x_{M2}}{\beta_M}) - \delta_{M1}x_{M1}$
  $- k_{12}x_{M1}x_T)dt) = \int_0^T -\frac{d\lambda_2}{dt} - \lambda_2 a_{t1}x_T(1 - \frac{2x_{M1}+x_{M2}}{\beta_M}) + \lambda_2\delta_{M1} + \lambda_2 k_{12}x_T dt,$
- $\frac{\partial}{\partial x_{M1}}\ (\int_0^T \lambda_3(t)(\dot{x_3}(t) - f_3(t))dt) = \frac{\partial}{\partial x_{M1}}(\int_0^T -\lambda_3 f_3 dt) = \frac{\partial}{\partial x_{M1}}(-\int_0^T \lambda_3(a_{t2}x_T x_{M2}(1 - \frac{x_{M1}+x_{M2}}{\beta_M})$
  $- \delta_{M2}x_{M2} + k_{12}x_{M1}x_T)dt) = \int_0^T \frac{\lambda_3 a_{t2}x_T x_{M2}}{\beta_M} - \lambda_3 k_{12}x_T dt.$

Detta ger oss följande uttryck.

$$L'_{x_{M1}}(\boldsymbol{v})(\overline{x}_{M1}) = \int_0^T (x_{M1} - g_2^c)\overline{x}_{M1}dt + \int_0^T (\lambda_1 d_{M1}x_T - \frac{d\lambda_2}{dt} - \lambda_2 a_{t1}x_T(1 - \frac{2x_{M1} + x_{M2}}{\beta_M}) + \lambda_2\delta_{M1} + \lambda_2 k_{12}x_T + \frac{\lambda_3 a_{t2}x_T x_{M2}}{\beta_M} - \lambda_3 k_{12}x_T)\overline{x}_{M1}dt.$$

De partiella Fréchetderivatorna av L med avseende på x_{M2} beräknas på samma sätt.

$$L'_{x_{M2}}(\boldsymbol{v})(\overline{x}_{M2}) = \frac{\partial}{\partial x_{M2}}\left(\frac{1}{2}\int_0^T (x_3(t) - g_3^c(t))^2 dt + \sum_{i=1}^{3}\int_0^T \lambda_i(t)(\dot{x_i}(t) - f_i(t))dt\right)\overline{x}_{M1}. \quad \text{(B.35)}$$

Vi delar upp i fyra delberäkningar, en för varje term.

- $\frac{\partial}{\partial x_{M2}}\ (\frac{1}{2}\int_0^T (x_3(t) - g_3^c(t))^2 dt) = \frac{1}{2}\frac{\partial}{\partial x_{M2}}(\int_0^T (x_{M2} - g_3^c)^2 dt) = \frac{1}{2}(\int_0^T 2(x_{M2} - g_3^c)dt) = \int_0^T (x_{M2} - g_3^c)dt,$
- $\frac{\partial}{\partial x_{M2}}\ (\int_0^T \lambda_1(t)(\dot{x_1}(t) - f_1(t))dt) = \frac{\partial}{\partial x_{M2}}(\int_0^T -\lambda_1 f_1 dt) = \frac{\partial}{\partial x_{M2}}(-\int_0^T \lambda_1(rx_T(1 - \frac{x_T}{\beta_T}) - d_{M1}x_{M1}x_T + d_{M2}x_{M2}x_T)dt) = \int_0^T -\lambda_1 d_{M2}x_T dt,$
- $\frac{\partial}{\partial x_{M2}}\ (\int_0^T \lambda_2(t)(\dot{x_2}(t) - f_2(t))dt) = \frac{\partial}{\partial x_{M2}}(\int_0^T -\lambda_2 f_2 dt) = \frac{\partial}{\partial x_{M2}}(-\int_0^T \lambda_2(a_{t1}x_T x_{M1}(1 - \frac{x_{M1}+x_{M2}}{\beta_M}) - \delta_{M1}x_{M1} - k_{12}x_{M1}x_T)dt) = \int_0^T \frac{\lambda_2 a_{t1}x_T x_{M1}}{\beta_M}dt,$
- $\frac{\partial}{\partial x_{M2}}\ (\int_0^T \lambda_3(t)(\dot{x_3}(t) - f_3(t))dt) = \frac{\partial}{\partial x_{M2}}(\int_0^T \lambda_3\frac{dx_{M2}}{dt}dt) - \frac{\partial}{\partial x_{M2}}(\int_0^T \lambda_3 f_3 dt) = \{\text{Partiell integration}\} = \frac{\partial}{\partial x_{M2}}([\lambda_3 x_{M2}]_0^T - \int_0^T \frac{d\lambda_3}{dt}x_{M2}dt) - \frac{\partial}{\partial x_{M2}}(\int_0^T \lambda_3(a_{t2}x_T x_{M2}(1 - \frac{x_{M1}+x_{M2}}{\beta_M}) - \delta_{M2}x_{M2} + k_{12}x_{M1}x_T)dt) = \int_0^T -\frac{d\lambda_3}{dt} - \lambda_3 a_{t2}x_T(1 - \frac{x_{M1}+2x_{M2}}{\beta_M}) + \lambda_3\delta_{M2}dt.$

Detta ger oss följande uttryck för derivatan.

$$L'_{x_{M2}}(\boldsymbol{v})(\overline{x}_{M2}) = \int_0^T (x_{M2} - g_3^c)\overline{x}_{M2}dt + \int_0^T (-\lambda_1 d_{M2}x_T + \frac{\lambda_2 a_{t1}x_T x_{M1}}{\beta_M} - \frac{d\lambda_3}{dt} - \lambda_3 a_{t2}x_T(1 - \frac{x_{M1} + 2x_{M2}}{\beta_M}) + \lambda_3\delta_{M2})\overline{x}_{M2}dt.$$

Notera att i första steget av deriveringarna av L så skrivs inte termer som deriveras till noll med.

Genom att sätta de olika derivatorna till noll är sats 5.3 bevisad. □

Bevis av sats 5.4. Under antagandet att $\boldsymbol{\lambda}, \boldsymbol{x}$ och $\boldsymbol{\alpha}$ kan varieras oberoende av varandra blir de partiella Fréchetderivatorna av L med avseende på $\boldsymbol{\alpha}$ följande.

$$\begin{aligned} L'_{d_{M1}}(\boldsymbol{v})(\overline{d}_{m1}) &= \frac{\partial}{\partial d_{M1}}\left(\frac{1}{2}\gamma_1\int_0^T (d_{M1}(t) - d_{M1}^0)^2 dt + \int_0^T \lambda_1(t)(\dot{x_1}(t) - f_1(t))dt\right)\overline{d}_{m1} \\ &= \left(\frac{1}{2}\gamma_1\int_0^T 2(d_{M1} - d_{M1}^0)dt + \frac{\partial}{\partial d_{M1}}\left(\int_0^T -\lambda_1 f_1 dt\right)\right)\overline{d}_{m1} = \\ &= \gamma_1\int_0^T (d_{M1} - d_{M1}^0)\overline{d}_{m1}dt + \frac{\partial}{\partial d_{M1}}\Big(\int_0^T -\lambda_1(rx_T(1 - \frac{x_T}{\beta_T}) \\ &\quad - d_{M1}x_{M1}x_T + d_{M2}x_{M2}x_T)dt\Big)\overline{d}_{m1} = \\ &= \gamma_1\int_0^T (d_{M1} - d_{M1}^0)\overline{d}_{m1}dt + \int_0^T \lambda_1 x_{M1}x_T\overline{d}_{m1}dt \end{aligned} \quad \text{(B.36a)}$$

$$L'_{d_{M2}}(\boldsymbol{v})(\overline{d}_{m2}) = \frac{\partial}{\partial d_{M2}}\left(\frac{1}{2}\gamma_2\int_0^T (d_{M2}(t)-d^0_{M2})^2dt + \int_0^T \lambda_1(t)(\dot{x_1}(t)-f_1(t))dt\right)\overline{d}_{m2} =$$

$$= \frac{1}{2}\gamma_2\int_0^T 2(d_{M2}-d^0_{M2})\overline{d}_{m2}dt + \frac{\partial}{\partial d_{M2}}\left(\int_0^T -\lambda_1 f_1 dt\right)\overline{d}_{m2} =$$

$$= \gamma_2\int_0^T (d_{M2}-d^0_{M2})\overline{d}_{m2}dt + \frac{\partial}{\partial d_{M2}}\Big(\int_0^T -\lambda_1(rx_T(1-\frac{x_T}{\beta_T})$$

$$- d_{M1}x_{M1}x_T + d_{M2}x_{M2}x_T)dt\Big)\overline{d}_{m2} =$$

$$= \gamma_2\int_0^T (d_{M2}-d^0_{M2})\bar{d}_{m2}dt - \int_0^T \lambda_1 x_{M2}x_T\overline{d}_{m2}dt \tag{B.36b}$$

$$L'_{a_{t1}}(\boldsymbol{v})(\overline{a}_{t1}) = \frac{\partial}{\partial a_{t1}}\left(\frac{1}{2}\gamma_3\int_0^T (a_{t1}(t)-a^0_{t1})^2dt + \int_0^T \lambda_2(t)(\dot{x_2}(t)-f_2(t))dt\right)\overline{a}_{t1} =$$

$$= \frac{1}{2}\gamma_3\int_0^T 2(a_{t1}-a^0_{t1})\overline{a}_{t1}dt + \frac{\partial}{\partial a_{t1}}\left(\int_0^T -\lambda_2 f_2 dt\right)\overline{a}_{t1} =$$

$$= \gamma_3\int_0^T (a_{t1}-a^0_{t1})\overline{a}_{t1}dt + \frac{\partial}{\partial a_{t1}}\Big(\int_0^T -\lambda_2(a_{t1}x_Tx_{M1}(1-\frac{x_{M1}+x_{M2}}{\beta_M}) \tag{B.36c}$$

$$- \delta_{M1}x_{M1} - k_{12}x_{M1}x_T)dt\Big)\overline{a}_{t1} =$$

$$= \gamma_3\int_0^T (a_{t1}-a^0_{t1})\overline{a}_{t1}dt - \int_0^T \lambda_2 x_{M1}x_T(1-\frac{x_{M1}+x_{M2}}{\beta_M})\overline{a}_{t1}dt$$

$$L'_{a_{t2}}(\boldsymbol{v})(\overline{a}_{t2}) = \frac{\partial}{\partial a_{t2}}\left(\frac{1}{2}\gamma_4\int_0^T (a_{t2}(t)-a^0_{t2})^2dt + \int_0^T \lambda_3(t)(\dot{x_3}(t)-f_3(t))dt\right)\overline{a}_{t2} =$$

$$= \frac{1}{2}\gamma_4\int_0^T 2(a_{t2}-a^0_{t2})\overline{a}_{t2}dt + \frac{\partial}{\partial a_{t2}}\left(\int_0^T -\lambda_3 f_3 dt\right)\overline{a}_{t2} =$$

$$= \gamma_4\int_0^T (a_{t2}-a^0_{t2})\overline{a}_{t2}dt + \frac{\partial}{\partial a_{t2}}\Big(\int_0^T -\lambda_3(a_{t2}x_Tx_{M2}(1-\frac{x_{M1}+x_{M2}}{\beta_M}) \tag{B.36d}$$

$$- \delta_{M2}x_{M2} + k_{12}x_{M1}x_T)dt\Big)\overline{a}_{t2} =$$

$$= \gamma_4\int_0^T (a_{t2}-a^0_{t2})\overline{a}_{t2}dt - \int_0^T \lambda_3 x_{M2}x_T(1-\frac{x_{M1}+x_{M2}}{\beta_M})\overline{a}_{t2}dt$$

$$L'_{k_{12}}(\boldsymbol{v})(\overline{k}_{12}) = \frac{\partial}{\partial k_{12}}\left(\frac{1}{2}\gamma_5\int_0^T (k_{12}(t)-k_{12}^0)^2dt + \sum_{i=2}^{3}\int_0^T \lambda_i(t)(\dot{x_i}(t)-f_i(t))dt\right)\overline{k}_{12}$$

- $\dfrac{\partial}{\partial k_{12}}\left(\dfrac{1}{2}\gamma_5\displaystyle\int_0^T (k_{12}-k_{12}^0)^2dt\right) = \dfrac{1}{2}\gamma_5\displaystyle\int_0^T 2(k_{12}-k_{12}^0)dt = \gamma_5\int_0^T k_{12}-k_{12}^0dt$
- $\dfrac{\partial}{\partial k_{12}}\left(\displaystyle\int_0^T \lambda_2(\dot{x}_2-f_2)dt\right) = \dfrac{\partial}{\partial k_{12}}\left(\displaystyle\int_0^T -\lambda_2 f_2 dt\right) =$

  $= \dfrac{\partial}{\partial k_{12}}\left(\displaystyle\int_0^T -\lambda_2(a_{t1}x_T x_{M1}(1-\frac{x_{M1}+x_{M2}}{\beta_M})-\delta_{M1}x_{M1}-k_{12}x_{M1}x_T)dt\right) =$

  $= \displaystyle\int_0^T \lambda_2 x_{M1}x_T dt$
- $\dfrac{\partial}{\partial k_{12}}\left(\displaystyle\int_0^T \lambda_3(\dot{x}_3-f_3)dt\right) = \dfrac{\partial}{\partial k_{12}}\left(\displaystyle\int_0^T -\lambda_3 f_3 dt\right) =$

  $= \dfrac{\partial}{\partial k_{12}}\left(\displaystyle\int_0^T -\lambda_3(a_{t2}x_T x_{M2}(1-\frac{x_{M1}+x_{M2}}{\beta_M})-\delta_{M2}x_{M2}+k_{12}x_{M1}x_T)dt\right) =$

  $= -\displaystyle\int_0^T \lambda_3 x_{M1}x_T dt$

$$\Rightarrow L'_{k_{12}}(\boldsymbol{v})(\overline{k}_{12}) = \gamma_5\int_0^T (k_{12}-k_{12}^0)\overline{k}_{12}dt + \int_0^T x_{M1}x_T(\lambda_2-\lambda_3)\overline{k}_{12}dt \tag{B.36e}$$

Notera att i första steget av deriveringarna av L så skrivs inte termer som deriveras till noll med. Genom att sätta de olika derivatorna till noll är sats 5.4 bevisad. □

B.3 Härledning av andra ordningens finita differens

Härledning av första ordningens approximation följer med Taylors sats

$$f(x+\tau) = f(x) + f'(x)\tau + \frac{f''(\xi)\tau^2}{2!} \tag{B.37}$$

Här är $f(x)$ funktion som vi vill approximera, x är punkten kring approximationen görs, τ är steglängden och ξ är en punkt som ligger mellan x och $x+\tau$.

$$\begin{aligned} f(x+\tau)-f(x) &= f'(x)\tau + \frac{f''(\xi)\tau^2}{2!} \\ \frac{f(x+\tau)-f(x)}{\tau} &= f'(x) + \frac{f''(\xi)\tau}{2!} \\ f'(x) &= \frac{f(x+\tau)-f(x)}{\tau} - \frac{f''(\xi)\tau}{2!} \end{aligned}$$

I praktiken är termen $\frac{f''(\xi)\tau}{2!}$ ofta liten, därmed kan derivatan approximeras genom att utesluta denna.

$$f'(x) \approx \frac{f(x+\tau)-f(x)}{\tau} \tag{B.38}$$

På samma sätt kan andra ordningens approximation härledas.

$$(*) f(x-\tau) = f(x) - f'(x)\tau + \frac{f''(\xi)\tau^2}{2!} + \frac{f'''(\xi)\tau^3}{3!} + \dots$$

$$(**) f(x+\tau) = f(x) + f'(x)\tau + \frac{f''(\xi)\tau^2}{2!} - \frac{f'''(\xi)\tau^3}{3!} + \dots$$

$$(*) - (**) : f(x+\tau) - f(x-\tau) = 2f'(x)\tau + 2\frac{f'''(\xi)\tau^3}{3!}$$

$$2f'(x)\tau = f(x+\tau) - f(x-\tau) - 2\frac{f'''(\xi)\tau^3}{3!}$$

$$f'(x) = \frac{f(x+\tau) - f(x-\tau)}{2\tau} - \frac{f'''(\xi)\tau^2}{3!}$$

C Appendix 3 – Newtons metod

I detta avsnitt beskrivs Newtons metod, tagen från [6], för lösning av det så kallade framåtproblemet respektive bakåtproblemet. Benämningen på problemen härstammar ifrån vilken riktning problemet löses, det vill säga framåtproblemet löses framåt i tiden medan bakåtproblemet löses bakåt i tiden.

C.1 Newtons metod för lösning av framåtproblemet

Betrakta differentialekvationen som beskriver systemets beteende:

$$\begin{cases} \frac{dx}{dt} = f(x(t), \alpha(t)), \ t \in [0, T], \\ x(0) = x_0, \end{cases} \tag{C.1}$$

här beskriver $x(t)$ systemets tillstånd vid tidpunkt t och $\alpha(t)$ representerar en vektor tidberoende parametrar.

Detta löses framåt i tiden genom att den diskretiseras med tidssteg τ. Detta resulterar i representationen

$$f(x^{k+1}, \alpha) = \frac{x^{k+1} - x^k}{\tau}, \tag{C.2}$$

där x^k och x^{k+1} är värdet på x vid tidpunkt k respektive $k+1$.

Ekvation (C.2) kan sedan omformuleras till följande två ekvationer

$$x^{k+1} = \tau f(x^{k+1}, \alpha) + x^k, \tag{C.3a}$$

$$x^{k+1} - \tau f(x^{k+1}, \alpha) - x^k = 0. \tag{C.3b}$$

Genom att införa en ny variabel $v := x^{k+1}$ kan funktionen $F(v)$ definieras enligt

$$F(v) := v - \tau f(v, \alpha) - x^k = 0. \tag{C.4}$$

Newtons metod för att lösa denna typen av problem kan då skrivas enligt

$$v^{n+1} = v^n - (F'(v^n))^{-1} \cdot F(v^n), \tag{C.5}$$

där n är antalet iterationer.

För att hitta $F'(v^n)$ används (C.4) vilket ger

$$F'(v^n) = I - \tau J(v^n), \tag{C.6}$$

där I är identitetsmatrisen och $J_f(v^n) := f'(v^n)$ är Jacobimatrisen av f i v^n. Jacobimatrisen i sig kan bestämmas enligt följande ekvation

$$J_f(v) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \frac{\partial f_1}{\partial x_3} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_3} \\ \frac{\partial f_3}{\partial x_1} & \frac{\partial f_3}{\partial x_2} & \frac{\partial f_3}{\partial x_3} \end{bmatrix} (v). \tag{C.7}$$

C.2 Newtons metod för lösning av bakåtproblemet

Lösning av bakåtproblemet följer en liknande metod till den som presenterats i avsnitt C.1 men det föreligger även några grundläggande skillnader som påverkar både tillämpning och formulering. Bland annat i lösningriktning, diskretiseringformulering och begynnelsevärde.

Efter att ha löst framåtproblemet med Newtons metod kan vi nu övergå till att lösa bakåtproblemet som formuleras på följande sätt

$$\begin{cases} \frac{d\lambda}{dt} = p(\lambda(t), \alpha(t)), \ t \in [0, T] \\ \lambda(T) = 0. \end{cases} \tag{C.8}$$

Lösningen av framåtproblemet används i högerledet av (C.8) i funktionen $p(\lambda(t), \alpha(t))$. Systemet av differentialekvationer måste lösas bakåt då högerledet är bestämt enligt (C.8). För att lösa ekvationssytemet diskretiseras funktionerna över tid, med tidssteget τ. Funktionen i ett tidigare tidssteg kan då formuleras enligt följande

$$p(\lambda^k, \alpha) = \frac{\lambda^{k+1} - \lambda^k}{\tau}, \tag{C.9}$$

där λ^{k+1} och λ^k är diskreta värden för tiderna k+1 och k. Eftersom systemet ska lösas bakåt kan λ^k lösas ut enligt följande två ekvationer

$$\lambda^k = \lambda^{k+1} - \tau p(\lambda^k, \alpha), \tag{C.10a}$$

$$\lambda^k + \tau p(\lambda^k, \alpha) - \lambda^{k+1} = 0. \tag{C.10b}$$

Genom att introducera $w := \lambda^k$ kan (C.10b) skrivas som:

$$Q(w) := w + \tau p(w, \alpha) - \lambda^{k+1} = 0. \tag{C.11}$$

För att lösa problemet formuleras Newtons metod enligt följande

$$w^{n+1} = w^n - (Q'(w^n)^{-1} \cdot Q(w^n), \tag{C.12a}$$

$$Q'(w^n) = I + \tau J(w^n). \tag{C.12b}$$

Här är I identitetsmatrisen och $J(w^n)$ är Jacobimatrisen, som ser ut på följande sätt,

$$J(w) = \begin{bmatrix} \frac{\partial p_1}{\partial \lambda_1} & \frac{\partial p_1}{\partial \lambda_2} & \frac{\partial p_1}{\partial \lambda_3} \\ \frac{\partial p_2}{\partial \lambda_1} & \frac{\partial p_2}{\partial \lambda_2} & \frac{\partial p_2}{\partial \lambda_3} \\ \frac{\partial p_3}{\partial \lambda_1} & \frac{\partial p_3}{\partial \lambda_2} & \frac{\partial p_3}{\partial \lambda_3} \end{bmatrix} (w). \tag{C.13}$$

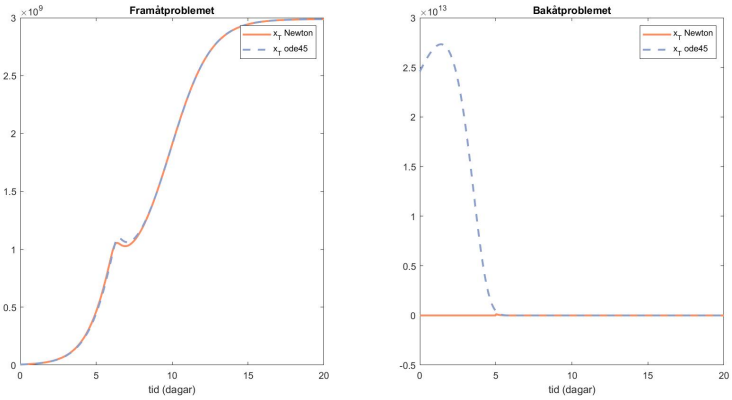
C.3 Analys av Newtons metod

Figurerna i detta avsnitt generaras med följande startvärden, $x_T = 5 \cdot 10^6$, $x_{M1} = 10^3$ och $x_{M2} = 10^3$. Startvärden för parametrarna är samlade i följande tabell.

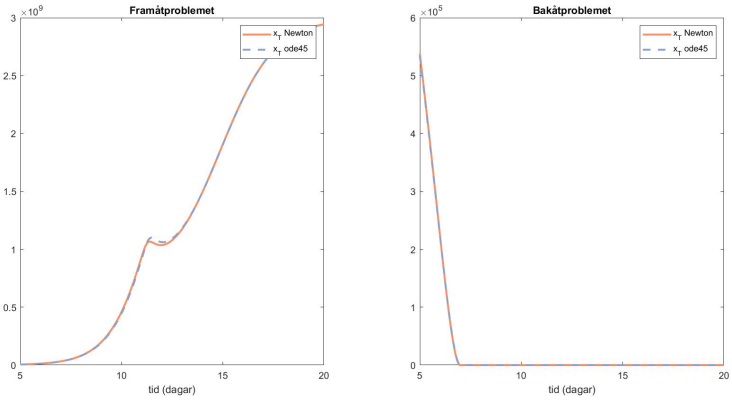
Tabell 4: Startvärden för parametrar vid analys av Newtons metod.

Parameter	d_{M1}	d_{M2}	a_{t1}	a_{t2}	k_{12}
Startvärde	10^{-9}	10^{-10}	10^{-8}	10^{-10}	$5 \cdot 10^{-10}$

Vid tillämpning av Newtons metod, beskriven i appendix C.1 och C.2, genererades inga bra resultat för lösningen av bakåtproblemet vilket kan ses i figur C.1. Lösningen med Newtons metod för framåtproblemet stämmer bra överens med den lösning som ges av ode45. För bakåtproblemet ser det ut som att Newtons metod ger en bra lösning efter cirka tiden 5 dagar, vilket kan ses mer tydligt i figur C.2.



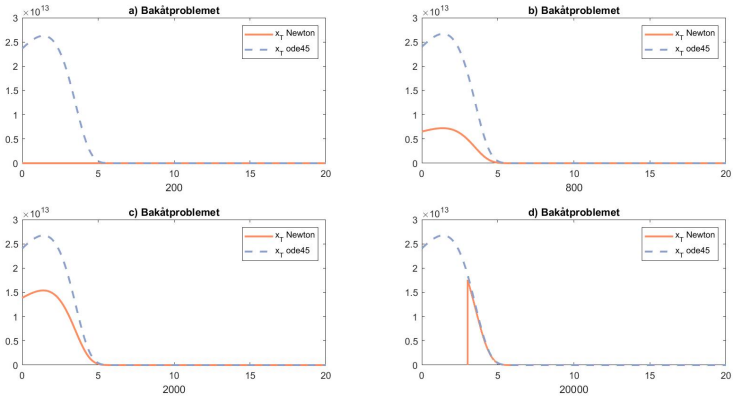
Figur C.1: Lösning av framåt och bakåtproblemet med Newtons metod och ode45 på tidsintervallet 0 till 20.



Figur C.2: Lösning av framåt och bakåtproblemet med Newtons metod och ode45 på tidsintervallet 5 till 20.

Några möjliga förklaringar till detta resultat på intervallet 0 till 5 är att det kan vara avrundningsfel eller att matrisen J beskriven i (C.13) har en determinant nära noll. Detta hade påverkat de begränsningar som ansätts i Newtonlösaren.

Lösning av bakåtproblemet med Newtons metod resulterar nollösning på intervallet 0-5 C.1. Eftersom ode45 i detta fall används som kontroll av lösningmetoden och genererar en godtycklig lösning framgår det att finns en problematik med lösningmetoden för bakåtproblemet. Då toleransen sänks från 10^{-5} till 10^{-3} i bakåtlösaren och det samtidigt testas olika antal punkter i tidspartitionen, alltså olika stora tidssteg, fås följande figur.



Figur C.3: Lösning av bakåtproblemet med Newtons metod med olika antal lösningspunkter.

Genom att successivt höja antalet punkter rör sig Newton mot ode45 lösningen vilket syns i figur C.3 b) och c). I d) höjs återigen antalet punkter, det får som effekt att lösningen istället skarpt vänder ner mot noll. Det beror troligtvis på att Newtons metod i detta fall inte konvergerar korrekt och att toleransen som ansatts i koden överskrids och lösaren söker sig mot nollan. Då lösningen för de fall med mindre tidssteg förbättras. Vidare felsökning för att hitta orsak till felen undersöktes inte utan istället användes MATLABs lösare, delvis då Newtons metod inte låg i fokus för projektet.

D Appendix 4 – Datautjämning

I Tikhonovfunktionalen beskrivs vektorn $\mathbf{g}^c(t)$ som en kontinuerlig approximation av givna datapunkter, $\mathbf{g}$. Den kontinuerliga approximationen beräknas genom polynomregression till de diskreta datapunkterna. Observera att denna behandling är nödvändig för att erhålla en teoretisk rigorös lösning. Utan databehandling hade det i Tikhonovfunktionalen uppstått ett uttryck som en differens mellan en kontinuerlig och en diskret funktion. Datautjämningen avser lösa detta problem så att konjugerade gradientmetoden kan implementeras korrekt.

Kontinuitetsproblemet löses genom extrapolering, där ett regressionsproblem formuleras som att hitta ett polynom av grad $j-1$:

$$p(t) = \sum_{i=0}^{j-1} c_i t^i, \tag{D.1}$$

där c_i betecknar polynomets koefficienter. Målet är att hitta koefficienterna $\mathbf{c} = (c_0, c_1, \ldots, c_{j-1})$ så att polynomet $p(t)$ bäst approximerar datan. Problemet kan då formuleras som

$$\min_{\mathbf{c}} \|V\mathbf{c} - \mathbf{y}\|_2^2. \tag{D.2}$$

I normalekvationen representerar V Vandermondematrisen, $\mathbf{c}$ är vektorn innehållande polynomets koefficienter, och $\mathbf{y}$ är en vektor innehållande de observerade datapunkterna. Vandermondematrisen kan generellt skrivas som

$$V = \begin{bmatrix} t_1^0 & t_1^1 & \ldots & t_1^{j-1} \\ t_2^0 & t_2^1 & \ldots & t_2^{j-1} \\ \vdots & \vdots & \ddots & \vdots \\ t_i^0 & t_i^1 & \ldots & t_i^{j-1} \end{bmatrix}. \tag{D.3}$$

Eller mer kompakt $\{V_{a,b}\}_{a=1,b=0}^{i,j-1} = \{t_a^{b-1}\}_{a=1,b=0}^{i,j-1}$. Vandermonde matrisen konstrueras med en given mängd datapunkter som utgör raderna och polynomets grad som kolumnerna i matrisen.

Normalekvationen löses vanligen som

$$V^T V\mathbf{c} = V^T\mathbf{y} \tag{D.4}$$

där V^T är transponatet av V.

LU-faktorisering är en teknik som används för att skriva om en matris som produkten av två triangulärmatriser, en nedre, L, och en övre, U. Choleskyfaktorisering är ett specialfall av LU-faktorisering där $U = L^T$. Denna faktorisering kräver att matrisen som ska faktoriseras är positivt definit, det vill säga att den är kvadratisk och alla dess egenvärden är större än 0 [19].

Regressionsproblemet, (D.4), och dess lösning inleds genom Choleskyfaktorisering, där V^TV skrivs om som $V^TV = LL^T$. Problemet kan då skrivas om som

$$LL^T\mathbf{c} = V^T\mathbf{y}. \tag{D.5}$$

Eftersom V^TV antas positivt definit för att kunna använda Choleskyfaktorisering medför det även att V^TV är inverterbar. Detta medför i sin tur att L och dess transponat är inverterbara. Problemet kan då skrivas om som

$$\mathbf{c} = L^{-1}((L^T)^{-1}V^T\mathbf{y}). \tag{D.6}$$

Genom att beräkna uttrycket (D.6) erhålls koefficienterna till regressionspolynomet. Choleskyfaktorisering är i detta sammanhang en användbar metod då den utnyttjar triangulärmatrisernas struktur vilket bidrar det till att öka beräkningseffektiviteten gentemot direkt lösning av problemet.

E Appendix 5 – Explicit beräkning av en parameter

Nedan följer resterande resultat av de metoder som applicerades för att försöka reducera felen i den explicita beräkningen av en parameter i avsnitt 6.1 och som inte fick plats i rapportens huvuddel.

E.1 Val av lösare samt förenkling av modell

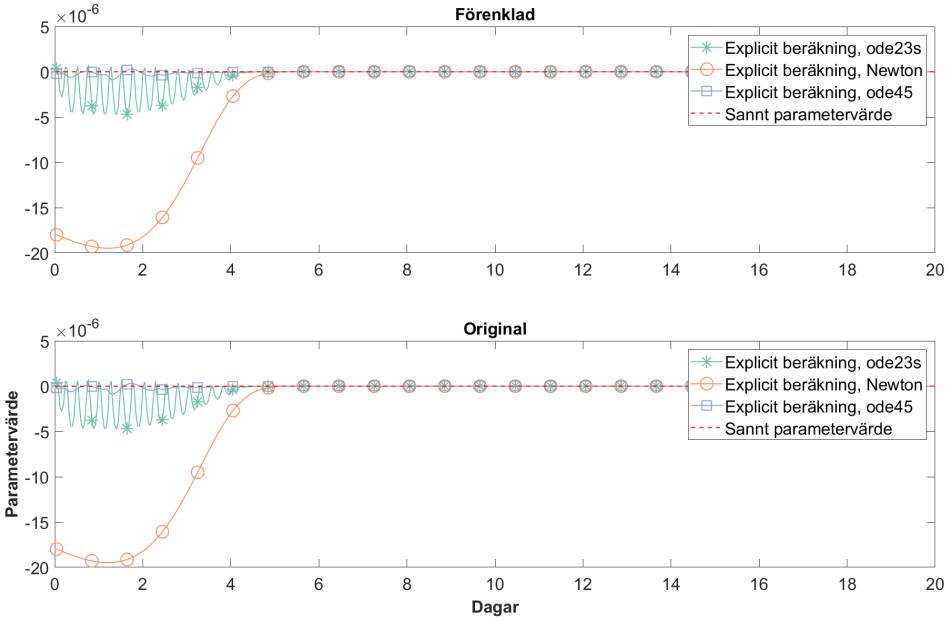
I avsnitt 6.1 analyseras två figurer vilka visade avvikelser i början av tidsintervallet. Liknande kan även konstateras för resterande parametrar. Flera metoder implementerades för att reducera dessa avvikelser. I projektet används MATLABs egna lösare samt Newtons metod för att lösa systemet av differentialekvationer. MATLAB har flera olika ode-lösare och vilken som passar bäst att använda beror bland annat på om funktionssystemet är styv eller inte.

I figur E.1, E.2, E.3, E.4 och E.5 visas grafer med lösningar till explicita beräkningar för $\boldsymbol{\alpha}$ beräknat med MATLABs egna lösare ode23s och ode45, samt Newtons metod, se appendix C. De startvärden för $\boldsymbol{\alpha}$ samt $\boldsymbol{x}$ som användes i beräkningarna visas i tabell 3, i avsnitt 6.1.

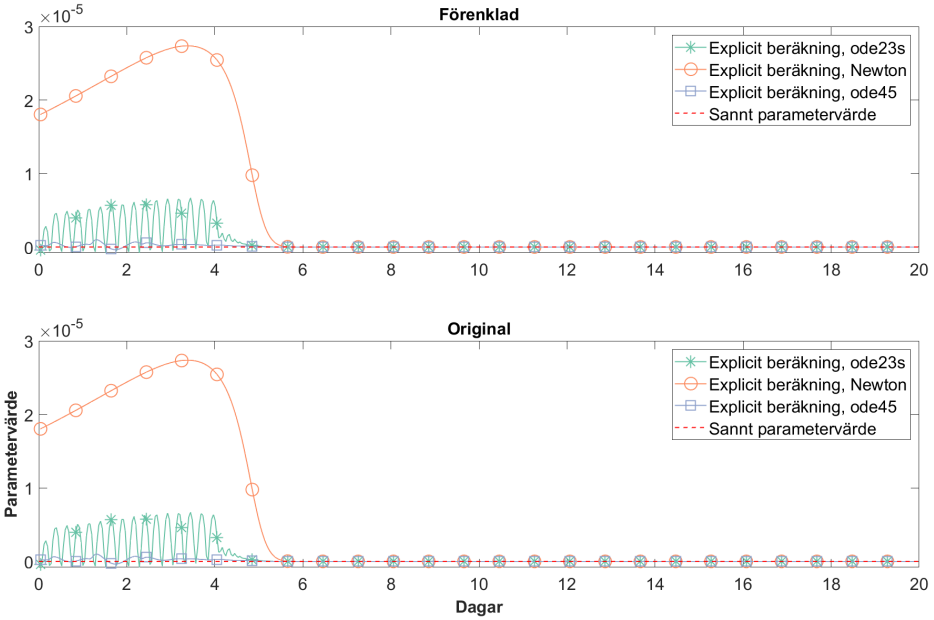
För alla grafer är resultaten av lösarna mellan dag 10 och 20 nästintill identiska, bortsett från figur E.3 där ode45 fluktuerar medans resterande lösare inte gör det. I början av tidsintervallet visar dock resultatet beräknat med Newtons metod relativt stora avvikelser och det beräknat med ode45 relativt små. Då alla grafer visar fluktuationer i början prioriterades dock huruvida lösaren presterade i den senare delen av tidsintervallet. På grund av ovanstående anledningar valdes ode23s som beräkningsalgoritm.

Den övre grafen i varje figur visar resultatet beräknat med en modifierad version av (5.1) och den nedre visar resultatet beräknat med originalekvationerna.

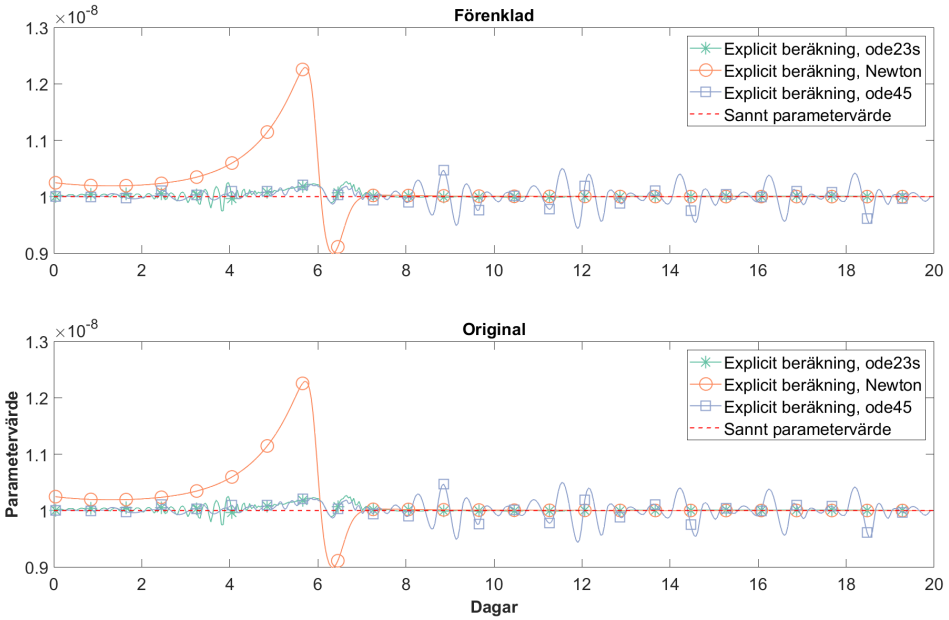
Generellt ser lösningarna beräknade med ode23s ut att variera minst kring det sanna värdet och ger därav den mest passande lösningen.



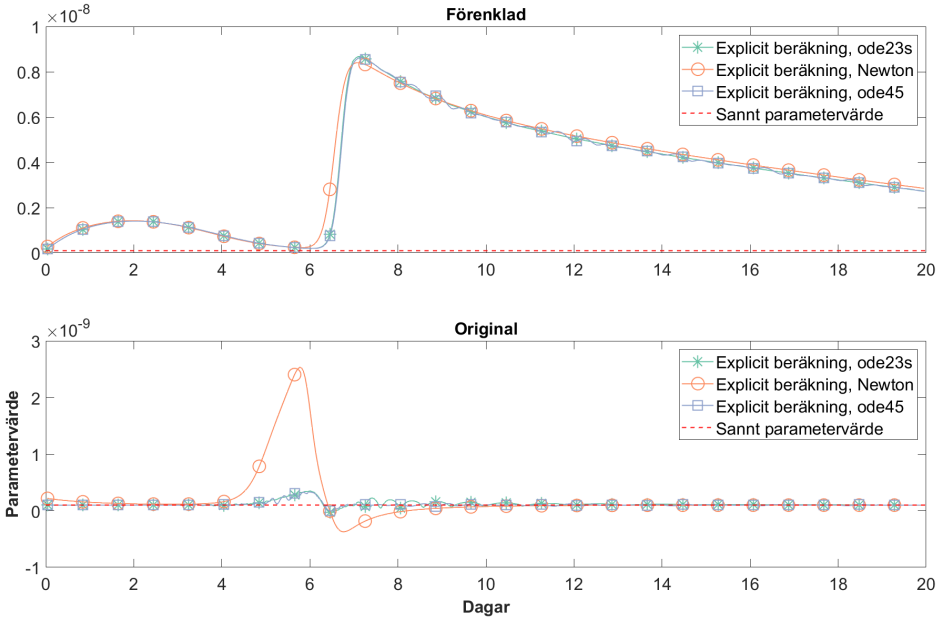
Figur E.1: Grafer för explicita beräkningar av d_{M1}. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). De olika färgerna representerar lösningen för olika ode-lösare.



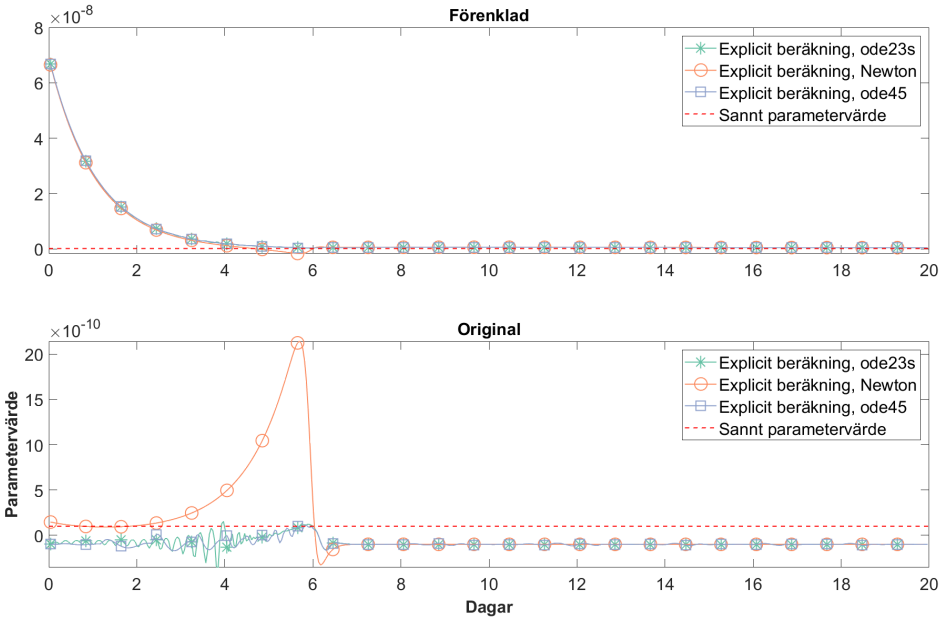
Figur E.2: Grafer för explicita beräkningar av d_{M2}. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). De olika färgerna representerar lösningen för olika ode-lösare.



Figur E.3: Grafer för explicita beräkningar av a_{t1}. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). De olika färgerna representerar lösningen för olika ode-lösare.



Figur E.4: Plot för explicita beräkningar av a_{t2}. Den övre plotten visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). De olika färgerna representerar lösningen för olika ode-lösare.

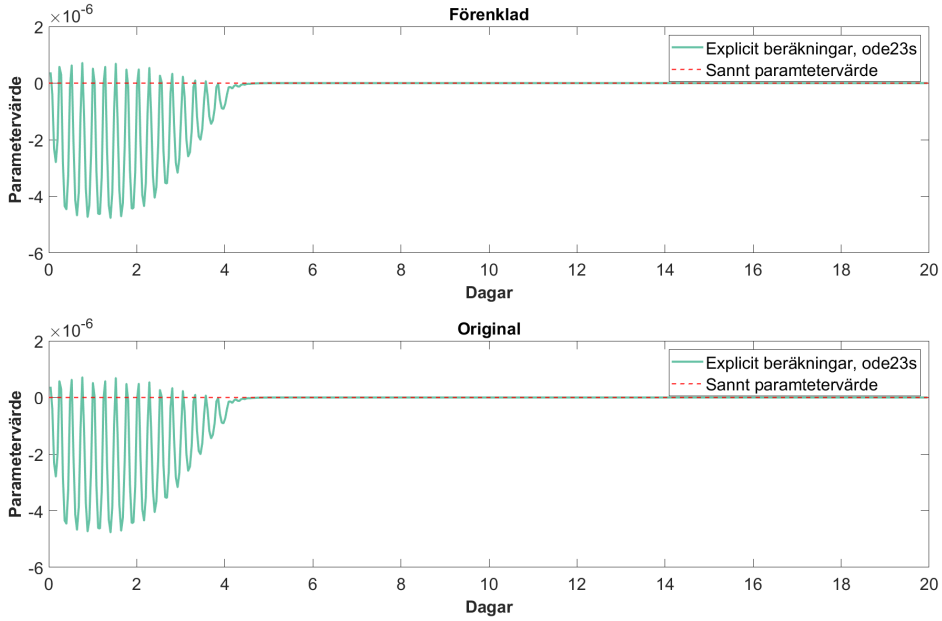


Figur E.5: Grafer för explicita beräkningar av k_{12}. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). De olika färgerna representerar lösningen för olika ode-lösare.

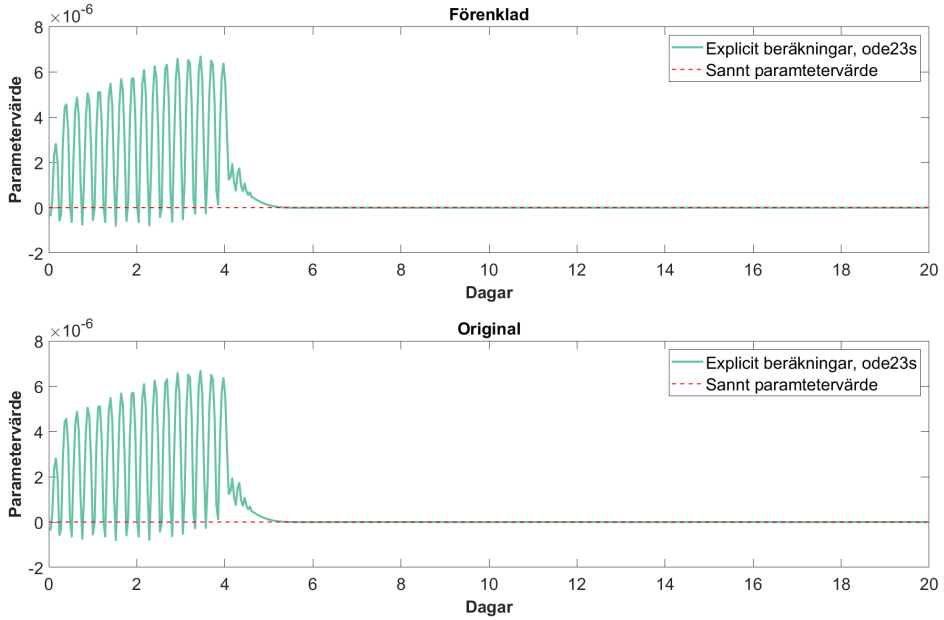
För parameter d_{M1}, d_{M2} och a_{t1} är resultatet av den modifierade versionen till synes identiskt med resultatet från originalekvationen. För parametern a_{t2} ger förenklingen en sämre approximation medan för parameter k_{12} resulterar beräkningen med den förenklade ekvationen i en stabilare graf med mindre variation.

E.1.1 Jämförelse mellan originalekvationer och förenklade ekvationer

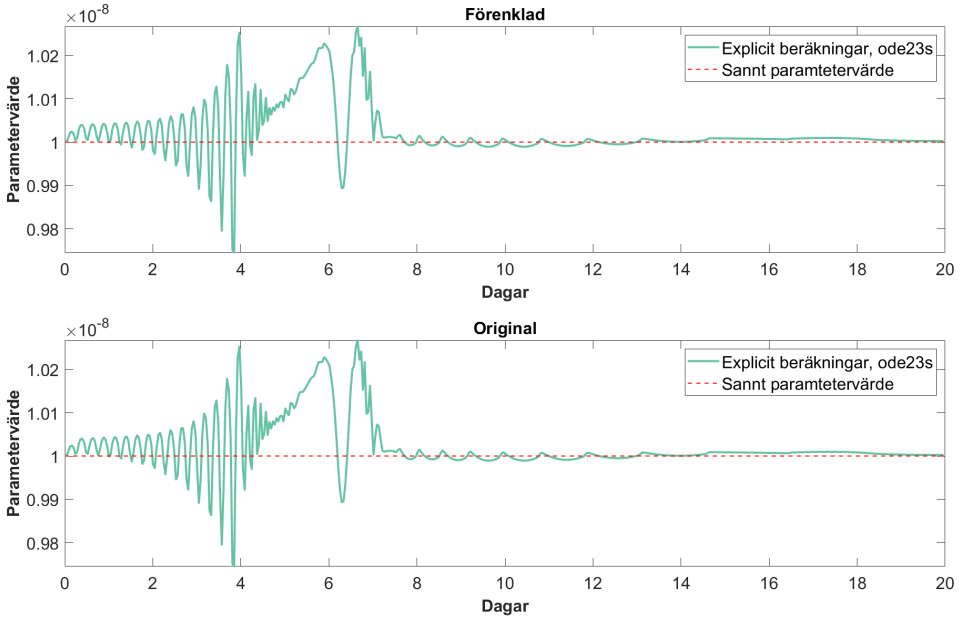
Figurerna E.6, E.7, E.8, E.9 och E.10 visas grafer med lösningar till den explicita beräkningen för $\boldsymbol{\alpha}$ beräknade med MATLABs egna lösare ode23s. Figur E.7 och E.10 är samma figurer som visas i avsnitt 6, det vill säga figur 6.2 respektive 6.3.



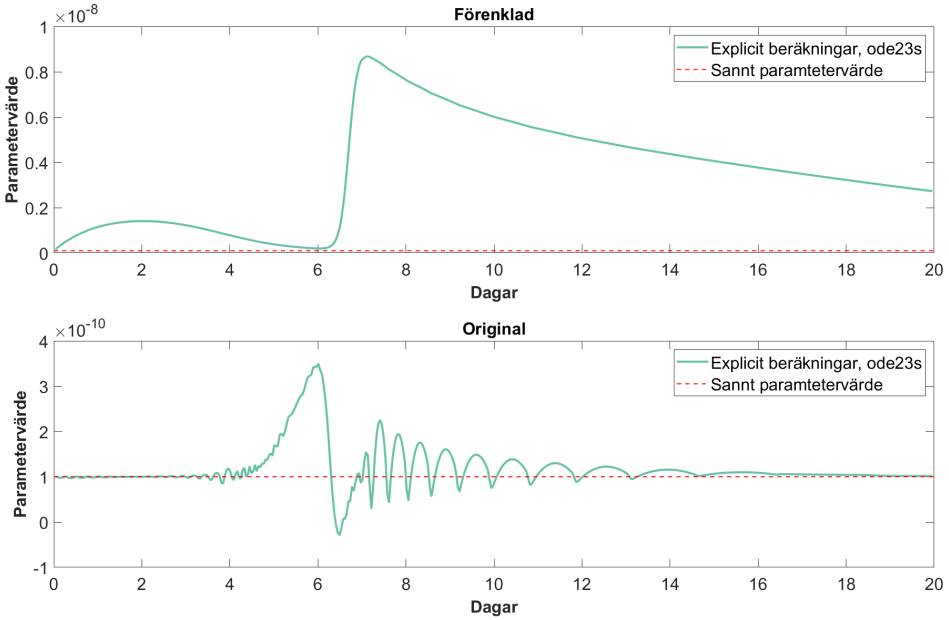
Figur E.6: Grafer för explicita beräkningar av d_{M1}. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $1,00 \cdot 10^{-9}$ för både den förenklade modellen och original modellen.



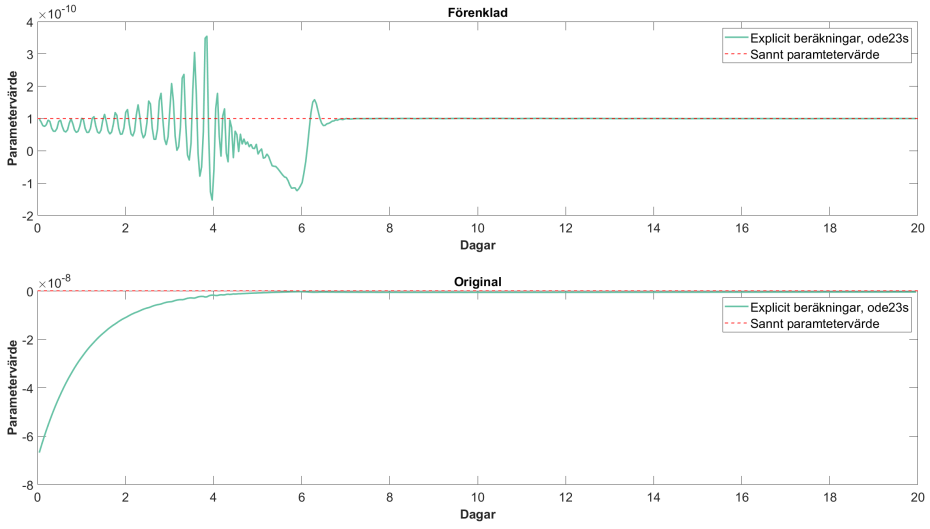
Figur E.7: Grafer för explicita beräkningar av d_{M2}. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $9,93 \cdot 10^{-11}$ för både den förenklade modellen och original modellen.



Figur E.8: Grafer för explicita beräkningar av a_{t1}. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $1,00 \cdot 10^{-8}$ för både den förenklade modellen och original modellen.



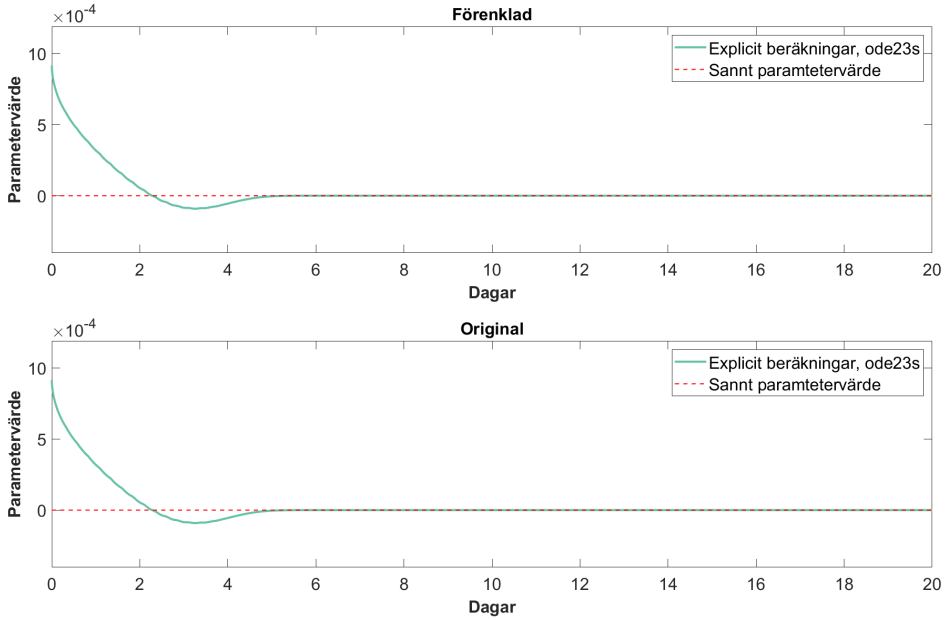
Figur E.9: Plot för explicita beräkningar av a_{t2}. Den övre plotten visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $4,15 \cdot 10^{-9}$ för den förenklade modellen och $1,09 \cdot 10^{-9}$ för original modellen.



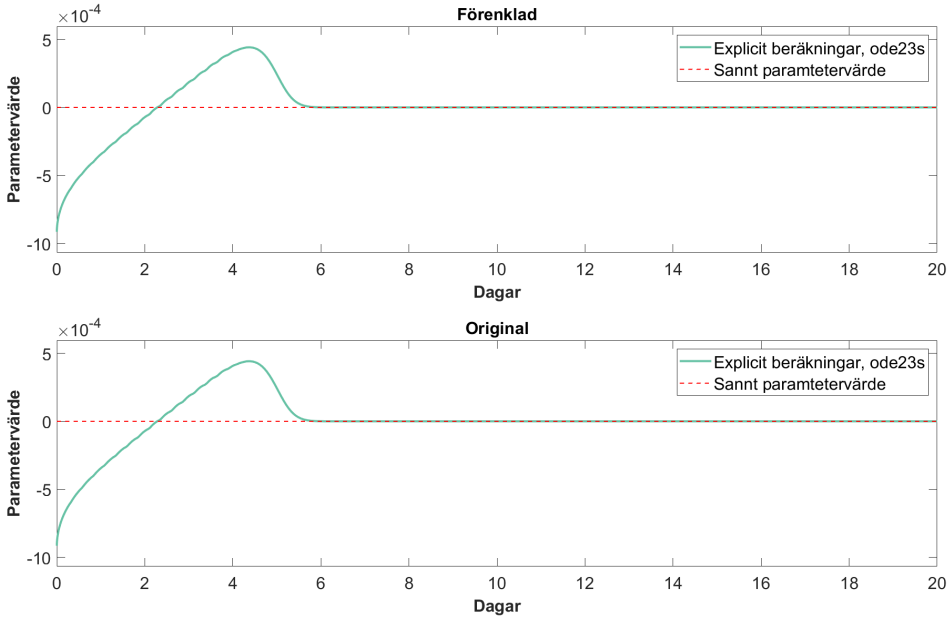
Figur E.10: Grafer för explicita beräkningar av k_{12}. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $9,99 \cdot 10^{-11}$ för den förenklade modellen och $-4,67 \cdot 10^{-10}$ för original modellen.

E.2 Tillämpning av Chebyshev noder

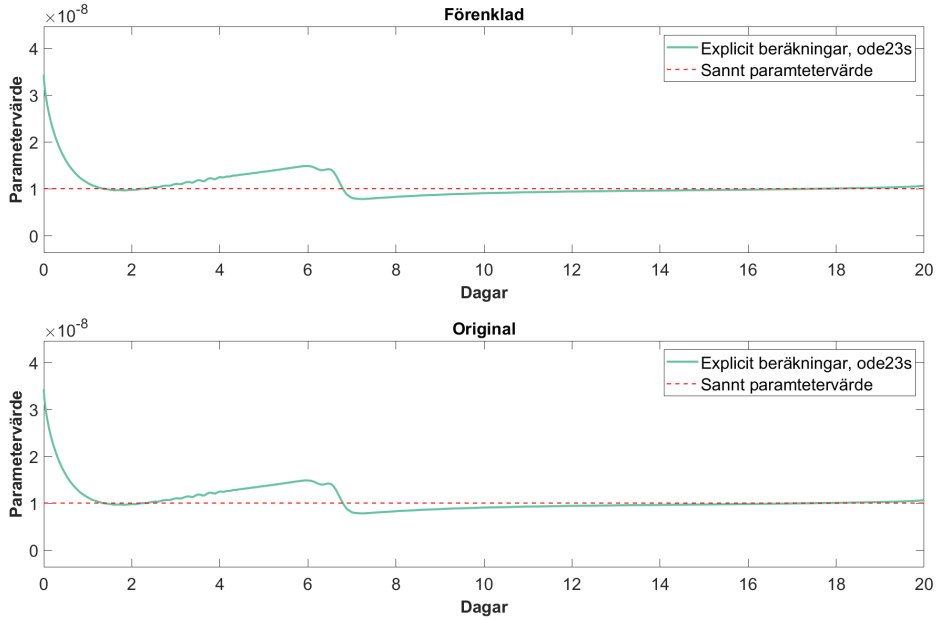
Resultatet av användning av Chebyshev noder enligt (5.5) visas i figur E.11, E.12, E.13, E.14 samt E.15. Startvärden för samtliga parametrar och densiteter är givna i tabell 3. Observera att figur E.15 är samma som figur 6.4.



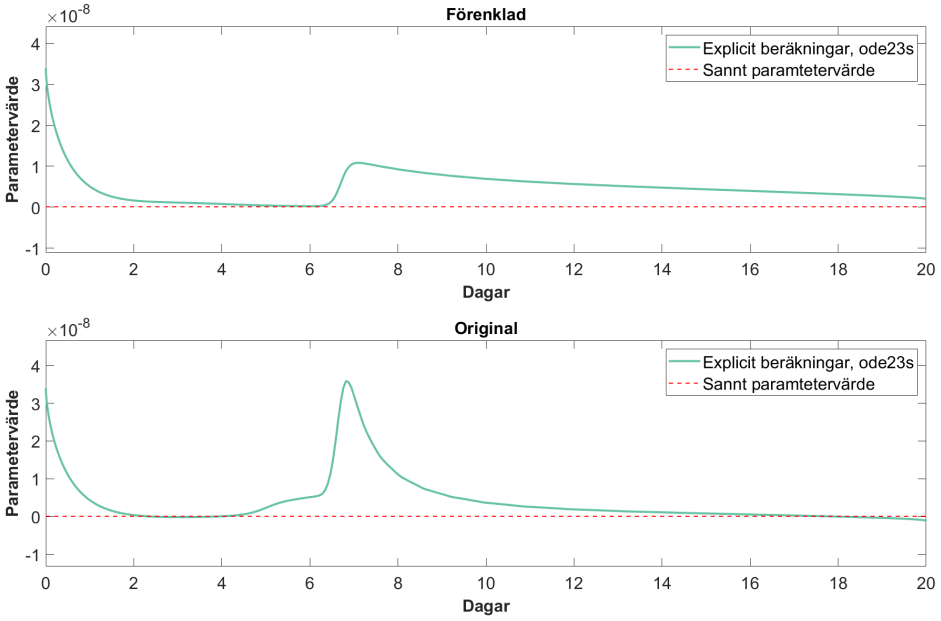
Figur E.11: Grafer för explicita beräkningar av d_{M1} utförda med Chebyshev noder. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $1,00 \cdot 10^{-9}$ för både den förenklade modellen och original modellen.



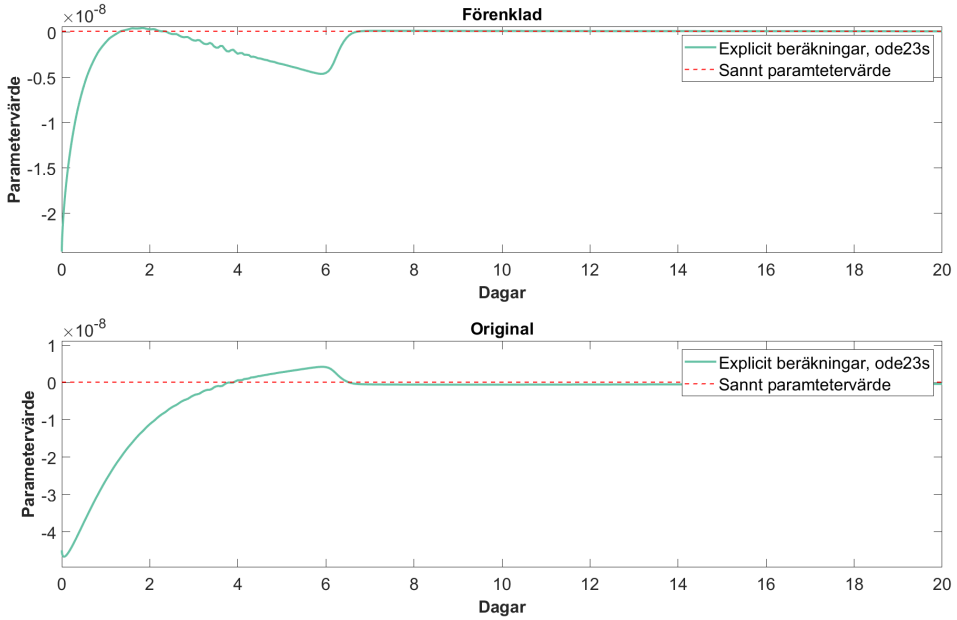
Figur E.12: Grafer för explicita beräkningar av d_{M2} utförda med Chebyshev noder. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $1,08 \cdot 10^{-10}$ för både den förenklade modellen och original modellen.



Figur E.13: Grafernför explicita beräkningar av a_{t1} utförda med Chebyshev noder. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $9,93 \cdot 10^{-9}$ för både den förenklade modellen och original modellen.



Figur E.14: Grafer för explicita beräkningar av a_{t2} utförda med Chebyshev noder. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $3,82 \cdot 10^{-9}$ för den förenklade modellen och $5,00 \cdot 10^{-10}$ för original modellen.

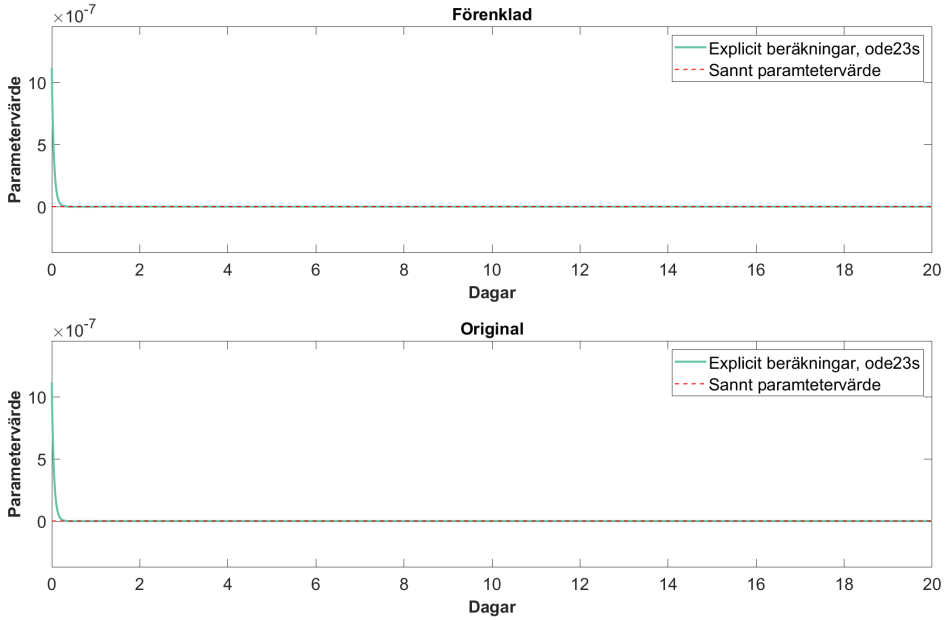


Figur E.15: Grafer för explicita beräkningar av k_{12} utförda med Chebyshev noder. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $1,03 \cdot 10^{-10}$ för den förenklade modellen och $-4,49 \cdot 10^{-10}$ för original modellen.

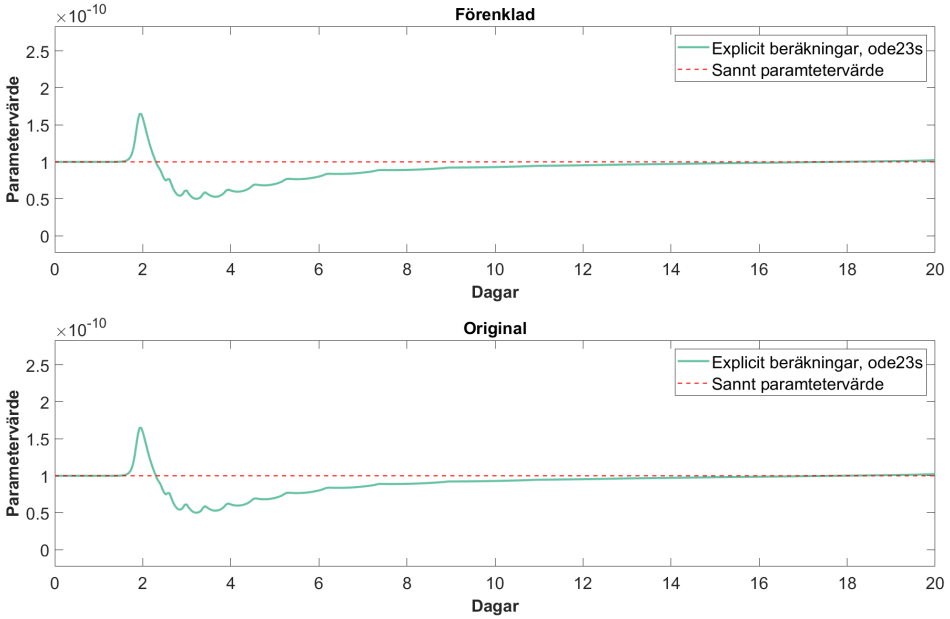
Införandet av Chebyshev noder gav jämnare grafer med mindre varians i jämförelse med de fall då beräkningspunkterna fördelats jämnt. En annan skillnad mellan dessa fallen är att resultaten för användande av olika lösare skiljer sig mindre vid användning av Chebyshev noder. Dock är det fortfarande en relativt lång period i början av tidsintervallet som de beräknade parametervärdena fluktuerar kring det sanna parametervärdet.

E.3 Chebyshev och steady state

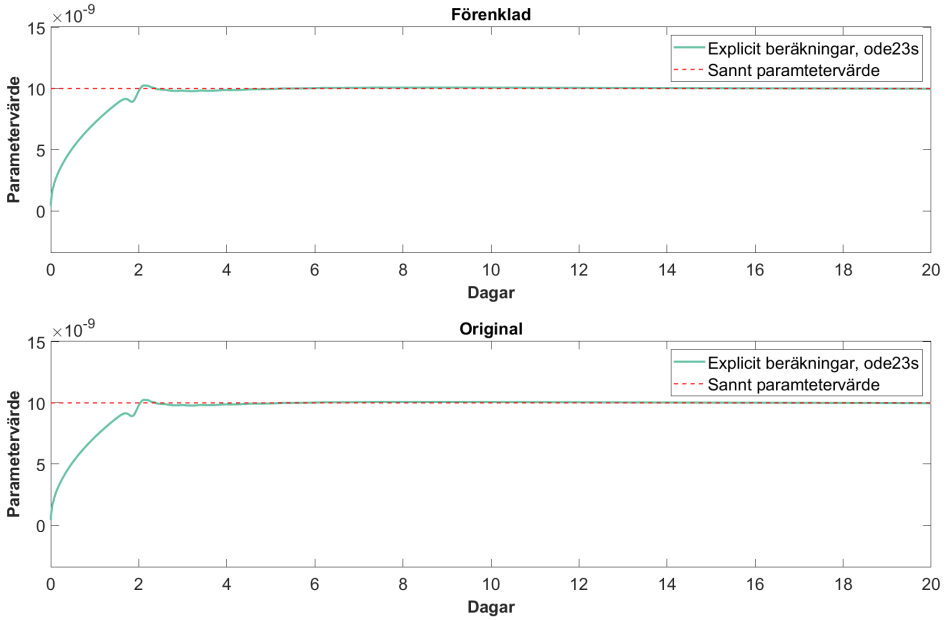
Figur E.16, E.17, E.18, E.19 samt E.20 visar resultatet av de explicita beräkningarna vid användning av Chebyshev noder, likt i E.2, fast där startvärdena på densiteterna $\boldsymbol{x}$ är satta nära ett steady state, med $x_T = 3,13 \cdot 10^9$, $x_{M1} = 10^{-5}$ och $x_{M2} = 4,03 \cdot 10^8$. Resterande parametervärden följer tabell 3. Observera att figur E.20 är samma som i figur 6.6.



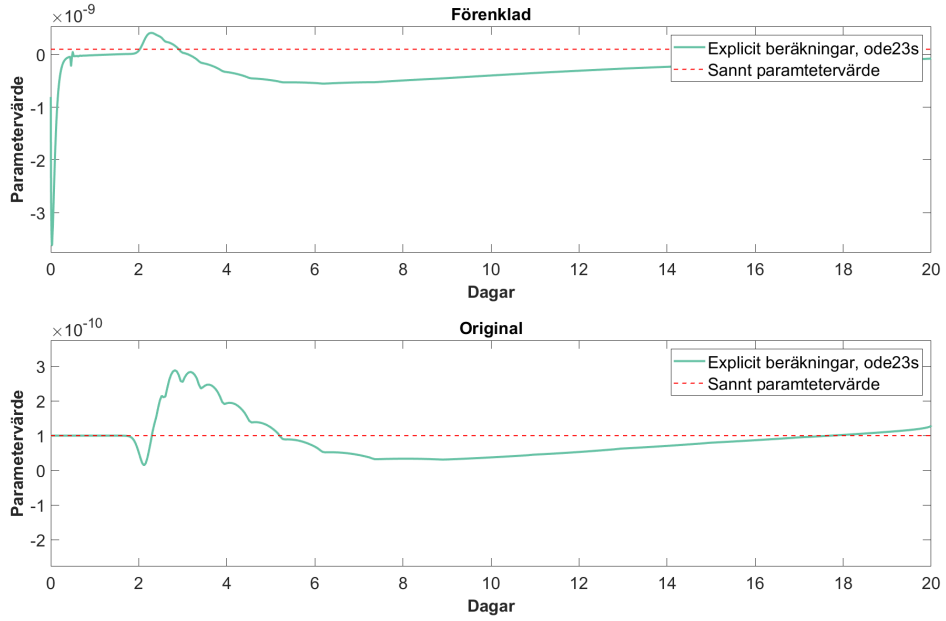
Figur E.16: Grafer för explicita beräkningar av d_{M1} utförda med Chebyshev noder samt nära steady state. Den övre grafen visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $1,00 \cdot 10^{-9}$ för både den förenklade modellen och original modellen.



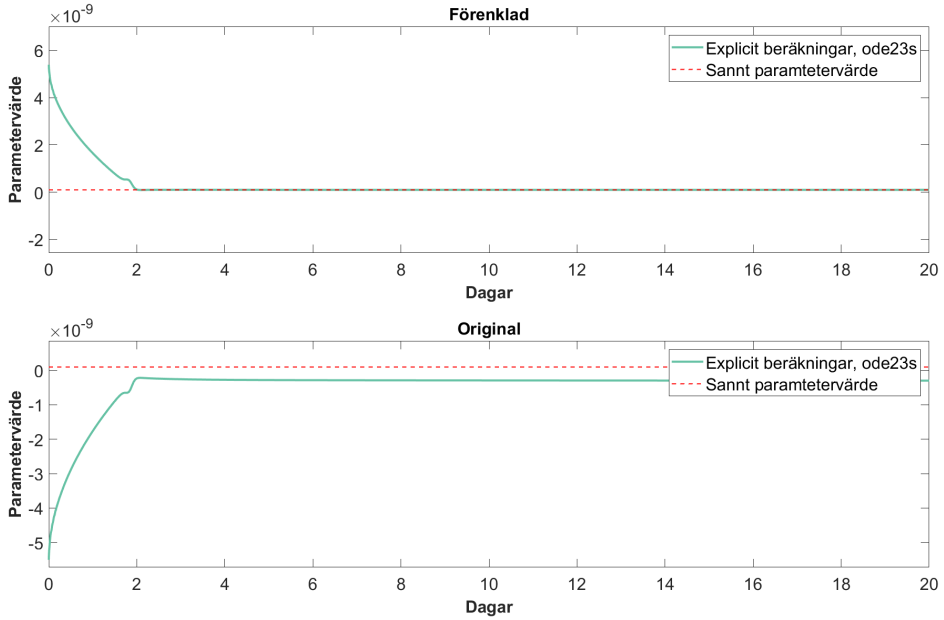
Figur E.17: Plot för explicita beräkningar av d_{M2} utförda med Chebyshev noder samt nära steady state. Den övre plotten visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $9,90 \cdot 10^{-11}$ för både den förenklade modellen och original modellen.



Figur E.18: Plot för explicita beräkningar av a_{t1} utförda med Chebyshev noder samt nära steady state. Den övre plotten visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $1,00 \cdot 10^{-8}$ för både den förenklade modellen och original modellen.



Figur E.19: Plot för explicita beräkningar av a_{t2} utförda med Chebyshev noder samt nära steady state. Den övre plotten visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $-1,77 \cdot 10^{-10}$ för den förenklade modellen och $9,02 \cdot 10^{-11}$ för original modellen.



Figur E.20: Plot för explicita beräkningar av k_{12} utförda med Chebyshev noder samt nära steady state. Den övre plotten visar resultatet vid användning av den förenklade ekvationen, se (5.4), och den nedre den ursprungliga ekvationen, se (5.1). Beräknat medelvärde mellan dag 10 och 20 är $9,98 \cdot 10^{-11}$ för den förenklade modellen och $-2,95 \cdot 10^{-10}$ för original modellen.

F Appendix 6 – Källkod

Det bör noteras att delar av den kod som används i arbete inte är vår egen. En stor del har specifikt utformats för att passa de ändamål och syften som nämns i rapporten. Viktigt att poängtera är dock att mycket av koden kommer från tidigare arbete, tillhörande vår handledare, Larisa Beilina. För en mer detaljerad förklaring av koden samt förtydling vilka delar som är skapade i projektet och vilka som är givna, se README.md filen.

Koden tillsammans med README.md är tillgänglig nedan samt via följande GitHub-länk: `https://github.com/reginag99/MVEX11-VT23-09.git`.

README.md

```
# MVEX11-VT23-09
Kod som användes i kandidatarbetet, kurs MVEX11-VT23-09. Vissa av filerna
är givna och enbart modifierade medans andra är skapade i projektet.
Se README för ytterligare instuktioner.

Nedan kommer en beskrivning för test-filerna, vilka är de som ska köras.
Resterande filer är funktionsfiler som används i testfilerna. I filerna
motsvarar alpha_unknown=1 d_m1, alpha_unknown=2 d_m2, alpha_unknown=3 a_t1,
alpha_unknown=4 a_t2 och alpha_unknown=5 k_12.

test_variation_colour.m bygger på given kod men delen %% Variations in
observations är skapat i projektet. Inskrivna max och min punkter för
respektive parameter ytterpunkterna på dess estimerade parameterintervall.

test_alpha_exp.m är de explicita beräkningarna av en parameter med den
ursprungliga modellen och inte den förensklade. Det går att välja om
resultatet ska presenteras i vanlig eller logaritmerad skala. Det går också
att välja vilken lösare som ska användas. Antingen ode23s, ode45 eller Newton.
Första delen av projektet är skapad i projektet medan den nedre delen,
skalningen, är tagen från given kod.

test3.m är given och visar lösningen av modellen med och utan tillagt
brus i figur 1. Figur 2 visar lösningen för de olika densiteterna. Koden är given
    och har inte skapats i projektet.

test2.m var redan given jämför lösningen av ode45 och Newton, bakåt och framåt.
    Koden är given och har inte skapats i projektet.

test2_Chebyshew.m jämför lösningen av ode45 och Newton, bakåt och framåt,
efter tillämpning av Chebyshew noder. Grunden är test2.m som sedan
modifierades i projektet.

test.m är given men modifierad då filen inte fungerade och genererade error.
Figur 1 visar lösningen med och utan  brus. Figur 2 visar framåtlösningen av ode45
och Newton. Figur 3 visar bakåtlösningen av ode45 och Newton.

mod_test_alpha_exp_delar.m jämför lösningen av den explicita beräkningen för
när delar av den förenklade modellen tagits bort. Detta gjordes för att
undersöka om avvikeler av modellen orsakar i synnerhet av en viss del av
ekvationerna. I fugur 2 visar genererade denisiteterna som också använts för de
    explicita
beräkningarna.  Förutom skaliningen som är tagen från givet script har all
kod i filen skapats i projektet.

mod_test_alpha_exp_Chebyshew.m genererar plottar för de explicita
beräkningarna efter tillämpning av Chebyshew noder. Plottarna visar resultatet
av olika lösare, skillnaden mellan förenklade verionen av modellen och orginal
modellen i vanlig samt logaritmisk skala. I fugur 2 visar genererade
denisiteterna som också använts för de explicita beräkningarna.Förutom
skaliningen som är tagen från givet script har all kod i filen skapats i projektet.

mod_test_alpha_exp_Chebyshew_plot.m är en anpassad verion för att användas
i rapporten av mod_test_alpha_exp_Chebyshew.m.

mod_test_alpha_exp.m genererar plottar för explicita beräkningen för
```

```
orginalmodellen samt förenklade modellen, löst med ode45,ode23s och Newton.
I fugur 2 visar genererade denisiteterna som också använts för de explicita
beräkningarna.Förutom skaliningen som är tagen från givet script har all
kod i filen skapats i projektet.

mod_test1_alpha_exp_ode23s.m är en anpassad version av mod_test1_alpha_exp.m
för att användas i rapporten.

Create_adjoint_figure.m gernererar figurerna för Newtons metod i rapporten.
Filen bygger på given kod som modifierats. Om denna ändras glöm inte att
ändra observationsintervallet så det matchar! För att generera samma plottar som i
    rapporten sätt:
obs_start = 2.1 för tidsintervall 0 till 20.
obs_start = 5 och intervallet är 5 till 20.

leastsquare.m är polynomregression vilket används för att skapa ett polynom
 från diskreta datapunkter. Koden är skapad i projektet.
```

test_variation_colour.m

```
%%  Startvärden
clc; close all; clear
%%Startvärden på x och tidsvariabler
m = 500; % Antal observationer
obs_start = 2.1; obs_end = 7; %Intervall för observationer
time_final = 20;

time_mesh = linspace(0,time_final,m);
% x_initial = [x_T(0); x_M1(0); x_M2(0)]
% Startvärden som verkar passa fig 1a
x_initial = [5*10^6; 10^3; 10^3];

%% Variations in observations
%Nedanstående max och min förvardera parameter är respektives
%parameterintervall, givet i rapporten.
alpha_max = [10^-7 10^-8 10^-6 10^-8 10^-7];
alpha_min = [10^-11 10^-12 10^-10 10^-12 10^-12];
alpha_mid = 10.^((log10(alpha_min)+log10(alpha_max))/2);

% Konstanter att ändra för att plotta andra parametrar och grafer
alpha = alpha_mid';  % Värden på parametrar som ej varieras
big_var = 2;         % Hur många stora linjer
small_var = 5;      % Hur många små linjer per stor linje
alpha_chosen = 5;    % Vilken parameter ska varieras

alpha_chosen_variance = [alpha_min(alpha_chosen) alpha_max(
    ↪ alpha_chosen)];
variation=logspace(log10(alpha_chosen_variance(1)),log10(
    ↪ alpha_chosen_variance(end)),big_var);
variation_step=variation(2)/variation(1);

figure("Name", "Variation av parameter  " + alpha_chosen)
subplot(3,1,1)
xlabel('Dagar','FontSize',12,'FontWeight','bold')
ylabel('X_T, [celler]','FontSize',12,'FontWeight','bold')
hold on
subplot(3,1,2)
xlabel('Dagar','FontSize',12,'FontWeight','bold')
```

```
ylabel('X_{M1}, [celler]','FontSize',12,'FontWeight','bold')
hold on
subplot(3,1,3)
xlabel('Dagar','FontSize',12,'FontWeight','bold')
ylabel('X_{M2}, [celler]','FontSize',12,'FontWeight','bold')
hold on



dark_green = [30,120,130]/255;
dark_orange = [112,81,28]/255;
dark_purple = [101,120,163]/255;

light_green = [122,214,185]/255;
light_orange = [232,131,78]/255;
light_purple = [151,190,203]/255;

greenGRADIENTlight = @(i,N) light_green + (dark_green-light_green)*((i-1)/(N-1));
greenGRADIENTdark  = @(i,N) dark_green - (dark_green-light_green)*((i-1)/(N-1));
orangeGRADIENTlight = @(i,N) light_orange + (dark_orange-light_orange)*((i-1)/(N-1));
orangeGRADIENTdark  = @(i,N) dark_orange - (dark_orange-light_orange)*((i-1)/(N-1));
purpGRADIENTlight = @(i,N) light_purple + (dark_purple-light_purple)*((i-1)/(N-1));
purpGRADIENTdark  = @(i,N) dark_purple - (dark_purple-light_purple)*((i-1)/(N-1));

N_small=small_var*big_var-1;
N_big=big_var;
alpha_read=[];

b=logspace(log10(alpha_chosen_variance(1)),log10(alpha_chosen_variance(end)),big_var*(small_var+1));

for i=1:length(b)
    alpha(alpha_chosen,:)=b(i);
    alpha=alpha_vec(alpha(1),alpha(2),alpha(3),alpha(4),alpha(5),time_mesh);
    F45 = ForwardODE45(alpha,time_mesh,x_initial);

    if i==1
        subplot(3,1,1)
        plot(time_mesh,F45(1,:),'color',orangeGRADIENTlight(i,length(b)),'linewidth',2);
        subplot(3,1,2)
        plot(time_mesh,F45(2,:),'color',greenGRADIENTlight(i,length(b)),'linewidth',2);
        subplot(3,1,3)
        plot(time_mesh,F45(3,:),'color',purpGRADIENTlight(i,length(b)),'linewidth',2);

    elseif i==length(b)

```

```
        subplot(3,1,1)
        plot(time_mesh,F45(1,:),'color',orangeGRADIENTlight(i,
    ↪ length(b)),'linewidth',2);
        subplot(3,1,2)
        plot(time_mesh,F45(2,:),'color',greenGRADIENTlight(i,length
    ↪ (b)),'linewidth',2);
        subplot(3,1,3)
        plot(time_mesh,F45(3,:),'color',purpGRADIENTlight(i,length(
    ↪ b)),'linewidth',2);
        F=F45

     else
     orangeGRADIENTlight(i,N_small)
     subplot(3,1,1)
     plot(time_mesh,F45(1,:),'color',orangeGRADIENTlight(i,length(b)
    ↪ ),'linestyle','--','linewidth',1);
     subplot(3,1,2)
     plot(time_mesh,F45(2,:),'color',greenGRADIENTlight(i,length(b))
    ↪ ,'linestyle','--','linewidth',1);
     subplot(3,1,3)
     plot(time_mesh,F45(3,:),'color',purpGRADIENTlight(i,length(b)),
    ↪ 'linestyle','--','linewidth',1);


     end

end


%
% for i_big_var=1:big_var
%     alpha(alpha_chosen,:)=variation(i_big_var);
%     alpha_read=[alpha_read; alpha(alpha_chosen,1)];
%     alpha=alpha_vec(alpha(1),alpha(2),alpha(3),alpha(4),alpha(5),
    ↪ time_mesh);
%     F45 = ForwardODE45(alpha,time_mesh,x_initial);
%
%     if i_big_var == 1
%         for i_small_var=1:small_var
%             line_width = 1;
%             alpha_plus_var = alpha;
%             (variation_step/2*i_small_var/small_var)
%             alpha_plus_var(alpha_chosen,:) = alpha_plus_var(
    ↪ alpha_chosen,:)*(variation_step/2*i_small_var/small_var);
%             alpha_read=[alpha_read; alpha_plus_var(alpha_chosen
    ↪ ,1)];
%             small_plus_F45 = ForwardODE45(alpha_plus_var,
    ↪ time_mesh,x_initial);
%
%             subplot(3,1,1)
%             plot(time_mesh,small_plus_F45(1,:),'color',
    ↪ orangeGRADIENTlight(i_small_var,N_small),'linestyle','--','
    ↪ linewidth',line_width);
%             subplot(3,1,2)
%             plot(time_mesh,small_plus_F45(2,:),'color',
    ↪ greenGRADIENTlight(i_small_var,N_small),'linestyle','--','
```

```
    ↪ linewidth',line_width);
%             subplot(3,1,3)
%             plot(time_mesh,small_plus_F45(3,:),'color',
    ↪ purpGRADIENTlight(i_small_var,N_small),'linestyle','--','
    ↪ linewidth',line_width);
%             a = alpha(:,1);
%         end
%     elseif i_big_var == length(variation)
%         for i_small_var=1:small_var-1
%             line_width = 1;
%             alpha_minus_var=alpha;
%             alpha_minus_var(alpha_chosen,:) = alpha_minus_var(
    ↪ alpha_chosen,:)*(0.5 + (i_small_var-1)/small_var/2);
%             alpha_read=[alpha_read; alpha_minus_var(alpha_chosen
    ↪ ,1)];
%             small_minus_F45 = ForwardODE45(alpha_minus_var,
    ↪ time_mesh,x_initial);
%
%             subplot(3,1,1)
%             plot(time_mesh,small_minus_F45(1,:),'color',
    ↪ orangeGRADIENTdark(small_var-i_small_var,N_small),'linestyle
    ↪ ','--','linewidth',line_width);
%             subplot(3,1,2)
%             plot(time_mesh,small_minus_F45(2,:),'color',
    ↪ greenGRADIENTdark(small_var-i_small_var,N_small),'linestyle
    ↪ ','--','linewidth',line_width);
%             subplot(3,1,3)
%             plot(time_mesh,small_minus_F45(3,:),'color',
    ↪ purpGRADIENTdark(small_var-i_small_var,N_small),'linestyle
    ↪ ','--','linewidth',line_width);
%         end
%     else
%         for i_small_var=1:small_var
%             line_width = 1;
%
%             alpha_plus_var = alpha;
%             alpha_plus_var(alpha_chosen,:) = alpha_plus_var(
    ↪ alpha_chosen,:)*(variation_step/2*i_small_var/small_var);
%             alpha_read=[alpha_read; alpha_plus_var(alpha_chosen
    ↪ ,1)];
%             small_plus_F45 = ForwardODE45(alpha_plus_var,
    ↪ time_mesh,x_initial);
%
%             subplot(3,1,1)
%             plot(time_mesh,small_plus_F45(1,:),'color',
    ↪ orangeGRADIENTlight(i_small_var,N_small),'linestyle','--','
    ↪ linewidth',line_width);
%             subplot(3,1,2)
%             plot(time_mesh,small_plus_F45(2,:),'color',
    ↪ greenGRADIENTlight(i_small_var,N_small),'linestyle','--','
    ↪ linewidth',line_width);
%             subplot(3,1,3)
%             plot(time_mesh,small_plus_F45(3,:),'color',
    ↪ purpGRADIENTlight(i_small_var,N_small),'linestyle','--','
    ↪ linewidth',line_width);
%         end
```

```
160 %          for i_small_var=1:small_var-1
161 %              line_width = 1;
162 %
163 %              alpha_minus_var=alpha;
164 %              alpha_minus_var(alpha_chosen,:) = alpha_minus_var(
    ↪ alpha_chosen,:)*(0.5 + (i_small_var-1)/small_var/2);
165 %              alpha_read=[alpha_read; alpha_minus_var(alpha_chosen
    ↪ ,1)];
166 %              small_minus_F45 = ForwardODE45(alpha_minus_var,
    ↪ time_mesh,x_initial);
167 %
168 %              subplot(3,1,1)
169 %              plot(time_mesh,small_minus_F45(1,:),'color',
    ↪ orangeGRADIENTdark(small_var-i_small_var,N_small),'linestyle
    ↪ ','--','linewidth',line_width);
170 %              subplot(3,1,2)
171 %              plot(time_mesh,small_minus_F45(2,:),'color',
    ↪ greenGRADIENTdark(small_var-i_small_var,N_small),'linestyle
    ↪ ','--','linewidth',line_width);
172 %              subplot(3,1,3)
173 %              plot(time_mesh,small_minus_F45(3,:),'color',
    ↪ purpGRADIENTdark(small_var-i_small_var,N_small),'linestyle
    ↪ ','--','linewidth',line_width);
174 %
175 %          end
176 %      end
177 %      subplot(3,1,1)
178 %      plot(time_mesh,F45(1,:),'color',orangeGRADIENTlight(i_big_var
    ↪ ,N_big),'linewidth',2);
179 %      subplot(3,1,2)
180 %      plot(time_mesh,F45(2,:),'color',greenGRADIENTlight(i_big_var,
    ↪ N_big),'linewidth',2)
181 %      subplot(3,1,3)
182 %      plot(time_mesh,F45(3,:),'color',purpGRADIENTlight(i_big_var,
    ↪ N_big),'linewidth',2)
183 % end
184 alpha_read=sort(alpha_read);
185
186 %% Inner functions
187
188 function alpha = alpha_vec(dm1,dm2,at1,at2,k12,time_mesh)
189 scaling_factor_dm1 = dm1;
190 scaling_factor_dm2 = dm2;
191 scaling_factor_at1 = at1;
192 scaling_factor_at2 = at2;
193 scaling_factor_k12 = k12;
194
195 function_flag = 0; % constant
196
197 exact_dm1 = ExactParameter(scaling_factor_dm1,function_flag,
    ↪ time_mesh); %Exact profile for dm1 to produce data.
198 exact_dm2 = ExactParameter(scaling_factor_dm2,function_flag,
    ↪ time_mesh); %Exact profile for dm2 to produce data.
199 exact_at1 = ExactParameter(scaling_factor_at1,function_flag,
    ↪ time_mesh); %Exact profile for at1 to produce data.
200 exact_at2 = ExactParameter(scaling_factor_at2,function_flag,
```

```
    ↪ time_mesh); %Exact profile for at2 to produce data.
exact_k12 = ExactParameter(scaling_factor_k12,function_flag,
    ↪ time_mesh); %Exact profile for k12 to produce data.

alpha = [exact_dm1; exact_dm2; exact_at1; exact_at2; exact_k12];

end
```

adj2func.m

```
function lambda = adj2func(t,alpha,x,lambda)
    % alpha = (d_m1, d_m2, a_t1, a_t2, k_12)
    % z = 1
     r = 0.93;
     beta_T = 3*10^9;
     d_m1 = alpha(1);
     d_m2 = alpha(2);
     a_t1 = alpha(3);
     a_t2 = alpha(4);
     beta_M = 9*10^8;
     sigma_m1 = 0.173;
     sigma_m2 = 0.173;
     k_12 = alpha(5);

     x_T = x(1); x_M1 = x(2); x_M2 = x(3);

     lambda1 = -lambda(1)*r*(1-2*x_T/beta_T) + lambda(1)*d_m1*x_M1
    ↪ - lambda(1)*d_m2*x_M2...
               -lambda(2)*a_t1*x_M1*(1-(x_M1+x_M2)/beta_M) + lambda
    ↪ (2)*k_12*x_M1...
               -lambda(3)*a_t2*x_M2*(1-(x_M1+x_M2)/beta_M) - lambda
    ↪ (3)*k_12*x_M1;
     lambda2 = -lambda(2)*a_t1*x_T*(1-(2*x_M1+x_M2)/beta_M) +
    ↪ lambda(2)*sigma_m1...
               +lambda(3)*a_t2*x_T*x_M2/beta_M - lambda(3)*k_12*x_T
    ↪  + lambda(1)*d_m1*x_T...
               +lambda(2)*k_12*x_T;
     lambda3 = lambda(3)*sigma_m2 - lambda(3)*a_t2*x_T*(1-(x_M1+2*
    ↪ x_M2)/beta_M)...
               -lambda(1)*d_m2*x_T + lambda(2)*a_t1*x_T*x_M1/beta_M
    ↪ ;
     lambda = [lambda1; lambda2; lambda3];

end
```

adjfunc.m

```
function lambda = adjfunc(t,alpha,x,lambda,g)
    % alpha = (d_m1, d_m2, a_t1, a_t2, k_12)
    % z = 1
     r = 0.93;
     beta_T = 3*10^9;
     d_m1 = alpha(1);
     d_m2 = alpha(2);
```

```
    a_t1 = alpha(3);
    a_t2 = alpha(4);
    beta_M = 9*10^8;
    sigma_m1 = 0.173;
    sigma_m2 = 0.173;
    k_12 = alpha(5);

    x_T = x(1); x_M1 = x(2); x_M2 = x(3);

    lambda1 = -lambda(1)*r*(1-2*x_T/beta_T) + lambda(1)*d_m1*x_M1 - lambda(1)*d_m2*x_M2...
              -lambda(2)*a_t1*x_M1*(1-(x_M1+x_M2)/beta_M) + lambda(2)*k_12*x_M1...
              -lambda(3)*a_t2*x_M2*(1-(x_M1+x_M2)/beta_M) - lambda(3)*k_12*x_M1...
              +(x_T - g(1));
    lambda2 = -lambda(2)*a_t1*x_T*(1-(2*x_M1+x_M2)/beta_M) + lambda(2)*sigma_m1...
              +lambda(3)*a_t2*x_T*x_M2/beta_M - lambda(3)*k_12*x_T + lambda(1)*d_m1*x_T...
              +lambda(2)*k_12*x_T + (x_M1-g(2));
    lambda3 = lambda(3)*sigma_m2 - lambda(3)*a_t2*x_T*(1-(x_M1+2*x_M2)/beta_M)...
              -lambda(1)*d_m2*x_T + lambda(2)*a_t1*x_T*x_M1/beta_M ...
              +(x_M2-g(3));
    lambda = [lambda1; lambda2; lambda3];

end
```

AdjfuncJacobi.m

```
function lambda = AdjfuncJacobi(x,alpha)
    % alpha = (d_m1, d_m2, a_t1, a_t2, k_12)
     r = 0.93;
     beta_T = 3*10^9;
     d_m1 = alpha(1);
     d_m2 = alpha(2);
     a_t1 = alpha(3);
     a_t2 = alpha(4);
     beta_M = 9*10^8;
     sigma_m1 = 0.173;
     sigma_m2 = 0.173;
     k_12 = alpha(5);

     x_T = x(1); x_M1 = x(2); x_M2 = x(3);
     lambda = zeros(3,3);

    lambda(1,1) = -r*(1-2*x_T/beta_T) + d_m1*x_M1 - d_m2*x_M2;
    lambda(1,2) = -a_t1*x_M1*(1-(x_M1+x_M2)/beta_M) + k_12*x_M1;
    lambda(1,3) = -a_t2*x_M2*(1-(x_M1+x_M2)/beta_M) - k_12*x_M1;
    lambda(2,1) = d_m1*x_T;
    lambda(2,2) = -a_t1*x_T*(1-(2*x_M1+x_M2)/beta_M) + sigma_m1 + k_12*x_T;
    lambda(2,3) = a_t2*x_T*x_M2/beta_M - k_12*x_T;
```

```
    lambda(3,1) = -d_m2*x_T;
    lambda(3,2) = a_t1*x_T*x_M1/beta_M;
    lambda(3,3) = sigma_m2 - a_t2*x_T*(1-(x_M1+2*x_M2)/beta_M);

end
```

AdjointNewton.m

```
%Computes the adjoint solution of the model problem.
function [lambda] = AdjointNewton(alpha, x, g, time_mesh, obs_start
    ↪ , obs_end)

    final_time = time_mesh(end); % here we choose the final time
    nodes = length(time_mesh);

    t=final_time;  % final  time in adjoint solver, the same final
    ↪ time is in the forward problem
    MaxIter = 1000;  % maximal number of iterations in Newton's
    ↪ method

    % values for lambda(T)= 0 at the final time
    lambda = zeros(length(x(:,1)),nodes);
    w = lambda(:,end);

    dt = time_mesh(2:end)-time_mesh(1:end-1);   %Time step

    %i = nodes + 1;
    for i = nodes:-1:2
        adjtol=1;adjiter=0;
        while adjtol>10^(-5) && adjiter < MaxIter   %Newton
    ↪ iterations
            if t >= obs_start & t <= obs_end % z = 1
                adjF = w - lambda(:,i) + dt(i-1)*adjfunc(t,alpha(:,
    ↪ i),x(:,i),...
                                    lambda(:,i),g(:,i));
            else % z = 0
                adjF = w - lambda(:,i) + dt(i-1)*adj2func(t,alpha(:,
    ↪ i),x(:,i),...
                                    lambda(:,i));
            end
            adjJ = eye(length(lambda(:,end))) + dt(i-1)*
    ↪ AdjfuncJacobi(x(:,i),alpha(:,i));
            dw = -adjJ\adjF;
            w=w+dw;                 %The Newton iteration
            adjiter = adjiter +1;
            adjtol = norm(dw,inf);
        end
        if adjiter==MaxIter         %If the Newton meth. does not
    ↪ converge
            disp('No convergence in the Newton method for adjoint
    ↪ problem')
            break
        end

        lambda(:,i-1) = w;
```

```
        t=t-dt(i-1);
    end

end
```

AdjointODE45.m

```
function lambda = AdjointODE45(alpha, x, g, time_mesh,obs_start,
    ↪ obs_end)
    [time,lambda] = ode45(@(t,lambda) ...
                Adjfunc(t,lambda,x,g,alpha,time_mesh,obs_start,
    ↪ obs_end),...
                flip(time_mesh),[0;0;0]);
    lambda = flip(lambda)';

    function func = Adjfunc(t,lambda,x,g,alpha,time_mesh,obs_start,
    ↪ obs_end)
        alphaPol =@(t) [];
        for i = 1:size(alpha,1)
           Pol_i =@(t) interp1(time_mesh,alpha(i,:),t);
           alphaPol =@(t) [alphaPol(t); Pol_i(t)];
        end
        alphat = alphaPol(t);

        xPol =@(t) [];
        for i = 1:size(x,1)
           Pol_i =@(t) interp1(time_mesh,x(i,:),t);
           xPol =@(t) [xPol(t); Pol_i(t)];
        end
        xt = xPol(t);

        gPol =@(t) [];
        for j = 1:size(g,1)
           Pol_j =@(t) interp1(time_mesh,g(j,:),t);
           gPol =@(t) [gPol(t); Pol_j(t)];
        end

        if t >= obs_start & t <= obs_end % z = 1
               gt = gPol(t);
               func = adjfunc(t,alphat,xt,...
                                   lambda,gt);
           else % z = 0
               func = adj2func(t,alphat,xt,...
                                   lambda);
           end
    end


end
```

AnalyticParameter.m

```
function [dm1_guess,g_obs] = AnalyticParameter(alpha, time_mesh,
    ↪ obs_start,obs_end,noise_flag,noise_level,x_initial)
```

```
2
3       %final_time = time_mesh(end);
4       nodes = length(time_mesh);
5       time_step = time_mesh(2:end) - time_mesh(1:end-1);
        ↪  %Time step for discrete derivatives.
6       g_prim = zeros(3,nodes);
7
8       %Simulate the true values of the model problem, with and
        ↪ without noise.
9       [g,g_brus,g_add] =  ExactNewton(alpha,time_mesh,noise_level,
        ↪ x_initial);
10      if noise_flag == 1
11          g_obs = g_brus;
12      elseif noise_flag == 2
13          g_obs = g_add;
14      else
15          g_obs = g;
16      end
17
18      %Compute derivatives of g.
19
20      g_prim(:,2:end) = (g_obs(:,2:end)-g_obs(:,1:end-1))./time_step;
21
22      %Compute analytic eta inside observation interval.
23      r = 0.93;
24      beta_T = 3*10^9;
25
26      d_m2 = alpha(2,:);
27      x_T = g_obs(1,:); x_M1 = g_obs(2,:); x_M2 = g_obs(3,:);
28      x_T_prim = g_prim(1,:);
29
30      dm1 = (r*x_T.*(1 - x_T/beta_T) + d_m2.*x_M2.*x_T - x_T_prim)./(
        ↪ x_M1.*x_T);
31
32      dm1_guess = dm1;
33
34      % plot(time_mesh,dm1_guess,'o')
35  end
```

calculate_alpha_exp.m

```
1   function alpha_exp=calculate_alpha_exp(alpha,alpha_unknown,x,t_min,
        ↪ t_max)
2
3   x_T=x(1,:);
4   x_M1=x(2,:);
5   x_M2=x(3,:);
6   m=length(x_T) - 1; %ändrade till -1 då jag tycker steglängden borde
        ↪  bli det
7   time_step=(t_max-t_min)/m;
8   alpha_exp=zeros(m-1,1); %ändrade från m-2 till m-1
9
10  for k=2:m
11      x_k=x(:,k);
12
```

```
        dxT_dt=(x_T(k+1)-x_T(k-1))/(2*time_step);
        dxM1_dt=(x_M1(k+1)-x_M1(k-1))/(2*time_step);
        dxM2_dt=(x_M2(k+1)-x_M2(k-1))/(2*time_step);

        dx_dt=[dxT_dt dxM1_dt dxM2_dt];
        alpha_exp(k-1)=calc_explicit(alpha,alpha_unknown,x_k,dx_dt)
    ↪ ;
end
end


function alpha_exp_k=calc_explicit(alpha,alpha_unknown,x,dx_dt)
dm1=alpha(1);
dm2=alpha(2);
at1=alpha(3);
at2=alpha(4);
k12=alpha(5);
r=0.93;
deltaM1=0.173;
deltaM2=0.173;
betaT=3*10^9;
betaM=9*10^8;

xT=x(1);
xM1=x(2);
xM2=x(3);

dxT_dt=dx_dt(1);
dxM1_dt=dx_dt(2);
dxM2_dt=dx_dt(3);

if alpha_unknown==1
    alpha_exp_k=(-dxT_dt+r*xT*(1-xT/betaT)+dm2*xM2*xT)/(xT*xM1);
elseif alpha_unknown==2
    alpha_exp_k=(dxT_dt-r*xT*(1-xT/betaT)+dm1*xM1*xT)/(xT*xM2);
elseif alpha_unknown==3
    alpha_exp_k=(dxM1_dt+deltaM1*xM1+k12*xM1*xT)/(xT*xM1*(1-(xM1+
    ↪ xM2)/betaM));
elseif alpha_unknown==4
    alpha_exp_k=(dxM2_dt+deltaM2*xM2-k12*xM1*xT)/(xT*xM2*(1-(xM1+
    ↪ xM2)/betaM));
else
    alpha_exp_k=(dxM1_dt-at1*xT*xM1*(1-(xM1+xM2)/betaM)-deltaM1*xM1
    ↪ )/(xT*xM1);
end
end
```

Create_Adjoint_figure.m

```
%%  Initials
clc; close all; clear all;
%%initial of x and time variables
m = 15; % number of observations
obs_start = 2.1; obs_end = 7; %interval of observations
time_final = 20;
```

```
time_start = 0;

time_mesh = linspace(time_start,time_final,m);
% x_initial = [x_T(0); x_M1(0); x_M2(0)]
% initial values that seems to fit fig 1a
x_initial = [5*10^6; 10^3; 10^3];

gron = [102,194,165]/255;
orange = [252,141,98]/255;
lila = [141,160,203]/255;


%% Plot forward
alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh);
noise_level = 0.1;% 10% noise

%using ode45 since newton seems to have problems with low numbers
    ↪ for
% this particular system of odes

[g, g_brus, g_add] = ExactODE45(alpha,time_mesh,noise_level,
    ↪ x_initial);
figure
hold on

m2=1000;
time_mesh2 = linspace(time_start,time_final,m2);
alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh2);
FN = ForwardNewton(alpha,time_mesh2,x_initial);
F45 = ForwardODE45(alpha,time_mesh2,x_initial);

subplot(1,2,1);
plot(time_mesh2,FN(1,:),'-','Color',orange,'linewidth',2)
    hold on
    plot(time_mesh2,F45(1,:),'--','Color',lila,'linewidth',2)
title('Framåtproblemet');
xlabel('tid (dagar)');
legend('x_T Newton', 'x_T ode45')

%% Plot Adjoint

time_obs = linspace(obs_start, obs_end, 15);
alpha_obs = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_obs);
[g, g_brus, g_add] = ExactODE45(alpha_obs,time_obs,noise_level,
    ↪ x_initial); % get observations
%interpolate to be able to use for other time meshes
gPol =@(t) [];
for j = 1:size(g_brus,1)
   Pol_j =@(t) interp1(time_mesh,g_brus(j,:),t);
   gPol =@(t) [gPol(t); Pol_j(t)];
end

time_mesh2 = linspace(time_start,time_final,m2);
alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh2);
g_brus = gPol(time_mesh2);
FN = ForwardNewton(alpha,time_mesh2,x_initial);
```

```
60 F45 = ForwardODE45(alpha,time_mesh2,x_initial);
61 AN = AdjointNewton(alpha, FN, g_brus, time_mesh2, obs_start,
   ↪ obs_end);
62 A45 = AdjointODE45(alpha, F45, g_brus, time_mesh2, obs_start,
   ↪ obs_end);
63
64 subplot(1,2,2);
65 plot(time_mesh2,AN(2,:),'-','Color',orange,'linewidth',2)
66     hold on
67 plot(time_mesh2,A45(2,:),'--','Color',lila,'linewidth',2)
68 title('Bakåtproblemet');
69 xlabel('tid (dagar)');
70 legend('x_T Newton', 'x_T ode45')
71
72
73 %% creating some special figures
74
75
76
77
78
79
80 %% observations
81 alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh);
82 noise_level = 0.1;% 10% noise
83
84 %using ode45 since newton seems to have problems with low numbers
   ↪ for
85 % this particular system of odes
86
87 [g, g_brus, g_add] = ExactODE45(alpha,time_mesh,noise_level,
   ↪ x_initial);
88 figure
89
90 plot(time_mesh,g(1,:),'linewidth',2)
91 hold on
92 plot(time_mesh,g_brus(1,:),'*')
93   legend('x_T, ode45','observations g1')
94   title(['ODE45 versus noisy data, noise , \delta= ',num2str(
   ↪ noise_level)]);
95
96
97 %% Test solver quality : Forward
98 % figure
99 %
100 % hold on
101 % sch = 0;
102 % for m2 = [50 100 200 400 800 1000]
103 %     time_mesh2 = linspace(time_start,time_final,m2);
104 %     alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,
   ↪ time_mesh);
105 %     FN = ForwardNewton(alpha,time_mesh2,x_initial);
106 %     F45 = ForwardODE45(alpha,time_mesh2,x_initial);
107 %
108 % sch = sch+1;
109 % subplot(2, 3, sch);
```

```
%
% plot(time_mesh2,FN(1,:),'-r',time_mesh2,F45(1,:),'--b','linewidth
    ↪ ',2);
% title(' forward problem');
% xlabel(m2)
%
% legend('x_T Newton', 'x_T ode45')
% %     text(time_mesh2(end),FN(1,end),['N', num2str(m2)])
% %     text(time_mesh2(end),FN(1,end),['45', num2str(m2)])
% end
%
% grid on
%
%
% hold off


%% Test solver quality : Adjoint

noise_level = 0.1;
time_obs = linspace(obs_start, obs_end, 15);
alpha_obs = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh)
    ↪ ;
[g, g_brus, g_add] = ExactODE45(alpha_obs,time_obs,noise_level,
    ↪ x_initial); % get observations
%interpolate to be able to use for other time meshes
gPol =@(t) [];
for j = 1:size(g_brus,1)
    Pol_j =@(t) interp1(time_mesh,g_brus(j,:),t);
    gPol =@(t) [gPol(t); Pol_j(t)];
end


figure
hold on

sch = 0;

for m2 = [200 800 2000 20000]
    disp(m2)
    time_mesh2 = linspace(time_start,time_final,m2);
    alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh2
    ↪ );
    g_brus = gPol(time_mesh2);
    FN = ForwardNewton(alpha,time_mesh2,x_initial);
    F45 = ForwardODE45(alpha,time_mesh2,x_initial);
    AN = AdjointNewton(alpha, FN, g_brus, time_mesh2, obs_start,
    ↪ obs_end);
    A45 = AdjointODE45(alpha, F45, g_brus, time_mesh2, obs_start,
    ↪ obs_end);

    sch = sch+1;
    subplot(2, 2, sch);
    subindex = {'a) ', 'b) ', 'c) ', 'd) '};
    plot(time_mesh2,AN(1,:),'-','Color',orange,'linewidth',2)
    hold on
```

```
    plot(time_mesh2,A45(1,:),'--','Color',lila,'linewidth',2)
    xlabel(m2)
    title([subindex{sch}, 'Adjoint problem']);
    legend('x_T Newton', 'x_T ode45')




end
figure

for m2=[2000]
    disp(m2)
    time_mesh2 = linspace(time_start,time_final,m2);
    alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh2
    ↪ );
    g_brus = gPol(time_mesh2);
    FN = ForwardNewton(alpha,time_mesh2,x_initial);
    F45 = ForwardODE45(alpha,time_mesh2,x_initial);
    AN = AdjointNewton(alpha, FN, g_brus, time_mesh2, obs_start,
    ↪ obs_end);
    A45 = AdjointODE45(alpha, F45, g_brus, time_mesh2, obs_start,
    ↪ obs_end);

    subplot(1,2,1)
    plot(time_mesh2,FN(1,:),'-','Color',orange,'linewidth',2)
    hold on
    plot(time_mesh2,F45(1,:),'--','Color',lila,'linewidth',2)
    title('Forward problem');
    legend('x_T Newton', 'x_T ode45')
    xlim([5 20])
    subplot (1,2,2)
    plot(time_mesh2,AN(2,:),'-','Color',orange,'linewidth',2)
    hold on
    plot(time_mesh2,A45(2,:),'--','Color',lila,'linewidth',2)
    title('Adjoint problem');
    legend('x_T Newton', 'x_T ode45')
    xlim([5 20])


end



hold off


%% Inner functions

function alpha = alpha_vec(dm1,dm2,at1,at2,k12,time_mesh)
    scaling_factor_dm1 = dm1;
    scaling_factor_dm2 = dm2;
    scaling_factor_at1 = at1;
    scaling_factor_at2 = at2;
    scaling_factor_k12 = k12;
```

```
213
214     function_flag = 0; % constant
215
216     exact_dm1 = ExactParameter(scaling_factor_dm1,function_flag,
    ↪ time_mesh); %Exact profile for dm1 to produce data.
217     exact_dm2 = ExactParameter(scaling_factor_dm2,function_flag,
    ↪ time_mesh); %Exact profile for dm2 to produce data.
218     exact_at1 = ExactParameter(scaling_factor_at1,function_flag,
    ↪ time_mesh); %Exact profile for at1 to produce data.
219     exact_at2 = ExactParameter(scaling_factor_at2,function_flag,
    ↪ time_mesh); %Exact profile for at2 to produce data.
220     exact_k12 = ExactParameter(scaling_factor_k12,function_flag,
    ↪ time_mesh); %Exact profile for k12 to produce data.
221
222     alpha = [exact_dm1; exact_dm2; exact_at1; exact_at2; exact_k12
    ↪ ];
223
224 end
```

ExactNewton.m

```
1 %Computes the exact solution of the model problem.
2 function [g,g_brus,g_add] = ExactNewton(alpha, time_mesh,
    ↪ noise_level,x_initial)%(alpha, time_mesh,obs_start,obs_end,
    ↪ noise_level,x_initial)
3
4
5     nodes = length(time_mesh); % Number of nodes in the time
    ↪ partition
6
7     g = ForwardNewton(alpha,time_mesh,x_initial);
8 %     lb = find(obs_start < time_mesh,1)-1;
9 %     ub = nodes - find(obs_end > flip(time_mesh),1) + 2;
10 %     g(:,1:lb) = 0; g(:,ub:end) = 0;
11
12     brus = (rand(3,nodes)*2 - 1)*noise_level;
13     g_brus = g + g.*brus;
14
15     g_add = g + g*noise_level;
16 end
```

ExactODE45.m

```
1 %Computes the exact solution of the model problem.
2 function [g,g_brus,g_add] = ExactODE45(alpha, time_mesh,noise_level
    ↪ ,x_initial)%(alpha, time_mesh,obs_start,obs_end,noise_level,
    ↪ x_initial)
3
4
5     nodes = length(time_mesh); % Number of nodes in the time
    ↪ partition
6
7     g = ForwardODE45(alpha,time_mesh,x_initial);
8 %     lb = find(obs_start < time_mesh,1)-1;
```

```
%      ub = nodes - find(obs_end > flip(time_mesh),1) + 2;
%
%      g(:,1:lb) = 0; g(:,ub:end) = 0;

    brus = (rand(3,nodes)*2 - 1)*noise_level;
    g_brus = g + g.*brus;

    g_add = g + g*noise_level;
end
```

ExactParameter.m

```
function exact_par = ExactParameter(scaling,flag,time_mesh)
    %Input: 1. scaling factor
    %2. function flag (0. const. 1. +linear 2. -linear 3. sin 4.
    ↪ exp(-x))
    %time = 0;
    s = zeros(1,length(time_mesh));
    s = 3*time_mesh/time_mesh(end);
    if flag == 1
        exact_par = scaling*s/3;
    elseif flag == 2
        exact_par = scaling*(1-s/3);
    elseif flag == 3
        exact_par = scaling*sin(s);
    elseif flag == 4
        exact_par = scaling*exp(-s);
    else
        exact_par = scaling*ones(1,length(time_mesh));
    end


%    figure
%   plot(time_mesh,exact_par,'b --s','LineWidth',3);
%legend(' linear 1')
% title('Test function in time')

end
```

Forwardfunc.m

```
function f = Forwardfunc(x,alpha)
    % alpha = (d_m1, d_m2, a_t1, a_t2, k_12)
     r = 0.93;
     beta_T = 3*10^9;
     d_m1 = alpha(1);
     d_m2 = alpha(2);
     a_t1 = alpha(3);
     a_t2 = alpha(4);
     beta_M = 9*10^8;
     sigma_m1 = 0.173;
     sigma_m2 = 0.173;
     k_12 = alpha(5);

```

```
    x_T = x(1); x_M1 = x(2); x_M2 = x(3);

    f1 = r*x_T*(1 - x_T/beta_T) - d_m1*x_M1*x_T + d_m2*x_M2*x_T;
    f2 = a_t1*x_T*x_M1*(1 - (x_M1+x_M2)/beta_M) - sigma_m1*x_M1 -
    ↪ k_12*x_M1*x_T;
    f3 = a_t2*x_T*x_M2*(1 - (x_M1+x_M2)/beta_M) - sigma_m2*x_M2 +
    ↪ k_12*x_M1*x_T;
    f = [f1; f2; f3];
end
```

ForwardfuncJacobi.m

```
function f = ForwardfuncJacobi(x,alpha)
    % alpha = (d_m1, d_m2, a_t1, a_t2, k_12)
     r = 0.93;
     beta_T = 3*10^9;
     d_m1 = alpha(1);
     d_m2 = alpha(2);
     a_t1 = alpha(3);
     a_t2 = alpha(4);
     beta_M = 9*10^8;
     sigma_m1 = 0.173;
     sigma_m2 = 0.173;
     k_12 = alpha(5);

     x_T = x(1); x_M1 = x(2); x_M2 = x(3);
    f = zeros(3,3);

    f(1,1) = r*(1-2*x_T/beta_T) - d_m1*x_M1 + d_m2*x_M2;
    f(1,2) = -d_m1*x_T;
    f(1,3) = d_m2*x_T;
    f(2,1) = a_t1*x_M1*(1-(x_M1+x_M2)/beta_M) - k_12*x_M1;
    f(2,2) = a_t1*x_T*(1-(2*x_M1+x_M2)/beta_M) - sigma_m1 - k_12*
    ↪ x_T;
    f(2,3) = -a_t1*x_T*x_M1/beta_M;
    f(3,1) = a_t2*x_M2*(1-(x_M1+x_M2)/beta_M) + k_12*x_M1;
    f(3,2) = -a_t2*x_T*x_M2/beta_M + k_12*x_T;
    f(3,3) = a_t2*x_T*(1-(x_M1+2*x_M2)/beta_M) - sigma_m2;

end
```

ForwardNewton.m

```
function x = ForwardNewton(alpha,time_mesh,x_initial)
%Computes the forward solution of the model problem.

    nodes = length(time_mesh)-1;

    MaxIter = 1000;  % maximal number of iterations in Newton
    ↪ method

    x = zeros(size(x_initial,1),nodes+1);

    x(:,1) = x_initial;
```

```
11
12      dt = time_mesh(2:end)-time_mesh(1:end-1);   %Time step
13
14      v = x_initial;
15      for i = 1:nodes
16          tol=1;
17          iter=0;
18          while tol>10^(-10) && iter < MaxIter   %Newton iterations
19              F= v - x(:,i) - dt(i)*Forwardfunc(v,alpha(:,i));
20              J=eye(length(x_initial)) - dt(i)*ForwardfuncJacobi(v,
    ↪ alpha(:,i));
21              dv = -J\F;
22              if norm(F) <10^(-5)
23  %                   disp('norm(F) is very small! Newtons method
    ↪ stops at time step ')
24  %                   time_mesh(i)
25                  break
26              end
27
28              v=v+dv;                %The Newton iteration
29              iter = iter +1;
30              tol = norm(dv,inf);
31          end
32          if iter==MaxIter         %If the Newton meth. does not
    ↪ converge
33              disp(['No convergence in the Newton method at time: '
    ↪ num2str(time_mesh(i))])
34              break
35          end
36          for k = 1:length(x_initial)
37              if v(k) < 0
38                  warning('Forward solution algorithm yields invalid
    ↪ solution. Try increasing the number of nodes in the time
    ↪ partition.')
39                  return
40              end
41          end
42          x(:,i+1) =  v;
43      end
44
45  end
```

ForwardODE45.m

```
1   function f = ForwardODE45(alpha,time_mesh,x_initial)
2       [time,f] = ode45(@(t,x) Forfunc(t,x,alpha,time_mesh),time_mesh,
    ↪ x_initial);
3       f = f';
4       function func = Forfunc(t,x,alpha,time_mesh)
5           alphaPol =@(t) [];
6           for i = 1:size(alpha,1)
7              Pol_i =@(t) interp1(time_mesh,alpha(i,:),t);
8              alphaPol =@(t) [alphaPol(t); Pol_i(t)];
9           end
10          alphat = alphaPol(t);
```

```
        func = Forwardfunc(x,alphat);
    end

end
```

leastsquare.m

```
clear,clc
% ----------------------------------------
%   Solution of least squares problem  min_x || Ax - y ||_2
% ----------------------------------------

grad=5; % graden på polynomet
points=200; %antalet diskreta punkter = ( rader i matrisen A)


% x = [1 2 3 4 5];
% y = [3 4 3.5 5 6];

x=zeros(1,points);
y=zeros(1,points);
V=[];
for i=1:1:points
  x = linspace(-10.0,10.0,points);
  %  exakt funktion
  y(i)= sin(pi*x(i)/5); %+ x(i)/5;
  % y=rand(1,i)*0.1;
  %y(i) = 1./(1+x(i).^2) + 0.1*randn(size(x(i)));

end


%skapar vandermode matrisen
V = bsxfun(@power,x(:),0:grad);
%vandemorde = V
% V*A=Y ekvation som ska lösas
% VL: V*V'*A dvs. VL=A
% L SNINGEN V*HL "(HL=A)"
%

% beräknar högeledet
HL=V'*y';

% beräknar vänsterledet
VL=V'*V;

%VL=zeros(degree+1);

% Lösning av normalekvationen med Cholesky faktorisering
%symmetri=issymmetric(VL)
if all(eig(VL) > 0) % vi kollar om matrisen är positivt definit
   ↪ eftersom chol kräver "symmetrisk positiv definita matriser"
   ↪ annars börjar den anta
    VL = chol(VL, 'lower'); % chol faktoriserar matrisen lhs till
   ↪ en triangulär matris så att lhs=l'*l
```

```
    %felet kan kollas med:
    %norm(l*l' - lhs )
    HL = VL' \ (VL \ HL);

    figure(1)
plot(x,y,'-- r', 'linewidth',2)
hold on

% beräknar approximation till det exakta polynomet med
    ↪ coefficienter c

polynom = V*HL;
plot(x,polynom,': b', 'linewidth',2)
hold off

str_xlabel = ['polynomgrad: ', num2str(grad)];

% Beräknar det relativa felet
%  norm(approx. value - true value) / norm(true value)
%e1=norm(y'- approx)/norm(y')

error=norm(y'- polynom)/norm(y');
%fprintf('Relativt fel: %.2f%%', error);

legend('exakt ',str_xlabel);
title(['Relativt fel: ' num2str(error)])
xlabel('x')
else
    error('Det uppfylls inte att matrisen är positivt definit; en
    ↪ matris A sådan att x^(T) Ax > 0 för alla x');
end
```

LinearClassNormalEqExample2.m

```
function [approx] = LinearClassNormalEqExample2(y,x,m)
    % ----------------------------------------
    %   Solution of least squares  problem  min_x || Ax - y ||_2
    %   using the method of normal equations.
    %   Matrix  A is constructed as a Vandermonde matrix.
    % ----------------------------------------


     d=5;  % degree of the polynomial
    %m=50;%number of discretization points or rows in the matrix A

    %y=zeros(1,m);
    A=[];

    %Values for parameter eta
    ↪ ------------------------------------------------
    scaling_factor = 0.7;
    %function_flag = 1;  % flag means choose of the model function
    ↪ for eta: can be chosen between 1 and 4
    %eta(t) = scaling_factor*function

```

```
20    % ext_eta = ExactEta(scaling_factor,function_flag,x); %Exact
    ↪ eta.
21
22
23
24    % construction of a Vandermonde matrix
25
26    for i=1:m
27      for j=1:d+1
28        A(i,j)=power(x(i),j-1);
29      end
30    end
31
32
33    % plot(x,y,'o')
34
35    % computing the right hand side in the method of normal
    ↪ equations
36    c=A'*y';
37
38    % computing matrix in the left hand side in the method of
    ↪ normal equations
39    C=A'*A;
40
41    l=zeros(d+1);
42
43    % solution of the normal equation using Cholesky decomposition
44
45    for j=1:d+1
46      s1=0;
47      for k=1:j-1
48        s1=s1+l(j,k)*l(j,k);
49      end
50      l(j,j)=(C(j,j)-s1)^(1/2);
51      for i=j+1:d+1
52        s2=0;
53        for k=1:j-1
54          s2=s2+l(i,k)*l(j,k);
55        end
56        l(i,j)=(C(i,j)-s2)/l(j,j);
57      end
58    end
59    for i=1:d+1
60      for k=1:1:i-1
61        c(i)=c(i)-c(k)*l(i,k);
62      end
63      c(i)=c(i)/l(i,i);
64    end
65    for i=d+1:-1:1
66      for k=d+1:-1:i+1
67        c(i)=c(i)-c(k)*l(k,i);
68      end
69      c(i)=c(i)/l(i,i);
70    end
71
72    size(x);
```

```
73
74      size(y);
75      figure(1)
76      plot(x,y,'o r', 'linewidth',2)
77      hold on
78
79      % compute approximation to this exact polynomial with comp.
        ↪ coefficients c
80
81      approx = A*c;
82      plot(x,approx,'*- ', 'linewidth',2)
83
84
85      str_xlabel = ['poly.degree d=', num2str(d)];
86
87
88
89      xlabel('Time')
90      legend('\eta from measured data',' approximated guess for \
        ↪ eta_0');
91      title(['Least squares  for classification, number of input
        ↪ points ',num2str(m)])
92
93      % computation of the relative error as
94      %  norm(approx. value - true value) / norm(true value)
95      %e1=norm(y'- approx)/norm(y')
96  end
```

mod_calculate_alpha_exp_del1.m

```
1  function alpha_exp=mod_calculate_alpha_exp_del1(alpha,alpha_unknown
       ↪ ,x,t_min,t_max)
2
3  x_T=x(1,:);
4  x_M1=x(2,:);
5  x_M2=x(3,:);
6  m=length(x_T) - 1; %ändrade till -1 då jag tycker steglängden borde
       ↪  bli det
7  time_step=(t_max-t_min)/m;
8  alpha_exp=zeros(m-1,1); %ändrade från m-2 till m-1
9
10 for k=2:m
11      x_k=x(:,k);
12
13          dxT_dt=(x_T(k+1)-x_T(k-1))/(2*time_step);
14          dxM1_dt=(x_M1(k+1)-x_M1(k-1))/(2*time_step);
15          dxM2_dt=(x_M2(k+1)-x_M2(k-1))/(2*time_step);
16
17          dx_dt=[dxT_dt dxM1_dt dxM2_dt];
18
19          alpha_exp(k-1)=calc_explicit_FE(alpha,alpha_unknown,x_k,
        ↪ dx_dt);
20 end
21 end
22
```

```
23
24
25
26
27 function alpha_exp_k_FE=calc_explicit_FE(alpha,alpha_unknown,x,
       ↪ dx_dt);
28 dm1=alpha(1);
29 dm2=alpha(2);
30 at1=alpha(3);
31 at2=alpha(4);
32 k12=alpha(5);
33 r=0.93;
34 deltaM1=0.173;
35 deltaM2=0.173;
36 betaT=3*10^9;
37 betaM=9*10^8;
38
39 xT=x(1);
40 xM1=x(2);
41 xM2=x(3);
42
43 dxT_dt=dx_dt(1);
44 dxM1_dt=dx_dt(2);
45 dxM2_dt=dx_dt(3);
46
47 if alpha_unknown==1
48     alpha_exp_k_FE = r*(1-xT/betaT)/xM1 - dxT_dt/(xM1*xT) ; %+ dm2*
       ↪ xM2/xM1
49 elseif alpha_unknown==2
50     alpha_exp_k_FE = dxT_dt/(xM2*xT) - r*(1-xT/betaT)/xM2 ; %+ dm1*
       ↪ xM1/xM2
51 elseif alpha_unknown==3
52     alpha_exp_k_FE = dxM1_dt/(xT*xM1*(1 - (xM2+xM1)/betaM)) +
       ↪ deltaM1/(xT*(1-(xM1+xM2)/betaM)); %  + k12/(1-(xM1+xM2)/betaM
       ↪ )
53 elseif alpha_unknown==4
54     alpha_exp_k_FE = dxM2_dt/(xT*xM1*(1 - (xM2+xM1)/betaM)) +
       ↪ deltaM2/(xT*(1 - (xM2+xM1)/betaM));%  - k12/(1 - (xM2+xM1)/
       ↪ betaM)
55 else
56     alpha_exp_k_FE = -dxM1_dt/(xM1*xT) - deltaM1/xT ; %+ at1*(1-(
       ↪ xM1 + xM2)/betaM)
57 end
58 end
```

mod_calculate_alpha_exp_del2.m

```
1 function alpha_exp=mod_calculate_alpha_exp_del2(alpha,alpha_unknown
      ↪ ,x,t_min,t_max)
2
3 x_T=x(1,:);
4 x_M1=x(2,:);
5 x_M2=x(3,:);
6 m=length(x_T) - 1; %ändrade till -1 då jag tycker steglängden borde
      ↪  bli det
```

```
time_step=(t_max-t_min)/m;
alpha_exp=zeros(m-1,1); %ändrade från m-2 till m-1

for k=2:m
    x_k=x(:,k);

        dxT_dt=(x_T(k+1)-x_T(k-1))/(2*time_step);
        dxM1_dt=(x_M1(k+1)-x_M1(k-1))/(2*time_step);
        dxM2_dt=(x_M2(k+1)-x_M2(k-1))/(2*time_step);

        dx_dt=[dxT_dt dxM1_dt dxM2_dt];

        alpha_exp(k-1)=calc_explicit_FE(alpha,alpha_unknown,x_k,
    ↪ dx_dt);
end
end



function alpha_exp_k_FE=calc_explicit_FE(alpha,alpha_unknown,x,
    ↪ dx_dt);
dm1=alpha(1);
dm2=alpha(2);
at1=alpha(3);
at2=alpha(4);
k12=alpha(5);
r=0.93;
deltaM1=0.173;
deltaM2=0.173;
betaT=3*10^9;
betaM=9*10^8;

xT=x(1);
xM1=x(2);
xM2=x(3);

dxT_dt=dx_dt(1);
dxM1_dt=dx_dt(2);
dxM2_dt=dx_dt(3);

if alpha_unknown==1
    alpha_exp_k_FE = - dxT_dt/(xM1*xT) + dm2*xM2/xM1;%r*(1-xT/betaT
    ↪ )/xM1
elseif alpha_unknown==2
    alpha_exp_k_FE = dxT_dt/(xM2*xT)  + dm1*xM1/xM2; % - r*(1-xT/
    ↪ betaT)/xM2
elseif alpha_unknown==3
    alpha_exp_k_FE = dxM1_dt/(xT*xM1*(1 - (xM2+xM1)/betaM))  + k12
    ↪ /(1-(xM1+xM2)/betaM); % + deltaM1/(xT*(1-(xM1+xM2)/betaM))
elseif alpha_unknown==4
    alpha_exp_k_FE = dxM2_dt/(xT*xM1*(1 - (xM2+xM1)/betaM))  - k12
    ↪ /(1 - (xM2+xM1)/betaM); % + deltaM2/(xT*(1 - (xM2+xM1)/betaM)
    ↪ )
else
    alpha_exp_k_FE = -dxM1_dt/(xM1*xT)  + at1*(1-(xM1 + xM2)/betaM)
```

```
    ↪ ; % - deltaM1/xT
end

end
```

mod_calculate_alpha_exp_del3.m

```
function alpha_exp=mod_calculate_alpha_exp_del3(alpha,alpha_unknown
    ↪ ,x,t_min,t_max)

x_T=x(1,:);
x_M1=x(2,:);
x_M2=x(3,:);
m=length(x_T) - 1;
time_step=(t_max-t_min)/m;
alpha_exp=zeros(m-1,1);

for k=2:m
    x_k=x(:,k);

        dxT_dt=(x_T(k+1)-x_T(k-1))/(2*time_step);
        dxM1_dt=(x_M1(k+1)-x_M1(k-1))/(2*time_step);
        dxM2_dt=(x_M2(k+1)-x_M2(k-1))/(2*time_step);

        dx_dt=[dxT_dt dxM1_dt dxM2_dt];

        alpha_exp(k-1)=calc_explicit_FE(alpha,alpha_unknown,x_k,
    ↪ dx_dt);
end
end




function alpha_exp_k_FE=calc_explicit_FE(alpha,alpha_unknown,x,
    ↪ dx_dt);
dm1=alpha(1);
dm2=alpha(2);
at1=alpha(3);
at2=alpha(4);
k12=alpha(5);
r=0.93;
deltaM1=0.173;
deltaM2=0.173;
betaT=3*10^9;
betaM=9*10^8;

xT=x(1);
xM1=x(2);
xM2=x(3);

dxT_dt=dx_dt(1);
dxM1_dt=dx_dt(2);
dxM2_dt=dx_dt(3);
```

```
46
47 if alpha_unknown==1
48     alpha_exp_k_FE = r*(1-xT/betaT)/xM1 + dm2*xM2/xM1 ; %- dxT_dt/(
   ↪ xM1*xT)
49 elseif alpha_unknown==2
50     alpha_exp_k_FE = - r*(1-xT/betaT)/xM2 + dm1*xM1/xM2; %dxT_dt/(
   ↪ xM2*xT)
51 elseif alpha_unknown==3
52     alpha_exp_k_FE = +deltaM1/(xT*(1-(xM1+xM2)/betaM))+ k12/(1-(xM1
   ↪ +xM2)/betaM); %  dxM1_dt/(xT*xM1*(1 - (xM2+xM1)/betaM))
53 elseif alpha_unknown==4
54     alpha_exp_k_FE =  + deltaM2/(xT*(1 - (xM2+xM1)/betaM))  - k12
   ↪ /(1 - (xM2+xM1)/betaM);%dxM2_dt/(xT*xM1*(1 - (xM2+xM1)/betaM)
   ↪ )
55 else
56     alpha_exp_k_FE = - deltaM1/xT + at1*(1-(xM1 + xM2)/betaM); %-
   ↪ dxM1_dt/(xM1*xT)
57 end
58 end
```

mod_calculate_alpha_exp.m

```
1  function alpha_exp=mod_calculate_alpha_exp(alpha,alpha_unknown,x,
   ↪ t_min,t_max)
2
3  x_T=x(1,:);
4  x_M1=x(2,:);
5  x_M2=x(3,:);
6  m=length(x_T) - 1; %ändrade till -1 då jag tycker steglängden borde
   ↪  bli det
7  time_step=(t_max-t_min)/m;
8  alpha_exp=zeros(m-1,1); %ändrade från m-2 till m-1
9
10 for k=2:m
11     x_k=x(:,k);
12
13         dxT_dt=(x_T(k+1)-x_T(k-1))/(2*time_step);
14         dxM1_dt=(x_M1(k+1)-x_M1(k-1))/(2*time_step);
15         dxM2_dt=(x_M2(k+1)-x_M2(k-1))/(2*time_step);
16
17         dx_dt=[dxT_dt dxM1_dt dxM2_dt];
18
19         alpha_exp(k-1)=calc_explicit_FE(alpha,alpha_unknown,x_k,
   ↪ dx_dt);
20 end
21 end
22
23
24
25
26 function alpha_exp_k_FE=calc_explicit_FE(alpha,alpha_unknown,x,
   ↪ dx_dt);
27 dm1=alpha(1);
28 dm2=alpha(2);
29 at1=alpha(3);
```

```
at2=alpha(4);
k12=alpha(5);
r=0.93;
deltaM1=0.173;
deltaM2=0.173;
betaT=3*10^9;
betaM=9*10^8;

xT=x(1);
xM1=x(2);
xM2=x(3);

dxT_dt=dx_dt(1);
dxM1_dt=dx_dt(2);
dxM2_dt=dx_dt(3);

if alpha_unknown==1
    alpha_exp_k_FE = r*(1-xT/betaT)/xM1 - dxT_dt/(xM1*xT) + dm2*xM2 /xM1;
elseif alpha_unknown==2
    alpha_exp_k_FE = dxT_dt/(xM2*xT) - r*(1-xT/betaT)/xM2 + dm1*xM1 /xM2;
elseif alpha_unknown==3
    alpha_exp_k_FE = dxM1_dt/(xT*xM1*(1 - (xM2+xM1)/betaM)) + deltaM1/(xT*(1-(xM1+xM2)/betaM)) + k12/(1-(xM1+xM2)/betaM);
elseif alpha_unknown==4
    alpha_exp_k_FE = dxM2_dt/(xT*xM1*(1 - (xM2+xM1)/betaM)) + deltaM2/(xT*(1 - (xM2+xM1)/betaM)) - k12/(1 - (xM2+xM1)/betaM );
else
    alpha_exp_k_FE = -dxM1_dt/(xM1*xT) - deltaM1/xT + at1*(1-(xM1 + xM2)/betaM);
end

end
```

mod_test_alpha_exp_Chebyshew_plot.m

```
clc; close all; clear all;

t_min=0;t_max=20; m=500; x_initial = [5*10^6; 10^3; 10^3];%För att ha startvärden nära ett steady state sätt in [3.1298490038*10^9; 10^-5; 402531911.894];
time_mesh =[];
for k = 1:m
    time_mesh(end+1) = -(t_max-t_min)/2*cos((2*k-1)*pi/(2*m)) + (t_max+t_min)/2;
end


alpha =  [10^-11 10^-12 10^-10 10^-12 10^-12]*100;
alpha_1=alpha_vec(alpha(1),alpha(2),alpha(3),alpha(4),alpha(5), time_mesh);
x_23s = ForwardODE23s(alpha_1,time_mesh,x_initial);
x_Newton = ForwardNewton(alpha_1,time_mesh,x_initial);
```

```
x_45 = ForwardODE45(alpha_1,time_mesh,x_initial);
t_plot = [1:500];


gron = [102,194,165]/255;
orange = [252,141,98]/255;
lila = [141,160,203]/255;

alpha_unknown=5;
%Förenklad
alpha_exp_23s=mod_calculate_alpha_exp(alpha,alpha_unknown,x_23s,
    ↪ t_min,t_max);
alpha_exp_Newton=mod_calculate_alpha_exp(alpha,alpha_unknown,
    ↪ x_Newton,t_min,t_max);
alpha_exp_45=mod_calculate_alpha_exp(alpha,alpha_unknown,x_45,t_min
    ↪ ,t_max);
alpha_exp_23s_T = alpha_exp_23s';
alpha_mod_medel = mean(alpha_exp_23s_T([250:498]))

%orginal
alpha_exp_23s_org=calculate_alpha_exp(alpha,alpha_unknown,x_23s,
    ↪ t_min,t_max);
alpha_exp_Newton_org=calculate_alpha_exp(alpha,alpha_unknown,
    ↪ x_Newton,t_min,t_max);
alpha_exp_45_org=calculate_alpha_exp(alpha,alpha_unknown,x_45,t_min
    ↪ ,t_max);
alpha_exp_23s_org_T = alpha_exp_23s_org';
alpha_mod_medel = mean(alpha_exp_23s_org_T([250:498]))


max_45_org = max(alpha_exp_45_org);
max_Newton_org = max(alpha_exp_Newton_org);
max_23s_org = max(alpha_exp_23s_org);

max_45_mod = max(alpha_exp_45);
max_Newton_mod = max(alpha_exp_Newton);
max_23s_mod = max(alpha_exp_23s);


max_org = max([max_45_org max_Newton_org max_23s_org]);
max_mod = max([max_45_mod max_Newton_mod max_23s_mod]);

min_45_org = min(alpha_exp_45_org);
min_Newton_org = min(alpha_exp_Newton_org);
min_23s_org = min(alpha_exp_23s_org);

min_45_mod= min(alpha_exp_45);
min_Newton_mod = min(alpha_exp_Newton);
min_23s_mod = min(alpha_exp_23s);

min_org = min([min_45_org min_Newton_org min_23s_org]);
min_mod = min([min_45_mod min_Newton_mod min_23s_mod]);

figure('name','Chebyshew')
subplot(2,1,1)
plot(time_mesh(2:end-1),alpha_exp_23s,'color', gron ,LineWidth=2)
```

```
hold on
plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--',
    ↪ LineWidth=1)
legend('Explicit beräkningar, ode23s','Sannt paramtetervärde')
title('Förenklad','FontSize',14)
ylim([min_mod-max_mod/3,max_mod*1.3])
xlabel('Dagar','FontSize',12,'FontWeight','bold')
ylabel('Parametervärde','FontSize',12,'FontWeight','bold')
%ylim([-5*10^-8 10^-7]) %För paramteter 5

subplot(2,1,2)
plot(time_mesh(2:end-1),alpha_exp_23s_org,'color',gron,LineWidth=2)
hold on
plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--',
    ↪ LineWidth=1)
legend('Explicit beräkningar, ode23s','Sannt paramtetervärde')
title('Original','FontSize',14)
ylim([min_org-max_org/3,max_org*1.3])
xlabel('Dagar','FontSize',12,'FontWeight','bold')
ylabel('Parametervärde','FontSize',12,'FontWeight','bold')
hold on
%ylim([-5*10^-8 10^-7]) %För paramteter 5


fontsize(16,"points")

%%



figure('name','Chebyshew')

subplot(2,1,1)
plot(time_mesh(2:end-1),alpha_exp_23s,'-*','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh), 'Color',gron,
    ↪ LineWidth=1)
hold on
plot(time_mesh(2:end-1),alpha_exp_Newton,'-o','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh),'Color',orange,
    ↪ LineWidth=1)
hold on
plot(time_mesh(2:end-1),alpha_exp_45,'-S','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh),'Color',lila,LineWidth
    ↪ =1)
hold on
plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--',
    ↪ LineWidth=1)
legend('Explicit beräkning, ode23s','Explicit beräkning, Newton' ,
    ↪ 'Explicit beräkning, ode45','Sannt parametervärde')
title('Förenklad','FontSize',14)
ylim([min_mod-max_mod/3,max_mod*1.3])
xlabel('Dagar','FontSize',12,'FontWeight','bold')
ylabel('Parametervärde','FontSize',12,'FontWeight','bold')
ylim([-5*10^-8 10^-7]) %För paramteter 5


```

```
subplot(2,1,2)
plot(time_mesh(2:end-1),alpha_exp_23s_org,'-*','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh), 'Color',gron,
    ↪ LineWidth=1)
hold on
plot(time_mesh(2:end-1),alpha_exp_Newton_org ,'-o','MarkerSize',12,
    ↪ 'MarkerIndices',1:20:length(time_mesh),'Color',orange,
    ↪ LineWidth=1)
hold on
plot(time_mesh(2:end-1),alpha_exp_45_org,'-S','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh),'Color',lila,LineWidth
    ↪ =1)
hold on
plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--',
    ↪ LineWidth=1)
legend('Explicit beräkning, ode23s','Explicit beräkning, Newton' ,
    ↪ 'Explicit beräkning, ode45','Sannt parametervärde')
title('Original','FontSize',14)
ylim([min_org-max_org/3,max_org*1.3])
xlabel('Dagar','FontSize',12,'FontWeight','bold')
ylabel('Parametervärde','FontSize',12,'FontWeight','bold')
ylim([-5*10^-8 10^-7]) %För paramteter 5

fontsize(16,"points")

%%
figure


plot(time_mesh,x_23s(1,:),'-*','MarkerSize',12,'MarkerIndices'
    ↪ ,1:20:length(time_mesh), 'Color',gron, LineWidth=1);
hold on
plot( time_mesh,x_23s(2,:),'-o','MarkerSize',12,'MarkerIndices'
    ↪ ,1:20:length(time_mesh),'Color',orange,LineWidth=1);
hold on
plot(time_mesh,x_23s(3,:),'-S','MarkerSize',12,'MarkerIndices'
    ↪ ,1:20:length(time_mesh),'Color',lila,LineWidth=1);
hold on
legend('x_T', 'x_{M1}', 'x_{M2}');
xlabel('Dagar','FontSize',12,'FontWeight','bold');
ylabel('Densitet, [celler]','FontSize',12,'FontWeight','bold')

fontsize(16,"points")

%%
function alpha = alpha_vec(dm1,dm2,at1,at2,k12,time_mesh)
scaling_factor_dm1 = dm1;
scaling_factor_dm2 = dm2;
scaling_factor_at1 = at1;
scaling_factor_at2 = at2;
scaling_factor_k12 = k12;

function_flag = 0; % constant

exact_dm1 = ExactParameter(scaling_factor_dm1,function_flag,
    ↪ time_mesh); %Exact profile for dm1 to produce data.
```

```
exact_dm2 = ExactParameter(scaling_factor_dm2,function_flag,
    ↪ time_mesh); %Exact profile for dm2 to produce data.
exact_at1 = ExactParameter(scaling_factor_at1,function_flag,
    ↪ time_mesh); %Exact profile for at1 to produce data.
exact_at2 = ExactParameter(scaling_factor_at2,function_flag,
    ↪ time_mesh); %Exact profile for at2 to produce data.
exact_k12 = ExactParameter(scaling_factor_k12,function_flag,
    ↪ time_mesh); %Exact profile for k12 to produce data.

alpha = [exact_dm1; exact_dm2; exact_at1; exact_at2; exact_k12];

end
```

mod_test_alpha_exp_Chebyshew.m

```
clc; close all; clear all;

t_min=0;t_max=20; m=500; x_initial =[5*10^6; 10^3; 10^3];%För att
    ↪ ha startvärden nära ett steady state sätt in
    ↪ [3.1298490038*10^9; 10^-5; 402531911.894];
time_mesh =[];
for k = 1:m
    time_mesh(end+1) = -(t_max-t_min)/2*cos((2*k-1)*pi/(2*m)) + (
    ↪ t_max+t_min)/2;
end


alpha =  [10^-11 10^-12 10^-10 10^-12 10^-12]*100;
alpha_1=alpha_vec(alpha(1),alpha(2),alpha(3),alpha(4),alpha(5),
    ↪ time_mesh);
x_23s = ForwardODE23s(alpha_1,time_mesh,x_initial);
x_Newton = ForwardNewton(alpha_1,time_mesh,x_initial);
x_45 = ForwardODE45(alpha_1,time_mesh,x_initial);
t_plot = [1:500];


gron = [102,194,165]/255;
orange = [252,141,98]/255;
lila = [141,160,203]/255;

alpha_unknown=4;
alpha_exp_23s=mod_calculate_alpha_exp(alpha,alpha_unknown,x_23s,
    ↪ t_min,t_max);
alpha_exp_Newton=mod_calculate_alpha_exp(alpha,alpha_unknown,
    ↪ x_Newton,t_min,t_max);
alpha_exp_45=mod_calculate_alpha_exp(alpha,alpha_unknown,x_45,t_min
    ↪ ,t_max);

%orginal
alpha_exp_23s_org=calculate_alpha_exp(alpha,alpha_unknown,x_23s,
    ↪ t_min,t_max);
alpha_exp_Newton_org=calculate_alpha_exp(alpha,alpha_unknown,
    ↪ x_Newton,t_min,t_max);
alpha_exp_45_org=calculate_alpha_exp(alpha,alpha_unknown,x_45,t_min
    ↪ ,t_max);
```

```
31
32 max_45_org = max(alpha_exp_45_org);
33 max_Newton_org = max(alpha_exp_Newton_org);
34 max_23s_org = max(alpha_exp_23s_org);
35
36 max_45_mod = max(alpha_exp_45);
37 max_Newton_mod = max(alpha_exp_Newton);
38 max_23s_mod = max(alpha_exp_23s);
39
40
41 max_org = max([max_45_org max_Newton_org max_23s_org]);
42 max_mod = max([max_45_mod max_Newton_mod max_23s_mod]);
43
44 min_45_org = min(alpha_exp_45_org);
45 min_Newton_org = min(alpha_exp_Newton_org);
46 min_23s_org = min(alpha_exp_23s_org);
47
48 min_45_mod= min(alpha_exp_45);
49 min_Newton_mod = min(alpha_exp_Newton);
50 min_23s_mod = min(alpha_exp_23s);
51
52 min_org = min([min_45_org min_Newton_org min_23s_org]);
53 min_mod = min([min_45_mod min_Newton_mod min_23s_mod]);
54
55 figure('name','Chebyshew')
56 subplot(2,2,1)
57 plot(time_mesh(2:end-1),alpha_exp_23s,'color', gron ,LineWidth=2)
58 hold on
59 plot(time_mesh(2:end-1),alpha_exp_Newton, 'color',orange,LineWidth
    ↪ =2)
60 hold on
61 plot(time_mesh(2:end-1),alpha_exp_45, 'color',lila ,LineWidth=2)
62 hold on
63 plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--',
    ↪ LineWidth=1)
64 legend('Explicit beräkning, ode23s','Explicit beräkning, Newton' ,
    ↪ 'Explicit beräkning, ode45','Sannt parametervärde')
65 title('Förenklad')
66 ylim([min_mod-max_mod/3,max_mod*1.3])
67 subplot(2,2,2)
68 plot(time_mesh(2:end-1),log10(alpha_exp_23s), 'color' ,gron,
    ↪ LineWidth=2)
69 hold on
70 plot(time_mesh(2:end-1),log10(alpha_exp_Newton),'color',orange,
    ↪ LineWidth=2)
71 hold on
72 plot(time_mesh(2:end-1),log10(alpha_exp_45),'color',lila,LineWidth
    ↪ =2)
73 hold on
74 plot([0 20],[log10(alpha(alpha_unknown)) log10(alpha(alpha_unknown)
    ↪ )], 'r--',LineWidth=1)
75 legend('Explicit beräkning i logaritmisk skala, ode23s','Explicit
    ↪ beräkning i logaritmisk skala, Newton' , 'Explicit beräkning
    ↪ i logaritmisk skala, ode45','Sannt parametervärde')
76 title('Förenklad, logaritmisk skala')
77
```

```
subplot(2,2,3)
plot(time_mesh(2:end-1),alpha_exp_23s_org,'color',gron,LineWidth=2)
hold on
plot(time_mesh(2:end-1),alpha_exp_Newton_org,'color',orange,
    ↪ LineWidth=2)
hold on
plot(time_mesh(2:end-1),alpha_exp_45_org,'color',lila,LineWidth=2)
hold on
plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--',
    ↪ LineWidth=1)
legend('Explicit beräkning, ode23s','Explicit beräkning, Newton' ,
    ↪ 'Explicit beräkning, ode45','Sannt parametervärde')
title('Original','FontSize',14)
ylim([min_org-max_org/3,max_org*1.3])
xlabel('Dagar','FontSize',12,'FontWeight','bold')
ylabel('Parametervärde','FontSize',12,'FontWeight','bold')

subplot(2,2,4)
plot(time_mesh(2:end-1),log10(alpha_exp_23s_org),'color', gron,
    ↪ LineWidth=2)
hold on
plot(time_mesh(2:end-1),log10(alpha_exp_Newton_org),'color',orange,
    ↪ LineWidth=2)
hold on
plot(time_mesh(2:end-1),log10(alpha_exp_45_org),'color',lila,
    ↪ LineWidth=2)
hold on
plot([0 20],[log10(alpha(alpha_unknown)) log10(alpha(alpha_unknown)
    ↪ )], 'r--',LineWidth=1)
legend('Explicit beräkning i logaritmisk skala, ode23s','Explicit
    ↪ beräkning i logaritmisk skala, Newton' , 'Explicit beräkning
    ↪ i logaritmisk skala, ode45','Sannt parametervärde')
title('Original','FontSize',14)
ylim([min_org-max_org/3,max_org*1.3])
xlabel('Dagar','FontSize',12,'FontWeight','bold')
ylabel('Parametervärde','FontSize',12,'FontWeight','bold')
hold on



%%
figure


plot(time_mesh,x_45(1,:), 'r');
hold on
plot( time_mesh,x_45(2,:),'g');
hold on
plot(time_mesh,x_45(3,:),  'b');
hold on
legend('x_T', 'x_{M1}', 'x_{M2}');
xlabel('Mätpunkt');
ylabel('Densitet')


```

```
%%
function alpha = alpha_vec(dm1,dm2,at1,at2,k12,time_mesh)
scaling_factor_dm1 = dm1;
scaling_factor_dm2 = dm2;
scaling_factor_at1 = at1;
scaling_factor_at2 = at2;
scaling_factor_k12 = k12;

function_flag = 0; % constant

exact_dm1 = ExactParameter(scaling_factor_dm1,function_flag,
    ↪ time_mesh); %Exact profile for dm1 to produce data.
exact_dm2 = ExactParameter(scaling_factor_dm2,function_flag,
    ↪ time_mesh); %Exact profile for dm2 to produce data.
exact_at1 = ExactParameter(scaling_factor_at1,function_flag,
    ↪ time_mesh); %Exact profile for at1 to produce data.
exact_at2 = ExactParameter(scaling_factor_at2,function_flag,
    ↪ time_mesh); %Exact profile for at2 to produce data.
exact_k12 = ExactParameter(scaling_factor_k12,function_flag,
    ↪ time_mesh); %Exact profile for k12 to produce data.

alpha = [exact_dm1; exact_dm2; exact_at1; exact_at2; exact_k12];

end
```

mod_test_alpha_exp_delar.m

```
clc; close all; clear all;

t_min=0;t_max=20; m=500; x_initial =[5*10^6; 10^3; 10^3];%Om startv
    ↪ ärden ska ligga nära ett steady state, sätt in
    ↪ [3.1298490038*10^9; 10^-5; 402531911.894];
time_mesh=linspace(t_min,t_max,m);
alpha =  [10^-11 10^-12 10^-10 10^-12 10^-12]*100;
alpha_1=alpha_vec(alpha(1),alpha(2),alpha(3),alpha(4),alpha(5),
    ↪ time_mesh);
x_23s = ForwardODE23s(alpha_1,time_mesh,x_initial);
x_Newton = ForwardNewton(alpha_1,time_mesh,x_initial);
x_45 = ForwardODE45(alpha_1,time_mesh,x_initial);
t_plot = [1:500];


gron = [102,194,165]/255;
orange = [252,141,98]/255;
lila = [141,160,203]/255;

alpha_unknown=5;
%modifierad, hela
alpha_exp_23s=mod_calculate_alpha_exp(alpha,alpha_unknown,x_23s,
    ↪ t_min,t_max);
alpha_exp_Newton=mod_calculate_alpha_exp(alpha,alpha_unknown,
    ↪ x_Newton,t_min,t_max);
alpha_exp_45=mod_calculate_alpha_exp(alpha,alpha_unknown,x_45,t_min
    ↪ ,t_max);

```

```
23
24 %modifierad, del 1 dvs  dm2*xM2/xM1 är borttaget
25 alpha_exp_23s_del1=mod_calculate_alpha_exp_del1(alpha,alpha_unknown
   ↪ ,x_23s,t_min,t_max);
26 alpha_exp_Newton_del1=mod_calculate_alpha_exp_del1(alpha,
   ↪ alpha_unknown,x_Newton,t_min,t_max);
27 alpha_exp_45_del1=mod_calculate_alpha_exp_del1(alpha,alpha_unknown,
   ↪ x_45,t_min,t_max);
28
29 %modifierad, del 2
30 alpha_exp_23s_del2=mod_calculate_alpha_exp_del2(alpha,alpha_unknown
   ↪ ,x_23s,t_min,t_max);
31 alpha_exp_Newton_del2=mod_calculate_alpha_exp_del2(alpha,
   ↪ alpha_unknown,x_Newton,t_min,t_max);
32 alpha_exp_45_del2=mod_calculate_alpha_exp_del2(alpha,alpha_unknown,
   ↪ x_45,t_min,t_max);
33
34 %modifierad, del 3
35 alpha_exp_23s_del3=mod_calculate_alpha_exp_del3(alpha,alpha_unknown
   ↪ ,x_23s,t_min,t_max);
36 alpha_exp_Newton_del3=mod_calculate_alpha_exp_del3(alpha,
   ↪ alpha_unknown,x_Newton,t_min,t_max);
37 alpha_exp_45_del3=mod_calculate_alpha_exp_del3(alpha,alpha_unknown,
   ↪ x_45,t_min,t_max);
38
39 figure
40 subplot(2,1,1)
41 plot(time_mesh(2:end-1),alpha_exp_23s,'-*','MarkerSize',12,'
   ↪ MarkerIndices',1:20:length(time_mesh), 'Color',gron,
   ↪ LineWidth=1)
42 hold on
43 plot(time_mesh(2:end-1),alpha_exp_Newton,'-o','MarkerSize',12,'
   ↪ MarkerIndices',1:20:length(time_mesh),'Color',orange,
   ↪ LineWidth=1)
44 hold on
45 plot(time_mesh(2:end-1),alpha_exp_45,'-S','MarkerSize',12,'
   ↪ MarkerIndices',1:20:length(time_mesh),'Color',lila,LineWidth
   ↪ =1)
46 hold on
47 plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--',
   ↪ LineWidth=1)
48 legend('Explicit calculation, ode23s','Explicit calculation, Newton
   ↪ ' , 'Explicit calculation, ode45','True value of parameter')
49 title('Förenklad, hela')
50 xlabel('Dagar','FontSize',12,'FontWeight','bold');
51 ylabel('Parametervärde','FontSize',12,'FontWeight','bold')
52
53
54 subplot(2,1,2)
55 plot(time_mesh(2:end-1),alpha_exp_23s_del1,'-*','MarkerSize',12,'
   ↪ MarkerIndices',1:20:length(time_mesh), 'Color',gron,
   ↪ LineWidth=1)
56 hold on
57 plot(time_mesh(2:end-1),alpha_exp_Newton_del1,'-o','MarkerSize',12,
   ↪ 'MarkerIndices',1:20:length(time_mesh),'Color',orange,
   ↪ LineWidth=1)
```

```
hold on
plot(time_mesh(2:end-1),alpha_exp_45_del1,'-S','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh),'Color',lila,LineWidth
    ↪ =1)
hold on
plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--',
    ↪ LineWidth=1)
legend('Explicit calculation, ode23s','Explicit calculation, Newton
    ↪ ' , 'Explicit calculation, ode45','True value of parameter')
title('Förenklad, del 1 borta')
xlabel('Dagar','FontSize',12,'FontWeight','bold');
ylabel('Parametervärde','FontSize',12,'FontWeight','bold')
fontsize(16,"points")


figure
subplot(2,1,1)
plot(time_mesh(2:end-1),alpha_exp_23s_del2,'-*','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh), 'Color',gron,
    ↪ LineWidth=1)
hold on
plot(time_mesh(2:end-1),alpha_exp_Newton_del2,'-o','MarkerSize',12,
    ↪ 'MarkerIndices',1:20:length(time_mesh),'Color',orange,
    ↪ LineWidth=1)
hold on
plot(time_mesh(2:end-1),alpha_exp_45_del2,'-S','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh),'Color',lila,LineWidth
    ↪ =1)
hold on
plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--',
    ↪ LineWidth=1)
legend('Explicit calculation, ode23s','Explicit calculation, Newton
    ↪ ' , 'Explicit calculation, ode45','True value of parameter')
title('Förenklad, del 2  borta')
xlabel('Dagar','FontSize',12,'FontWeight','bold');
ylabel('Parametervärde','FontSize',12,'FontWeight','bold')

subplot(2,1,2)
plot(time_mesh(2:end-1),alpha_exp_23s_del3,'-*','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh), 'Color',gron,
    ↪ LineWidth=1)
hold on
plot(time_mesh(2:end-1),alpha_exp_Newton_del3,'-o','MarkerSize',12,
    ↪ 'MarkerIndices',1:20:length(time_mesh),'Color',orange,
    ↪ LineWidth=1)
hold on
plot(time_mesh(2:end-1),alpha_exp_45_del3,'-S','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh),'Color',lila,LineWidth
    ↪ =1)
hold on
plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--',
    ↪ LineWidth=1)
legend('Explicit calculation, ode23s','Explicit calculation, Newton
    ↪ ' , 'Explicit calculation, ode45','True value of parameter')
title('Förenklad, del 3  borta')
xlabel('Dagar','FontSize',12,'FontWeight','bold');
```

```
ylabel('Parametervärde','FontSize',12,'FontWeight','bold')

hold on

fontsize(16,"points")
%%
figure
plot(time_mesh,x_45(1,:), 'r');
hold on
plot( time_mesh,x_45(2,:),'g');
hold on
plot(time_mesh,x_45(3,:),  'b');
hold on
legend('x_T', 'x_{M1}', 'x_{M2}');
xlabel('Mätpunkt');
ylabel('Densitet')


%%
function alpha = alpha_vec(dm1,dm2,at1,at2,k12,time_mesh)
scaling_factor_dm1 = dm1;
scaling_factor_dm2 = dm2;
scaling_factor_at1 = at1;
scaling_factor_at2 = at2;
scaling_factor_k12 = k12;

function_flag = 0; % constant

exact_dm1 = ExactParameter(scaling_factor_dm1,function_flag,
    ↪ time_mesh); %Exact profile for dm1 to produce data.
exact_dm2 = ExactParameter(scaling_factor_dm2,function_flag,
    ↪ time_mesh); %Exact profile for dm2 to produce data.
exact_at1 = ExactParameter(scaling_factor_at1,function_flag,
    ↪ time_mesh); %Exact profile for at1 to produce data.
exact_at2 = ExactParameter(scaling_factor_at2,function_flag,
    ↪ time_mesh); %Exact profile for at2 to produce data.
exact_k12 = ExactParameter(scaling_factor_k12,function_flag,
    ↪ time_mesh); %Exact profile for k12 to produce data.

alpha = [exact_dm1; exact_dm2; exact_at1; exact_at2; exact_k12];

end
```

mod_test_alpha_exp.m

```
clc; close all; clear all;

t_min=0;t_max=20; m=500; x_initial =[5*10^6; 10^3; 10^3];%För att
    ↪ ha startvärden nära ett steady state sätt in
    ↪ [3.1298490038*10^9; 10^-5; 402531911.894];
time_mesh=linspace(t_min,t_max,m);
alpha =  [10^-11 10^-12 10^-10 10^-12 10^-12]*100;
alpha_1=alpha_vec(alpha(1),alpha(2),alpha(3),alpha(4),alpha(5),
    ↪ time_mesh);
x_23s = ForwardODE23s(alpha_1,time_mesh,x_initial);
```

```
x_Newton = ForwardNewton(alpha_1,time_mesh,x_initial);
x_45 = ForwardODE45(alpha_1,time_mesh,x_initial);
t_plot = [1:500];

gron = [102,194,165]/255;
orange = [252,141,98]/255;
lila = [141,160,203]/255;

alpha_unknown=1;
alpha_exp_23s=mod_calculate_alpha_exp(alpha,alpha_unknown,x_23s,
    ↪ t_min,t_max);
alpha_exp_Newton=mod_calculate_alpha_exp(alpha,alpha_unknown,
    ↪ x_Newton,t_min,t_max);
alpha_exp_45=mod_calculate_alpha_exp(alpha,alpha_unknown,x_45,t_min
    ↪ ,t_max);

%orginal
alpha_exp_23s_org=calculate_alpha_exp(alpha,alpha_unknown,x_23s,
    ↪ t_min,t_max);
alpha_exp_Newton_org=calculate_alpha_exp(alpha,alpha_unknown,
    ↪ x_Newton,t_min,t_max);
alpha_exp_45_org=calculate_alpha_exp(alpha,alpha_unknown,x_45,t_min
    ↪ ,t_max);


figure('name','Vanlig')
subplot(2,1,1)
plot(time_mesh(2:end-1),alpha_exp_23s,'-*','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh), 'Color',gron,
    ↪ LineWidth=1)
hold on
plot(time_mesh(2:end-1),alpha_exp_Newton,'-o','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh),'Color',orange,
    ↪ LineWidth=1)
hold on
plot(time_mesh(2:end-1),alpha_exp_45, '-S','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh),'Color',lila,LineWidth
    ↪ =1)
hold on
plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--',
    ↪ LineWidth=1)
legend('Explicit beräkning, ode23s','Explicit beräkning, Newton' ,
    ↪ 'Explicit beräkning, ode45','Sannt parametervärde')
title('Förenklad')

%subplot(2,2,2)
%plot(time_mesh(2:end-1),log10(alpha_exp_23s), 'color',gron,
    ↪ LineWidth=1.5)
%hold on
%plot(time_mesh(2:end-1),log10(alpha_exp_Newton), 'color',orange,
    ↪ LineWidth=1.5)
%hold on
%plot(time_mesh(2:end-1),log10(alpha_exp_45), 'color',lila,
    ↪ LineWidth=1.5)
%hold on
%plot([0 20],[log10(alpha(alpha_unknown)) log10(alpha(alpha_unknown
```

```
    ↪ ))], 'r--')
%legend('Logarithm of explicit calculation, ode23s','Logarithm of
    ↪ explicit calculation, Newton' , 'Logarithm of explicit
    ↪ calculation, ode45','True value of parameter')
%title('Modifierad, logaritmisk skala')

subplot(2,1,2)
plot(time_mesh(2:end-1),alpha_exp_23s_org, '-*','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh), 'Color',gron,
    ↪ LineWidth=1)
hold on
hold on
plot(time_mesh(2:end-1),alpha_exp_Newton_org,'-o','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh),'Color',orange,
    ↪ LineWidth=1)
hold on
plot(time_mesh(2:end-1),alpha_exp_45_org, '-S','MarkerSize',12,'
    ↪ MarkerIndices',1:20:length(time_mesh),'Color',lila,LineWidth
    ↪ =1)
hold on
plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--',
    ↪ LineWidth=1)
legend('Explicit beräkning, ode23s','Explicit beräkning, Newton' ,
    ↪ 'Explicit beräkning, ode45','Sannt parametervärde')
title('Original','FontSize',14)
xlabel('Dagar','FontSize',12,'FontWeight','bold')
ylabel('Parametervärde','FontSize',12,'FontWeight','bold')


fontsize(16,"points")

%subplot(2,2,4)
%plot(time_mesh(2:end-1),log10(alpha_exp_23s_org), 'color',gron,
    ↪ LineWidth=1.5)
%hold on
%plot(time_mesh(2:end-1),log10(alpha_exp_Newton_org), 'color',
    ↪ orange,LineWidth=1.5)
%hold on
%plot(time_mesh(2:end-1),log10(alpha_exp_45_org), 'color',lila,
    ↪ LineWidth=1.5)
%hold on
%plot([0 20],[log10(alpha(alpha_unknown)) log10(alpha(alpha_unknown
    ↪ ))], 'r--')
%legend('Logarithm of explicit calculation, ode23s','Logarithm of
    ↪ explicit calculation, Newton' , 'Logarithm of explicit
    ↪ calculation, ode45','True value of parameter')
%title('Orginal, logaritmisk skala')

%%
figure


plot(time_mesh,x_23s(1,:),'-*','MarkerSize',12,'MarkerIndices'
    ↪ ,1:20:length(time_mesh), 'Color',gron, LineWidth=1);
hold on
plot( time_mesh,x_23s(2,:),'-o','MarkerSize',12,'MarkerIndices'
```

```
    ↪ ,1:20:length(time_mesh),'Color',orange,LineWidth=1);
hold on
plot(time_mesh,x_23s(3,:),'-S','MarkerSize',12,'MarkerIndices'
    ↪ ,1:20:length(time_mesh),'Color',lila,LineWidth=1);
hold on
legend('x_T', 'x_{M1}', 'x_{M2}');
xlabel('Dagar','FontSize',12,'FontWeight','bold');
ylabel('Densitet','FontSize',12,'FontWeight','bold')

fontsize(16,"points")



%%
function alpha = alpha_vec(dm1,dm2,at1,at2,k12,time_mesh)
scaling_factor_dm1 = dm1;
scaling_factor_dm2 = dm2;
scaling_factor_at1 = at1;
scaling_factor_at2 = at2;
scaling_factor_k12 = k12;

function_flag = 0; % constant

exact_dm1 = ExactParameter(scaling_factor_dm1,function_flag,
    ↪ time_mesh); %Exact profile for dm1 to produce data.
exact_dm2 = ExactParameter(scaling_factor_dm2,function_flag,
    ↪ time_mesh); %Exact profile for dm2 to produce data.
exact_at1 = ExactParameter(scaling_factor_at1,function_flag,
    ↪ time_mesh); %Exact profile for at1 to produce data.
exact_at2 = ExactParameter(scaling_factor_at2,function_flag,
    ↪ time_mesh); %Exact profile for at2 to produce data.
exact_k12 = ExactParameter(scaling_factor_k12,function_flag,
    ↪ time_mesh); %Exact profile for k12 to produce data.

alpha = [exact_dm1; exact_dm2; exact_at1; exact_at2; exact_k12];


end
```

mod_test1_alpha_exp_ode23s.m

```
clc; close all; clear all;

t_min=0;t_max=20; m=500; x_initial =[5*10^6; 10^3; 10^3];%För att
    ↪ ha startvärden nära ett steady state sätt in
    ↪ [3.1298490038*10^9; 10^-5; 402531911.894];
time_mesh=linspace(t_min,t_max,m);
alpha =  [10^-11 10^-12 10^-10 10^-12 10^-12]*100;  %[1,1,1,1,1];%
alpha_1=alpha_vec(alpha(1),alpha(2),alpha(3),alpha(4),alpha(5),
    ↪ time_mesh);
x_23s = ForwardODE23s(alpha_1,time_mesh,x_initial);
t_plot = [1:500];

gron = [102,194,165]/255;
orange = [252,141,98]/255;
```

```
lila = [141,160,203]/255;

alpha_unknown=2;

%modifierad
alpha_exp_23s=mod_calculate_alpha_exp(alpha,alpha_unknown,x_23s,
    ↪ t_min,t_max);
alpha_exp_23s_T = alpha_exp_23s';
alpha_mod_medel = mean(alpha_exp_23s_T([250:498]))


%orginal
alpha_exp_23s_org=calculate_alpha_exp(alpha,alpha_unknown,x_23s,
    ↪ t_min,t_max);
alpha_exp_23s_org_T = alpha_exp_23s_org';
alpha_mod_medel = mean(alpha_exp_23s_org_T([250:498]))

figure('name','Vanlig')
subplot(2,1,1)
plot(time_mesh(2:end-1),alpha_exp_23s,'color', gron,LineWidth=2)
hold on
plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--',
    ↪ LineWidth=1)
legend('Explicit beräkningar, ode23s','Sannt paramtetervärde')
title('Förenklad','FontSize',14)
xlabel('Dagar','FontSize',12,'FontWeight','bold')
ylabel('Parametervärde','FontSize',12,'FontWeight','bold')


subplot(2,1,2)
plot(time_mesh(2:end-1),alpha_exp_23s_org, 'color',gron,LineWidth
    ↪ =2)
hold on
plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--',
    ↪ LineWidth=1)
legend('Explicit beräkningar, ode23s','Sannt paramtetervärde')
title('Original','FontSize',14)
xlabel('Dagar','FontSize',12,'FontWeight','bold')
ylabel('Parametervärde','FontSize',12,'FontWeight','bold')

fontsize(16,"points")

%%
figure


plot(time_mesh,x_23s(1,:),'-*','MarkerSize',12,'MarkerIndices'
    ↪ ,1:20:length(time_mesh), 'Color',gron, LineWidth=1);
hold on
plot( time_mesh,x_23s(2,:),'-o','MarkerSize',12,'MarkerIndices'
    ↪ ,1:20:length(time_mesh),'Color',orange,LineWidth=1);
hold on
plot(time_mesh,x_23s(3,:),'-S','MarkerSize',12,'MarkerIndices'
    ↪ ,1:20:length(time_mesh),'Color',lila,LineWidth=1);
hold on
legend('x_T', 'x_{M1}', 'x_{M2}');
```

```
xlabel('Dagar','FontSize',12,'FontWeight','bold');
ylabel('Densitet, [celler]','FontSize',12,'FontWeight','bold')

fontsize(16,"points")
%%
function alpha = alpha_vec(dm1,dm2,at1,at2,k12,time_mesh)
scaling_factor_dm1 = dm1;
scaling_factor_dm2 = dm2;
scaling_factor_at1 = at1;
scaling_factor_at2 = at2;
scaling_factor_k12 = k12;

function_flag = 0; % constant

exact_dm1 = ExactParameter(scaling_factor_dm1,function_flag,
    ↪ time_mesh); %Exact profile for dm1 to produce data.
exact_dm2 = ExactParameter(scaling_factor_dm2,function_flag,
    ↪ time_mesh); %Exact profile for dm2 to produce data.
exact_at1 = ExactParameter(scaling_factor_at1,function_flag,
    ↪ time_mesh); %Exact profile for at1 to produce data.
exact_at2 = ExactParameter(scaling_factor_at2,function_flag,
    ↪ time_mesh); %Exact profile for at2 to produce data.
exact_k12 = ExactParameter(scaling_factor_k12,function_flag,
    ↪ time_mesh); %Exact profile for k12 to produce data.

alpha = [exact_dm1; exact_dm2; exact_at1; exact_at2; exact_k12];
end
```

solver_adapt.m

```
%% Specify fixed parameter values.
clear all
close all
clc
%Time parameters
    ↪ ------------------------------------------------------------
final_time = 20;
%obs_start = 100;            %Starting time for observations of
    ↪ true solution.
obs_start = 0;
obs_end = 20;            %End time for observations of true
    ↪ solution.
%Parameters from the model problem
    ↪ ---------------------------------------


dm1=10^-11;
dm2 =10^-12;
at1 =10^-10;
at2 =10^-12;
k12=10^-12;

%Optimization parameters
    ↪ ------------------------------------------------
gamma_start =10^(-11);         %Initial value of regularization
```

```
    ↪ parameter.
opt_step_length = 0.001;  %Step length in optimization algorithm.
optim_maxiter = 500;      %Maximum iterations of optimization alg.
%Noise
    ↪ ------------------------------------------------------------------------
    ↪
noise_level = 0.01;   %Noise level of observations.
%noise_level = 0.1;   %Noise level of observations.
noise_flag = 1;       % 0 = no noise, 1 = random noise, 2 =
    ↪ additive noise.

% Create initial time mesh, observations and starting guess for
    ↪ parameter dm1.
%time_mesh = 0:final_time; % will be adaptively updated.
m=15;  %number of observations
time_mesh =linspace(0,final_time ,m);

refined = time_mesh;


%Values for   scaling factors for parameters
    ↪ ------------------------------------------------

scaling_factor_dm1 = 10^-11;
scaling_factor_dm2 = 10^-12;
scaling_factor_at1 = 10^-10;
scaling_factor_at2 = 10^-12;
scaling_factor_k12 = 10^-12;


function_flag = 0;       %dm1(t) = scaling_factor*function
%function flag can be between 1 and 4, determines model function
    ↪ which should be reconstructed
%Function flag: 0. const. 1. +linear 2. -linear 3. sin 4. exp(-x)
%NB! For now, we will specify these values inside the time-adaptive
    ↪  loop

%instad.
%They will be moved back here at a later stage.
%[dm1_guess,g] = AnalyticDm1(exact_dm1,time_mesh,obs_start,obs_end,
    ↪ noise_flag,noise_level);
%Input: 1. exact dm1 2. time mesh 3. observation start 4.
    ↪ observation end. 5. noise_flag. 6. noise level

% initials for forward problem
x_initial = [5*10^6; 10^3; 10^3];

%===============  values of exact constant parameters

eps = 10^(-7);
% Run algorithm
    figure
for meshref = 1:10  %Outer loop is for time adaptivity, will be
    ↪ replaced with while loop later.

    time_mesh = refined;                                          %
```

```
    ↪ Our new time mesh (if mesh refinement has been made.
66
67      nodes = length(time_mesh);
68      %res_time= 0.1*ones(1,nodes);
        ↪
69
70
71      exact_dm1 = ExactParameter(scaling_factor_dm1,function_flag,
        ↪ time_mesh); %Exact profile for dm1 to produce data.
72      exact_dm2 = ExactParameter(scaling_factor_dm2,function_flag,
        ↪ time_mesh); %Exact profile for dm2 to produce data.
73      exact_at1 = ExactParameter(scaling_factor_at1,function_flag,
        ↪ time_mesh); %Exact profile for at1 to produce data.
74      exact_at2 = ExactParameter(scaling_factor_at2,function_flag,
        ↪ time_mesh); %Exact profile for at2 to produce data.
75      exact_k12 = ExactParameter(scaling_factor_k12,function_flag,
        ↪ time_mesh); %Exact profile for k12 to produce data.
76
77      alpha = [exact_dm1; exact_dm2; exact_at1; exact_at2; exact_k12
        ↪ ];
78
79      %Observations and first guess for dm1.
        ↪
80      [dm1_guess,g] = AnalyticParameter(alpha, time_mesh,obs_start,
        ↪ obs_end,noise_flag,noise_level,x_initial);
81
82      %plot(time_mesh,dm1_guess,'o')
83      %obtain initial guess for gradient method using method of
        ↪ normal equations
84
85       %use polynomial of degree 5 to recover noisy data via method
        ↪ of normal equations and obtain initial guess
86       %dm1_guess=LinearClassAdapt(dm1_guess, time_mesh,m);
87      dm1_guess = LinearClassNormalEqExample2(dm1_guess,time_mesh,m);
88      dm1 = zeros(optim_maxiter,nodes);                              %
        ↪ Preallocation of dm1 (for use in for-loop).
89
90
91      dm1(1,:) = dm1_guess;                                          %
        ↪ First guess for dm1 in first row.
92      beta = opt_step_length*ones(1,nodes);                          %
93      gamma = gamma_start;                                           %
        ↪ Restore beta and gamma.
94      big_grad = zeros(1,nodes);                                     %
        ↪ Preallocation of vector for adaptivity.
95      u1 = ForwardODE45(alpha,time_mesh,x_initial);
        ↪    %Compute initial forward sol.
96      lambda1 = AdjointODE45(alpha,u1,g,time_mesh,obs_start,obs_end);
        ↪    %Compute adjoint sol.
97
98      % u1(1,:) = X_T; u1(2,:) = X_M1; u1(3,:) = X_M2;
99
100     g0 = (lambda1(1,:).*u1(1,:).*u1(2,:));            %Compute grad
        ↪  with respect to dm1  without reg.term
101
102
```

```
103     grad_hist = zeros(optim_maxiter,nodes);
104     grad_hist(1,:) = g0;                                                %
    ↪ Save gradient for later plot.
105     if norm(g0) < 1000
106         dm1(1:end,:) = ones(length(1:optim_maxiter),1)*dm1(1,:);
    ↪  %Check if gradient is already small
107         disp('The algorithm has converged at first step!')
108         break
109     end
110     bm = 1/gamma;                                                       %
    ↪ For Conjugate Gradient algorithm.
111     gn = -g0;
112     dn = gn;
113     g0 = sign(g0);
    ↪  %Normalize gradient.
114     dm1(2,:) = dm1(1,:) + beta.*g0;
    ↪  %Compute new dm1.
115     iter = 0;
116     for i = 2:optim_maxiter-1
    ↪  %Here we start calculating dm1.
117         gamma = gamma/i^0.5;
    ↪  %Update gamma.
118         u = ForwardODE45(alpha,time_mesh,x_initial);
    ↪          %Update forward and adjoint sol.
119         lambda = AdjointODE45(alpha,u,g,time_mesh,obs_start,obs_end
    ↪ );
120
121        % update gradient for dm1  with reg.term: gamma is reg.
    ↪ parameter
122         gm = gamma*(dm1(i,:)-dm1(1,:)) + lambda1(1,:).*u1(1,:).*u1
    ↪ (2,:);
123
124         bs = (norm(gm)/norm(gn))^2;
    ↪  %For CG.
125         dm = -gm + bs*dn;
126         bm = -(gm*dn')/(gamma*(dn*dn'));
    ↪  %beta = -<g,d>/gamma<d,d>
127         grad_hist(i,:) = gm;
    ↪  %Save gradient for later plot.
128         if norm(gm) < 0.001
    ↪  %Check if gradient is small enough.
129             dm1(i+1:end,:) = ones(length(i+1:optim_maxiter),1)*dm1(
    ↪ i,:);
130             grad_hist(i+1:end,:) = ones(length(i+1:optim_maxiter)
    ↪ ,1)*grad_hist(i,:);
131             disp('The algorithm has converged!')
132             break
133         elseif 0.999999*norm(dm1(i-1,:)) < norm(dm1(i,:)) && norm(
    ↪ dm1(i,:)) < 1.000001*norm(dm1(i-1,:))
134             dm1(i+1:end,:) = ones(length(i+1:optim_maxiter),1)*dm1(
    ↪ i,:);
135             grad_hist(i+1:end,:) = ones(length(i+1:optim_maxiter)
    ↪ ,1)*grad_hist(i,:);
136             disp('Dm1 does not change!')
    ↪  %Also terminate if dm1 stabilizes.
137             break
```

```
138         %elseif norm(grad_hist(i-1,:)) < norm(grad_hist(i,:))
139         %    dm1(i+1,:) = dm1(i,:) + 0.5*bdm1.*sign(grad_hist(i,:))
    ↪ ;
140         %    dm1(i+2:end,:) = ones(length(i+2:optim_maxiter),1)*dm1
    ↪ (i+1,:);
141         %    disp('Gradient grows.')
142         %    break
143         %    %Termination if the gradient increases turned out to
    ↪ give
144         %    worse results, and is therefore disabled.
145         else
146             gm = sign(gm);                                          %
    ↪ Otherwise, continue with line search.
147             for j = 1:nodes
148                 if dm1(i,j) < 10^-9
    ↪  %Force dm1=0 if computed dm1 is negative.
149                     dm1(i+1,j) = 10^-9;
150                 elseif dm1(i,j) > 10^-3
151                     dm1(i+1,j) = 10^-3;
152                 elseif bm < 0.01
153                     dm1(i+1,j) = dm1(i,j) + bm*gm(j);           %
    ↪ Use conjugate gradient if appropriate.
154                 else
155                     if gm(j) == -gn(j)                          %
    ↪ Otherwise use fixed step length.
156                         %bdm1(j) = beta(j)/2;                    %
    ↪ If gradient change sign, halve step-length.
157                         beta(j) = beta(j)/2;
158                     end
159                     dm1(i+1,j) = dm1(i,j) - beta(j)*gm(j);      %
    ↪ Compute new dm1 closer to the actual solution.
160                 end
161             end
162         end
163         gn = gm;                %Save gradient for use in the CG
    ↪ algorithm.
164         dn = dm;
165         iter = iter + 1;
166     end
167
168
169     iter;
170     max_grad = max(abs(grad_hist(end,:)));
171     big_grad = abs(grad_hist(end,:)) > 0.2*max_grad;
172
173     refined = [];               %Start refinement of time mesh ...
174     refining = [];
175     k = 1;
176     r = 1;
177     while r < length(big_grad)
178         if big_grad(r+1) == 1 && big_grad(r) == 0
179             refined = [refined,time_mesh(k:r)];
180         end
181         if big_grad(r) == 0
182             r = r + 1;
183             if r == length(big_grad)
```

```
                refined = [refined,time_mesh(k:r)];
            end
            continue
        else
            %j = i;
            while big_grad(r) == 1 && r < length(time_mesh)
                refining = [refining,time_mesh(r),(time_mesh(r)+
↪ time_mesh(r+1))/2];
                r = r + 1;
                if r == length(time_mesh)
                    refining = [refining,time_mesh(r)];
                end
            end
            refined = [refined, refining];
            k = r;
        end
        refining = [];
    end

    dm1_final = dm1(end,:);


    %relativeerror = norm((dm1(end,:)-exact_dm1)/exact_dm1)/m;
    relativeerror = norm((dm1(end,:)-exact_dm1)/exact_dm1);
    reler(meshref) = relativeerror;



    %if (meshref > 1 & relativeerror < reler(meshref-1) )
    figure

    plot(time_mesh,exact_dm1,'b','LineWidth',3)
    hold on
    plot(time_mesh,dm1_final,'-*','LineWidth',1)

    title(['Calculated dm1, number of refinements: ',num2str(
↪ meshref),'number of points',num2str(m)])
    legend('exact dm1', 'computed dm1 (CGM)')

    hold off



    %figure(2)

    %plot(time_mesh,exact_dm1,'b','LineWidth',3)

    %hold on


    % plot(time_mesh,dm1_final,'-*','LineWidth',1)

    %    title(['Calculated \dm1, number of refinements: ',num2str(
↪ meshref),'number of  points',num2str(m)])
    %      legend('exact \dm1','computed \dm1')

```

```
237
238     %    hold off
239
240
241     meshref
242       %  plot(time_mesh, dm1(end,:)) %Plot calculated dm1 for each
    ↪ iteration in time adaptive algorithm.
243
244     %end
245     reler
246         %include time partitioning
247     %% Visualize   computed dm1
248     %figure
249     %surf(dm1)
250     %xlabel('time partition');
251     %ylabel('iteration');
252     %zlabel('dm1');
253     %title('Computed drug efficiency, dm1');
254
255
256     % presentation of the computed gradient
257     %figure
258     %surf(grad_hist)
259     %xlabel('time partition');
260     %ylabel('iteration');
261     %zlabel('gradient');
262     %title('Gradient of the functional');
263
264
265
266
267     if ( relativeerror < eps)
268                   break
269     end
270
271
272       if (meshref > 1 & relativeerror >  reler(meshref-1) )
273                       break
274                   end
275
276     %% Apply least-squares smoothing on the solution
277
278     %dm1_final = dm1(end,:);
279     N = length(dm1_final);
280     e = ones(N,1);
281     D = spdiags([e -2*e e], 0:2, N-2, N);
282     F = speye(N) + 50*(D'*D);
283     dm1_smooth = F\dm1_final';
284
285     figure
286     plot(time_mesh,exact_dm1,'LineWidth',2)
287
288                hold on
289     plot(time_mesh,dm1_guess,'LineWidth',2)
290
291     plot(time_mesh,dm1_smooth,'LineWidth',2)
```

```
                legend('exact dm1','guess for dm1 (LS)','computed dm1 (smoothed CGM)')



    %   plot(time_mesh,dm1_final,'LineWidth',2)
    %           legend('exact \dm1','guess for \dm1','computed \dm1 ')


                xlabel('Time')


    title(['Calculated dm1, number of refinements: ',num2str(meshref),'number of  points',num2str(m)])

    hold off
end
```

test_alpha_exp.m

```
%%
clc, clf

t_min=0;t_max=20; m=5000; x_initial = [5*10^6; 10^3; 10^3];
time_mesh=linspace(t_min,t_max,m);
alpha = [10^-11 10^-12 10^-10 10^-12 10^-12]*100;
alpha_1=alpha_vec(alpha(1),alpha(2),alpha(3),alpha(4),alpha(5),time_mesh);
x = ForwardODE23s(alpha_1,time_mesh,x_initial);
%x = ForwardNewton(alpha_1,time_mesh,x_initial);
%x = ForwardODE45(alpha_1,time_mesh,x_initial);
t_plot = linspace(0,20,5000);


alpha_unknown=1;
alpha_exp=calculate_alpha_exp(alpha,alpha_unknown,x,t_min,t_max);

%Ger vanlig skala
%plot(t_plot(2:end-1),alpha_exp,LineWidth=1.5)
%hold on
%plot([0 20],[alpha(alpha_unknown) alpha(alpha_unknown)], 'r--')
%legend('Explicit beräkning','Sanna parametervärdet')

%Ger logaritmisk skala
plot(t_plot(2:end-1),log10(alpha_exp),LineWidth=1.5)
hold on
plot([0 20],[log10(alpha(alpha_unknown)) log10(alpha(alpha_unknown))], 'r--')
legend('Logaritmiska värdet av den explicita beräkningen','Sanna parametervärdet')

%%
function alpha = alpha_vec(dm1,dm2,at1,at2,k12,time_mesh)
scaling_factor_dm1 = dm1;
```

```
scaling_factor_dm2 = dm2;
scaling_factor_at1 = at1;
scaling_factor_at2 = at2;
scaling_factor_k12 = k12;

function_flag = 0; % constant

exact_dm1 = ExactParameter(scaling_factor_dm1,function_flag,
    ↪ time_mesh); %Exact profile for dm1 to produce data.
exact_dm2 = ExactParameter(scaling_factor_dm2,function_flag,
    ↪ time_mesh); %Exact profile for dm2 to produce data.
exact_at1 = ExactParameter(scaling_factor_at1,function_flag,
    ↪ time_mesh); %Exact profile for at1 to produce data.
exact_at2 = ExactParameter(scaling_factor_at2,function_flag,
    ↪ time_mesh); %Exact profile for at2 to produce data.
exact_k12 = ExactParameter(scaling_factor_k12,function_flag,
    ↪ time_mesh); %Exact profile for k12 to produce data.

alpha = [exact_dm1; exact_dm2; exact_at1; exact_at2; exact_k12];

end
```

test.m

```
%%  Initials
clc; close all; clear all;
%%initial of x and time variables
m = 15; % number of observations
obs_start = 2.1; obs_end = 7; %interval of observations
time_final = 20;

time_mesh = linspace(0,time_final,m);
% x_initial = [x_T(0); x_M1(0); x_M2(0)]
% initial values that seems to fit fig 1a
x_initial = [5*10^6; 10^3; 10^3];


%% observations
alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh);
noise_level = 0.1;% 10% noise

%using ode45 since newton seems to have problems with low numbers
    ↪ for
% this particular system of odes

[g, g_brus1, g_add] = ExactODE45(alpha,time_mesh,noise_level,
    ↪ x_initial);
figure

plot(time_mesh,g(1,:),'linewidth',2)
hold on
plot(time_mesh,g_brus1(1,:),'*')
  legend('x_T, ode45','observations g1')
  title(['ODE45 versus noisy data, noise , \delta= ',num2str(
    ↪ noise_level)]);
```

```
29
30
31 %% Test solver quality : Forward
32 figure
33 hold on
34 sch = 0;
35 for m2 = [15 20 50 100 500 1000]
36     time_mesh2 = linspace(0,time_final,m2);
37     alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh2
   ↪ );
38     FN = ForwardNewton(alpha,time_mesh2,x_initial);
39     F45 = ForwardODE45(alpha,time_mesh2,x_initial);
40     subtitle('ODE45 versus Newton  for forward problem');
41 sch = sch+1;
42 subplot(2, 3, sch);
43 plot(time_mesh2,FN(1,:),'-r',time_mesh2,F45(1,:),'--b','linewidth'
   ↪ ,2);
44 subtitle('ODE45 versus Newton  for forward problem');
45 xlabel(m2)
46
47 legend('x_T Newton', 'x_T ode45')
48 %     text(time_mesh2(end),FN(1,end),['N', num2str(m2)])
49 %     text(time_mesh2(end),FN(1,end),['45', num2str(m2)])
50 end
51
52 grid on
53
54
55
56
57 %% Test solver quality : Adjoint
58 clc
59 noise_level = 0.1;
60 time_obs = linspace(obs_start, obs_end, 15);
61 alpha_obs = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_obs);
62 [g, g_brus, g_add] = ExactODE45(alpha_obs,time_obs,noise_level,
   ↪ x_initial); % get observations
63 %interpolate to be able to use for other time meshes
64 gPol =@(t) [];
65 for j = 1:size(g_brus,1)
66    Pol_j =@(t) interp1(time_mesh,g_brus(j,:),t);
67    gPol =@(t) [gPol(t); Pol_j(t)];
68 end
69
70
71 figure
72 hold on
73
74 sch = 0;
75
76 for m2 = [15 20 50 100 500 1000]
77     time_mesh2 = linspace(0,time_final,m2);
78     alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh2
   ↪ );
79     g_brus = gPol(time_mesh2);
80     FN = ForwardNewton(alpha,time_mesh2,x_initial);
```

```
    F45 = ForwardODE45(alpha,time_mesh2,x_initial);
    AN = AdjointNewton(alpha, FN, g_brus, time_mesh2, obs_start,
    ↪ obs_end);
    A45 = AdjointODE45(alpha, F45, g_brus, time_mesh2, obs_start,
    ↪ obs_end);
    sch = sch+1;
    subplot(2, 3, sch);
    plot(time_mesh2,AN(2,:),'-r',time_mesh2,A45(2,:),'--b','
    ↪ linewidth',2)
    %text(time_mesh2(1),A45(2,1),['', num2str(m2)])
    %text(time_mesh2(end),FN(1,end),['45', num2str(m2)])
    xlabel(m2)
    legend('x_T Newton', 'x_T ode45')
    title('ODE45 versus Newton for adjoint problem');
    hold on
end
   title('ODE45 versus Newton for adjoint problem');

grid on



hold off
%% Test Forward Newton vs ode45
alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh);
FN = ForwardNewton(alpha,time_mesh,x_initial);
F45 = ForwardODE45(alpha,time_mesh,x_initial);
figure
subplot(2, 3, 1);
plot(time_mesh,FN(1,:),time_mesh,F45(1,:),'--','linewidth',2)
legend('x_T Newton', 'x_T ode45')
grid on
hold on
%legend('x_T Newton', 'x_M1','x_M2','x_T ode45', 'x_M1','x_M2')

%% Newton and Raluca
%fig 1a
figure

alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh);
FN = ForwardNewton(alpha,time_mesh,x_initial);

subplot(2, 2, 1);
plot(time_mesh,FN,'--','linewidth',2)
grid on
legend('x_T', 'x_{M1}','x_{M2}')
title('(a)')
hold on

%fig 1b

alpha = alpha_vec(10^-11,10^-10,10^-8,10^-10,5*10^-10,time_mesh);
FN1 = ForwardNewton(alpha,time_mesh,x_initial);
alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh);
FN2 = ForwardNewton(alpha,time_mesh,x_initial);
alpha = alpha_vec(10^-7,10^-10,10^-8,10^-10,5*10^-10,time_mesh);
```

```
FN3 = ForwardNewton(alpha,time_mesh,x_initial);

subplot(2, 2, 2);
plot(time_mesh,FN1(1,:),':r',time_mesh,FN2(1,:),'-r',time_mesh,FN3(1,:),'-.r','linewidth',2)
title('(b)')
legend('d_{m1} = 10^{-11}','d_{m1} = 10^{-9}','d_{m1} = 10^{-7}')
ylim([0 4*10^9])
grid on
hold on

%fig 1c

alpha = alpha_vec(10^-9,10^-12,10^-8,10^-10,5*10^-10,time_mesh);
FN1 = ForwardNewton(alpha,time_mesh,x_initial);
alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh);
FN2 = ForwardNewton(alpha,time_mesh,x_initial);
alpha = alpha_vec(10^-9,10^-8,10^-8,10^-10,5*10^-10,time_mesh);
FN3 = ForwardNewton(alpha,time_mesh,x_initial);

subplot(2, 2, 3);
plot(time_mesh,FN1(1,:),':r',time_mesh,FN2(1,:),'-r',time_mesh,FN3(1,:),'-.r','linewidth',2)
title('(c)')
legend('d_{m1} = 10^{-11}','d_{m1} = 10^{-9}','d_{m1} = 10^{-7}')
ylim([0 4*10^9])
grid on
hold on

%fig 1d

alpha = alpha_vec(10^-9,10^-10,10^-10,10^-10,5*10^-10,time_mesh);
FN1 = ForwardNewton(alpha,time_mesh,x_initial);
alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh);
FN2 = ForwardNewton(alpha,time_mesh,x_initial);
alpha = alpha_vec(10^-9,10^-10,10^-6,10^-10,5*10^-10,time_mesh);
FN3 = ForwardNewton(alpha,time_mesh,x_initial);

subplot(2, 2, 4);
plot(time_mesh,FN1(1,:),':r',time_mesh,FN2(1,:),'-r',time_mesh,FN3(1,:),'-.r','linewidth',2)
title('(d)')
legend('d_{m1} = 10^{-11}','d_{m1} = 10^{-9}','d_{m1} = 10^{-7}')
ylim([0 4*10^9])
grid on

%% Test Adjoint newton vs ode45
alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh);
FN = ForwardNewton(alpha,time_mesh,x_initial);
F45 = ForwardODE45(alpha,time_mesh,x_initial);
AN = AdjointNewton(alpha, FN, g_brus1, time_mesh, obs_start, obs_end);
A45 = AdjointODE45(alpha, F45, g_brus1, time_mesh, obs_start, obs_end);

figure
```

```
for i = 1:size(AN,1)
    subplot(2, 3, i);
    plot(time_mesh,AN(i,:),'linewidth',2)
    plot(time_mesh,A45(i,:),'--','linewidth',2)
    legend(['\lambda',num2str(i),'Newton'],['\lambda',num2str(i),'
    ↪ ode45'])
    hold on
end


%% Inner functions

function alpha = alpha_vec(dm1,dm2,at1,at2,k12,time_mesh)
    scaling_factor_dm1 = dm1;
    scaling_factor_dm2 = dm2;
    scaling_factor_at1 = at1;
    scaling_factor_at2 = at2;
    scaling_factor_k12 = k12;

    function_flag = 0; % constant

    exact_dm1 = ExactParameter(scaling_factor_dm1,function_flag,
    ↪ time_mesh); %Exact profile for dm1 to produce data.
    exact_dm2 = ExactParameter(scaling_factor_dm2,function_flag,
    ↪ time_mesh); %Exact profile for dm2 to produce data.
    exact_at1 = ExactParameter(scaling_factor_at1,function_flag,
    ↪ time_mesh); %Exact profile for at1 to produce data.
    exact_at2 = ExactParameter(scaling_factor_at2,function_flag,
    ↪ time_mesh); %Exact profile for at2 to produce data.
    exact_k12 = ExactParameter(scaling_factor_k12,function_flag,
    ↪ time_mesh); %Exact profile for k12 to produce data.

    alpha = [exact_dm1; exact_dm2; exact_at1; exact_at2; exact_k12
    ↪ ];

end
```

test2_Chebyshew.m

```
%%  Initials
clc; close all; clear all;
%%initial of x and time variables
m = 15; % number of observations
obs_start = 2.1; obs_end = 7; %interval of observations
time_final = 20;
t_min = 0;

%time_mesh = linspace(0,time_final,m);
% x_initial = [x_T(0); x_M1(0); x_M2(0)]
% initial values that seems to fit fig 1a
x_initial = [5*10^6; 10^3; 10^3];
time_mesh =[];
for k = 1:m
    time_mesh(end+1) = -(time_final-t_min)/2*cos((2*k-1)*pi/(2*m))
    ↪ + (time_final+t_min)/2;
```

```
end

%% observations
alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh);
noise_level = 0.1;% 10% noise

%using ode45 since newton seems to have problems with low numbers for
% this particular system of odes

[g, g_brus, g_add] = ExactODE45(alpha,time_mesh,noise_level, x_initial);
figure

plot(time_mesh,g(1,:),'linewidth',2)
hold on
plot(time_mesh,g_brus(1,:),'*')
  legend('x_T, ode45','observations g1')
  title(['ODE45 versus noisy data, noise , \delta= ',num2str( noise_level)]);


%% Test solver quality : Forward
figure

hold on
sch = 0;
for m2 = [50 100 200 400 800 1000]
    time_mesh2 = linspace(0,time_final,m2);
    alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh2 );
    FN = ForwardNewton(alpha,time_mesh2,x_initial);
    F45 = ForwardODE45(alpha,time_mesh2,x_initial);

sch = sch+1;
subplot(2, 3, sch);

plot(time_mesh2,FN(1,:),'-r',time_mesh2,F45(1,:),'--b','linewidth' ,2);
title(' forward problem');
xlabel(m2)

legend('x_T Newton', 'x_T ode45')
%     text(time_mesh2(end),FN(1,end),['N', num2str(m2)])
%     text(time_mesh2(end),FN(1,end),['45', num2str(m2)])
end

grid on


hold off


%% Test solver quality : Adjoint
clc
noise_level = 0.1;
```

```
time_obs = linspace(obs_start, obs_end, 15);
alpha_obs = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_obs);
[g, g_brus, g_add] = ExactODE45(alpha_obs,time_obs,noise_level,
    ↪ x_initial); % get observations
%interpolate to be able to use for other time meshes
gPol =@(t) [];
for j = 1:size(g_brus,1)
    Pol_j =@(t) interp1(time_mesh,g_brus(j,:),t);
    gPol =@(t) [gPol(t); Pol_j(t)];
end


figure
hold on

sch = 0;

for m2 = [50 100 200 400 800 1000]
    time_mesh2 = linspace(0,time_final,m2);
    alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh2
    ↪ );
    g_brus = gPol(time_mesh2);
    FN = ForwardNewton(alpha,time_mesh2,x_initial);
    F45 = ForwardODE45(alpha,time_mesh2,x_initial);
    AN = AdjointNewton(alpha, FN, g_brus, time_mesh2, obs_start,
    ↪ obs_end);
    A45 = AdjointODE45(alpha, F45, g_brus, time_mesh2, obs_start,
    ↪ obs_end);

sch = sch+1;
subplot(2, 3, sch);
    plot(time_mesh2,AN(2,:),'-r',time_mesh2,A45(2,:),'--b','
    ↪ linewidth',2)
%    text(time_mesh2(1),A45(2,1),['', num2str(m2)])
%     text(time_mesh2(end),FN(1,end),['45', num2str(m2)])
    xlabel(m2)
    title('adjoint problem');
    legend('x_T Newton', 'x_T ode45')
end


grid on



hold off

%% Inner functions

function alpha = alpha_vec(dm1,dm2,at1,at2,k12,time_mesh)
    scaling_factor_dm1 = dm1;
    scaling_factor_dm2 = dm2;
    scaling_factor_at1 = at1;
    scaling_factor_at2 = at2;
    scaling_factor_k12 = k12;

```

```
    function_flag = 0; % constant

    exact_dm1 = ExactParameter(scaling_factor_dm1,function_flag,
    ↪ time_mesh); %Exact profile for dm1 to produce data.
    exact_dm2 = ExactParameter(scaling_factor_dm2,function_flag,
    ↪ time_mesh); %Exact profile for dm2 to produce data.
    exact_at1 = ExactParameter(scaling_factor_at1,function_flag,
    ↪ time_mesh); %Exact profile for at1 to produce data.
    exact_at2 = ExactParameter(scaling_factor_at2,function_flag,
    ↪ time_mesh); %Exact profile for at2 to produce data.
    exact_k12 = ExactParameter(scaling_factor_k12,function_flag,
    ↪ time_mesh); %Exact profile for k12 to produce data.

    alpha = [exact_dm1; exact_dm2; exact_at1; exact_at2; exact_k12
    ↪ ];

end
```

test2.m

```
%%  Initials
clc; close all; clear all;
%%initial of x and time variables
m = 15; % number of observations
obs_start = 2.1; obs_end = 7; %interval of observations
time_final = 20;

time_mesh = linspace(0,time_final,m);
% x_initial = [x_T(0); x_M1(0); x_M2(0)]
% initial values that seems to fit fig 1a
x_initial = [5*10^6; 10^3; 10^3];


%% observations
alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh);
noise_level = 0.1;% 10% noise


[g, g_brus, g_add] = ExactODE45(alpha,time_mesh,noise_level,
   ↪ x_initial);
figure

plot(time_mesh,g(1,:),'linewidth',2)
hold on
plot(time_mesh,g_brus(1,:),'*')
  legend('x_T, ode45','observations g1')
  title(['ODE45 versus noisy data, noise , \delta= ',num2str(
   ↪ noise_level)]);


%% Test solver quality : Forward
figure

hold on
sch = 0;
```

```
for m2 = [50 100 200 400 800 1000]
    time_mesh2 = linspace(0,time_final,m2);
    alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh2
    );
    FN = ForwardNewton(alpha,time_mesh2,x_initial);
    F45 = ForwardODE45(alpha,time_mesh2,x_initial);

sch = sch+1;
subplot(2, 3, sch);

plot(time_mesh2,FN(1,:),'-r',time_mesh2,F45(1,:),'--b','linewidth'
    ,2);
title(' forward problem');
xlabel(m2)

legend('x_T Newton', 'x_T ode45')
%     text(time_mesh2(end),FN(1,end),['N', num2str(m2)])
%     text(time_mesh2(end),FN(1,end),['45', num2str(m2)])
end

grid on


hold off


%% Test solver quality : Adjoint
clc
noise_level = 0.1;
time_obs = linspace(obs_start, obs_end, 15);
alpha_obs = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_obs);
[g, g_brus, g_add] = ExactODE45(alpha_obs,time_obs,noise_level,
    x_initial); % get observations
%interpolate to be able to use for other time meshes
gPol =@(t) [];
for j = 1:size(g_brus,1)
   Pol_j =@(t) interp1(time_mesh,g_brus(j,:),t);
   gPol =@(t) [gPol(t); Pol_j(t)];
end


figure
hold on

sch = 0;

for m2 = [50 100 200 400 800 1000]
    time_mesh2 = linspace(0,time_final,m2);
    alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh2
    );
    g_brus = gPol(time_mesh2);
    FN = ForwardNewton(alpha,time_mesh2,x_initial);
    F45 = ForwardODE45(alpha,time_mesh2,x_initial);
    AN = AdjointNewton(alpha, FN, g_brus, time_mesh2, obs_start,
    obs_end);
    A45 = AdjointODE45(alpha, F45, g_brus, time_mesh2, obs_start,
```

```
    ↪ obs_end);

sch = sch+1;
subplot(2, 3, sch);
    plot(time_mesh2,AN(2,:),'-r',time_mesh2,A45(2,:),'--b','
    ↪ linewidth',2)
%    text(time_mesh2(1),A45(2,1),['', num2str(m2)])
%     text(time_mesh2(end),FN(1,end),['45', num2str(m2)])
    xlabel(m2)
    title('adjoint problem');
    legend('x_T Newton', 'x_T ode45')
end


grid on



hold off

%% Inner functions

function alpha = alpha_vec(dm1,dm2,at1,at2,k12,time_mesh)
    scaling_factor_dm1 = dm1;
    scaling_factor_dm2 = dm2;
    scaling_factor_at1 = at1;
    scaling_factor_at2 = at2;
    scaling_factor_k12 = k12;

    function_flag = 0; % constant

    exact_dm1 = ExactParameter(scaling_factor_dm1,function_flag,
    ↪ time_mesh); %Exact profile for dm1 to produce data.
    exact_dm2 = ExactParameter(scaling_factor_dm2,function_flag,
    ↪ time_mesh); %Exact profile for dm2 to produce data.
    exact_at1 = ExactParameter(scaling_factor_at1,function_flag,
    ↪ time_mesh); %Exact profile for at1 to produce data.
    exact_at2 = ExactParameter(scaling_factor_at2,function_flag,
    ↪ time_mesh); %Exact profile for at2 to produce data.
    exact_k12 = ExactParameter(scaling_factor_k12,function_flag,
    ↪ time_mesh); %Exact profile for k12 to produce data.

    alpha = [exact_dm1; exact_dm2; exact_at1; exact_at2; exact_k12
    ↪ ];

end
```

test3.m

```
%%  Initials
clc; close all; clear all;
%%initial of x and time variables
m = 15; % number of observations
obs_start = 2.1; obs_end = 7; %interval of observations
time_final = 20;
```

```
7
8  time_mesh = linspace(0,time_final,m);
9  % x_initial = [x_T(0); x_M1(0); x_M2(0)]
10 % initial values that seems to fit fig 1a
11 x_initial = [5*10^6; 10^3; 10^3];
12
13
14 %% observations
15 alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh);
16 noise_level = 0.1;% 10% noise
17
18 %using ode45 since newton seems to have problems with low numbers
     ↪ for
19 % this particular system of odes
20
21 [g, g_brus, g_add] = ExactODE45(alpha,time_mesh,noise_level,
     ↪ x_initial);
22 figure
23
24 plot(time_mesh,g(1,:),'linewidth',2)
25 hold on
26 plot(time_mesh,g_brus(1,:),'*')
27   legend('x_T, ode45','observations g1')
28   title(['ODE45 versus noisy data, noise , \delta= ',num2str(
     ↪ noise_level)]);
29
30
31 %% Test solver quality : Forward
32 figure
33
34 hold on
35 sch = 0;
36 for m2 = [50 100 200 400 800 1000]
37     time_mesh2 = linspace(0,time_final,m2);
38     alpha = alpha_vec(10^-9,10^-10,10^-8,10^-10,5*10^-10,time_mesh2
     ↪ );
39
40     F45 = ForwardODE45(alpha,time_mesh2,x_initial);
41
42 sch = sch+1;
43 subplot(2, 3, sch);
44
45
46 plot(time_mesh2,F45(1,:),'--r',time_mesh2,F45(2,:),'--b',time_mesh2
     ↪ ,F45(3,:),'--m','linewidth',2);
47
48
49 title(' forward problem');
50 %xlabel(m2)
51 xlabel(['nr.of discr. points:',num2str(m2)]);
52 %title(['ODE45 versus noisy data, noise , \delta= ',num2str(
     ↪ noise_level)]);
53
54 legend('x_T','x_{M_1}','x_{M_2}')
55
56
```

```
%      text(time_mesh2(end),FN(1,end),['N', num2str(m2)])
%      text(time_mesh2(end),FN(1,end),['45', num2str(m2)])
end

grid on


hold off


%% Inner functions

function alpha = alpha_vec(dm1,dm2,at1,at2,k12,time_mesh)
    scaling_factor_dm1 = dm1;
    scaling_factor_dm2 = dm2;
    scaling_factor_at1 = at1;
    scaling_factor_at2 = at2;
    scaling_factor_k12 = k12;

    function_flag = 0; % constant

    exact_dm1 = ExactParameter(scaling_factor_dm1,function_flag,
    ↪ time_mesh); %Exact profile for dm1 to produce data.
    exact_dm2 = ExactParameter(scaling_factor_dm2,function_flag,
    ↪ time_mesh); %Exact profile for dm2 to produce data.
    exact_at1 = ExactParameter(scaling_factor_at1,function_flag,
    ↪ time_mesh); %Exact profile for at1 to produce data.
    exact_at2 = ExactParameter(scaling_factor_at2,function_flag,
    ↪ time_mesh); %Exact profile for at2 to produce data.
    exact_k12 = ExactParameter(scaling_factor_k12,function_flag,
    ↪ time_mesh); %Exact profile for k12 to produce data.

    alpha = [exact_dm1; exact_dm2; exact_at1; exact_at2; exact_k12
    ↪ ];

end
```