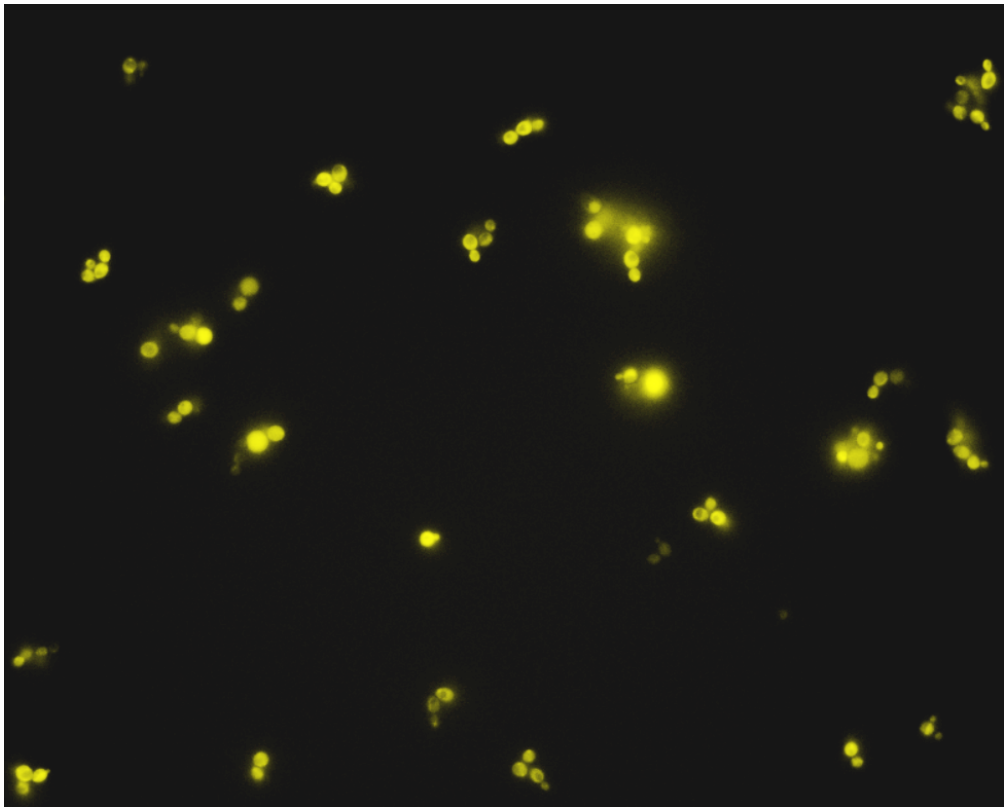
CHALMERS | GÖTEBORGS UNIVERSITET



Modellering av SUC2-responsen i Snf1-signalvägen hos S. cerevisiae efter extern glukosminskning

Kandidatarbete inom civilingenjörsprogrammen Bioteknik och Teknisk Matematik

Felix Augustsson, Ellen Sandén, Tilia Selldén, Hanna Zetterberg

Institutionen för matematiska vetenskaper
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2019

Modellering av SUC2-responsen i
Snf1-signalvägen hos S. cerevisiae
efter extern glukosminskning

Felix Augustsson, Ellen Sandén, Tilia Selldén, Hanna Zetterberg

Handledare: Johannes Borgqvist,
Niek Welkenhuysen,
Marija Cvijovic

Examinator: Maria Roginskaya,
Ulla Dinger

Kandidatarbete MVEX01-19-02
Institutionen för matematiska vetenskaper
Cvijovic Lab
Chalmers Tekniska Högskola och Göteborgs Universitet

Omslag: YFP-markerade jästceller i ett av experimenten som använts i arbetet. Bild framtagen av Niek Welkenhuysen och färgad av Felix Augustsson.

Typeset in LaTeX
Göteborg, Sverige 2019

Förord

För nästan fem månader sedan påbörjade fyra forskningslandkrabbor sin första resa genom vetenskapens vatten. Välkomnade med öppna armar av besättningen på ett forskningsskepp dedikerade till att hitta svaret på den mytomspunna gåtan om åldrande satte vi direkt segel. Trots att vi hade fått en karta med målet markerat har vi under arbetets gång flera gånger fått byta riktning. Vi har behövt överge delmål även efter att veckor lagts på att utforma en plan och otaliga timmar har lagts under däck för att hålla oss flytande. Nedan följer därför en kort sammanfattning av vår loggbok.

Tilia har varit vår utkik, konstant letandes efter nya kuster att lägga till vid. Hon har under projektets gång letat genom litteratur efter biologiska förklaringar och skapat modeller baserat på det hon funnit. När litteraturen inte gav de svar som behövdes tog hon kontakt med forskaren bakom studien. När tiden kom att skriva denna rapport om våra äventyr har hon bidragit med en detaljrikedom så beskrivande att den i flera fall behövts tas bort för att hålla rapporten någorlunda koncis.

Ellen har varit vår kartläsare, tätt samarbetande med Tilia för att förstå biologin i projektet. Under skapandet av de biologiska modellerna har Ellen använt strukturell identifierbarhet för att hitta de grund som gömt sig under ytan hos Tilias observationer. Likt en äkta kartograf har ingen detalj undgått Ellen under rapportskrivandet, och hon har obevekligt rättat de fel vi andra gjort och fyllt i tomrummen vi lämnat.

Felix har varit vår båtsman, ansvarig för att all MATLAB-kod gjorde det den skulle. Han har sett över parameteruppskattningen och sett till att vi kunnat hissa ankar vid simuleringarna. I rapportskrivandet har Felix släckt de eldar som uppstått ombord, och sett till att alla matematiska knopar var rätt knutna.

Hanna har varit vår styrman, ansvarig för att vi använde rätt matematiska segel för att ta oss till vårt mål. Genom att granska resultaten från Felix simuleringar med numerisk identifierbarhet har hon sett till att hela arbetet inte var förgäves när målet var i sikte. Hanna har sett till att vi har legat i kurs mot en färdig slutrapport under skrivandet, och inte hamnat på villospår.

Runt omkring oss har vi hela tiden haft en forskargrupp fylld med så mycket erfarenhet och hjälpsamhet att vi inte skulle kunnat utnyttja allt hur många månader vi än fick på oss. Tack så enormt mycket inte bara för all hjälp, utan även för den underbara stämning ni inkluderat oss i. Särskilt mycket tack till våra handledare Johannes Borgqvist, Niek Welkenhuysen och Marija Cvijovic som inte bara hjälpt oss undvika skeppsbrott ett flertal gånger, utan även gjort det med ett engagemang utöver det vanliga. Vi vill även rikta ett stort tack till Joachim Almquist som har svarat på alla våra frågor kring det koncept som sedan kom att ligga till grund för vår slutgiltiga modell.

Alla inblandade har gjort hela kandidatarbetet till ett enda stort äventyr. Även om vi nu ska lägga till i hamn, så tar vi med oss erfarenheter och vänner hem från resan. Och vem vet, vi kanske ses ute på havet igen?

Populärvetenskaplig text

Hur kan vi rädda liv med hjälp av matte och bagerijäst? Det kanske låter som en osannolik kombination, men det är faktiskt ofta en oslagbar sådan.

Världen vi lever i är komplex, och det är en utmaning att försöka förstå oss på alla de biologiska fenomen som sker inuti och runt omkring oss. Hur ska vi kunna veta vad som försiggår inne i en cell eller en människokropp när vi inte kan se det med blotta ögat eller ens med hjälp av ett mikroskop? Ett sätt är att använda matematik. Genom att skapa matematiska modeller över vad som sker kan biologiska mekanismer belysas och förstås på ett sätt som annars inte hade varit möjligt. Att modellera biologiska system handlar om att sätta upp matematiska ekvationer som kan beskriva hur det biologiska systemet beter sig. Det kan till exempel handla om att beskriva hur en tumör reagerar på medicinen patienten får, eller om att öka förståelsen kring en specifik gen. För att en modell ska vara användbar behöver den förstås stämma väl överens med verkligheten. Det måste gå att jämföra det modellen förutspår med vad som faktiskt händer, och till detta behövs experimentella data. Ett traditionellt tillvägagångssätt för att få tag på sådana har varit, och är fortfarande, att utföra djurtest. Dessa är dock både kostsamma och ofta djupt plågsamma för djuren; 2016 startades över 350 000 djurförsök i Sverige, varav nästan 30 000 klassades som ingrepp av den allra värsta och mest smärtsamma, ångestbringande sorten [1]. Det är här bagerijästen kommer in.

Jäst är små encelliga organismer som finns lite här och var omkring oss, på allt från skalet på vindruvor till i sanden på havsbottnen [2]. Trots att jäst kan tyckas vara en väldigt enkel livsform i jämförelse med en människa, har de små organismerna faktiskt mycket gemensamt med oss. Precis som människor, men till skillnad från till exempel bakterier, har jästceller en cellkärna där allt det genetiska materialet i form av DNA är samlat.

Genom evolutionens gång har mycket av det här DNA:t bevarats ungefär så som det var för miljoner år sedan, i den organism som en gång var anfader till både jäst och människa. Mycket av vårt mänskliga DNA är därför väldigt likt jäst-DNA. För att undersöka funktionen hos en mänsklig gen är det en bra idé att titta på om samma gen finns hos jäst, och hur den i så fall fungerar där.

En sådan gen, som finns hos och är viktig för både jäst och människa, är den som styr så att energinivåerna i cellerna är konstanta. Hos människa heter den AMPK och hos jäst har den fått namnet Snf1. Detta är en gen som i sin tur styr andra gener och kan slå på och av produktionen av protein och enzymer. För att undersöka genens funktion närmare har vi i detta kandidatarbete skapat en modell över hur mängden av ett av enzymerna som styrs av Snf1 påverkas av koncentrationen druvsocker som finns i jästens omgivning.

Druvsocker är det som jäst allra helst vill äta. Det är en relativt liten sockermolekyl, och den är därför enkel för jästen att ta upp från omgivningen in genom cellmembranet. Det är dock inte alltid denna typ av socker finns tillgängligt. Ibland hamnar jästen på ett ställe där det bara finns större sockermolekyler att tillgå, såsom betsocker, som är

dubbelt så stora som druvsocker. Betsockermolekylen är för stor för jästen att ta upp, och sockret måste därför brytas ned i mindre delar för att jästen ska kunna tillgodogöra sig det. För att göra detta har jästen utvecklat speciella enzymer som den kan utsöndra till omgivningen och som kan bryta sönder de stora sockermolekylerna. Ett sådant enzym heter SUC2, och är specialdesignat för att bryta sönder betsocker. Eftersom det kostar energi att producera enzym, vill jästen inte göra detta i onödan. Den har därför utvecklat ett system för att känna av om SUC2 behöver produceras eller inte. Detta system styrs av Snf1, och är baserat på hur mycket druvsocker som finns i omgivningen. Om det finns mycket druvsocker i omgivningen kan jästen obekymrat mumsa i sig detta, utan att behöva slösa energi på att producera SUC2. Om det däremot inte finns något druvsocker måste jästen hitta något annat att äta. För att testa om det kanske kan finnas betsocker i omgivningen produceras därför SUC2. Om betsocker finns kommer detta att brytas ned och de numera små sockermolekylerna kommer att åka in i jästen, där energin kan utvinnas. Snf1 slår alltså på och av SUC2 baserat på hur omgivningen ser ut.

Även om de flesta människor kanske har annan favoritmat än just druvsocker, innebär en ökad kunskap om Snf1 hos jäst en ökad kunskap om AMPK hos människa. På liknande sätt kan massor med andra mänskliga gener och funktioner undersökas med hjälp av den lilla encelliga organismen. I takt med att vi blir bättre och bättre på att skapa verklighetstrogna modeller, och i takt med att våra datorer klarar av att hantera mer och mer data, kan vi öka vår kunskap om oss själva och världen omkring oss utan att behöva använda oss av plågsamma djurförsök. På sikt kan matte och jäst rädda liv, både mänskliga och icke-mänskliga.

Sammanfattning

Snf1-signalvägen är en viktig del av näringssignaleringen hos jäst. För att undersöka Snf1-signalvägen hos *S. cerevisiae* skapades i detta projekt fyra olika matematiska modeller över hur uttrycket av invertaset SUC2 varierade efter ett skifte i den yttre glukoskoncentrationen från 4 % till 0,1 %. I signalvägen ingår förutom Snf1 och SUC2 även transkriptionsfaktorn Mig1. Modellerna byggdes upp av ordinära differentialekvationer.

För att validera de skapade modellerna anpassades de till uppmätt data. De två första modellerna, modell 0 och 1, lyckades inte beskriva det experimentellt observerade beteendet hos uttrycket av SUC2 och analyserades inte vidare. Detta lyckades dock de två andra modellerna, X-modellerna, bättre med. X-modellerna är två förenklingar av samma modell, reducerade för att vara strukturellt identifierbara. I X-modellerna utgjorde Snf1 en ”svart låda” och alla de okända faktorer som påverkar uttrycket av SUC2 samlades i tillståndsvariabeln X. Den ena X-modellen, den svagt reducerade versionen, fångade datans beteende i början av experimentet medan den andra, den tungt reducerade versionen, fångade stabiliserandet av SUC2-uttrycket som uppkom i slutet av experimentet.

Residualanalys genomfördes för X-modellerna och för att undersöka modellernas prediktionsförmåga gjordes även ett försök till numerisk identifierbarhetsanalys. Detta misslyckades dock då approximationerna av Hessianerna inte kunde användas för att beräkna kovariansmatriserna.

Abstract

The Snf1 pathway is an important part of the nutritional signaling network in yeast. In this project the Snf1 pathway in *S. cerevisiae* was investigated using mathematical modelling. Four different models of the expression of the invertase SUC2 after a shift in the external glucose concentration from 4 % to 0.1 % were created. In addition to Snf1 and SUC2 the modelled pathway also includes the transcription factor Mig1. The models were made using ordinary differential equations.

In order to validate the models they were fitted to experimental data. The first two models, model 0 and model 1, did not accurately describe the observed behavior of the expression of SUC2 and were not analysed further. The other two models, however, did. These were named the X-models, and are two simplifications of the same model, reduced to become structurally identifiable. In the X-models Snf1 was black boxed and all the unknown factors affecting the expression of SUC2 were lumped together in the state variable X. One of the X-models, the lightly reduced version, captured the observed behavior in the beginning of the experiment, whereas the other X-model, the heavily reduced version, captured the stabilization of the expression of SUC2 which occured by the end of the experiment.

Residual analysis was performed on the X-models and in order to investigate their ability for prediction a numerical identifiability analysis was initiated. This, however, failed, as the Hessian approximations could not be used to calculate the covariance matrices.

Ordlista

- **Eukaryoter** är organismer vars celler har membran kring cellkärnan. Exempel på eukaryoter är djur, växter och svampar.
- ***Saccharomyces cerevisiae*** är det vetenskapliga namnet på bagerijäst, en vanlig jästsort som bland annat används inom industri, forskning och till matlagning.
- **Uttryck av gener** innebär att koden i en DNA-sekvens avläses och översätts till ett protein.
- **Promotor** är en kort DNA-sekvens, placerad framför en gen som fungerar som på och av knapp för den aktuella genen.
- **Inihibering** av en gen innebär att den hindras från att uttryckas.
- **Transkriptionsfaktorer** är proteiner som kan aktivera eller hindra uttrycket av en gen, till exempel genom att sätta sig på dess promotor.
- **Enzymer** är stora proteiner som hjälper till att katalysera kemiska reaktioner i och utanför cellen.
- **Invertaser** är en kategori enzymer som har i uppgift att bryta ned större sockermolekyler, sackaros, till mindre sockerarter så som glukos och fruktos som cellen använder som energikällor.
- **Kinaser** är en kategori enzymer som kan katalysera fosforyleringen av andra enzym eller protein.
- **Fosforylering** är ett sätt för cellen att skicka vidare en signal genom bindning av en fosfatgrupp. Kinaser fosforylerar proteiner så att de aktiveras/deaktiveras.

Innehåll

1 Inledning och bakgrund

Alla celler, från bakterier och jäst till våra egna celler i kroppen, har förmågan att sätta på eller stänga av sina egna gener. Det kan till exempel handla om att reglera vilken del av cellcykeln cellen ska befinna sig i eller att stänga av en gen för att inte slösa onödig energi på att producera saker som för tillfället inte behövs. Regleringen av genuttrycket hos en cell börjar med att en förändring i omgivningen registreras. Detta leder till att en signal skickas genom cellen via olika molekyler såsom enzymer och andra proteiner. Om cellen är eukaryot, och alltså har cellkärna, måste signalen nå in till denna och aktivera transkriptionsfaktorer som kan starta eller inhibera uttrycket av de specifika gener som behövs. Hela processen, från att en signal skickas tills att en gen slås på kallas signalväg, och det finns ett enormt antal sådana i en cell. Då olika signalvägar dessutom påverkar varandra bildar de tillsammans ett komplext nätverk som tillåter cellen att dynamiskt anpassa sig allteftersom omgivningen förändras.

En av alla de signalvägar som är livsviktiga för eukaryota celler är den som ser till att deras energinivåer hålls konstanta [3]. Signalvägen styrs framför allt av en gen, som genom evolutionen bevarats så att den är nästan identisk i olika eukaryota organismer. Hos människan heter genen "*A*denosine *m*onophosphate-activated *p*rotein *k*inase" (AMPK) och påverkar förutom energinivåerna även cellulär stressresistens, hur cellen återvinner gamla proteiner och det spekuleras till och med i huruvida AMPK kan påverka vår livslängd [4]. Genen interagerar dessutom med läkemedel både mot diabetes och cancer, men hur detta går till är inte känt [5].

Det är av stor vikt att öka kunskapen om AMPK-signalvägen, men att studera mänskliga celler är svårt. Att studera jästceller är däremot ett vanligt och relativt enkelt sätt att undersöka cellulära egenskaper. Cellerna tillväxer snabbt och klarar sig på förhållandevis enkla näringsmedium, som glukos. Dessutom gör faktumet att jästceller precis som människor är eukaryoter, att även de har den aktuella signalvägen inbyggd. Hos jäst heter den styrande genen inte AMPK utan "*S*ucrose *n*on*f*ermenting *1*" (Snf1).

Snf1 reglerar många proteiner, däribland transkriptionsfaktorn"*M*ulticopy *i*nhibitor of *G*AL gene expression *1*" (Mig1), som i sin tur reglerar omkring 350 olika gener [6]. En av de gener som regleras av Mig1, och därmed även av Snf1, är "*Suc*rose 2" (SUC2). Det är ett enzym som bryter ned sackaros till glukos och fruktos, och uttrycks bara då varken maltos, fruktos eller glukos finns i jästcellens omgivning. Exakt hur detta går till, alltså hur Snf1 känner av halterna maltos, fruktos och glukos i omgivningen, är inte känt [7]. Det är därför av intresse att vidare undersöka signalvägen där SUC2, Mig1 och Snf1 ingår. Det görs i detta arbete genom att halten glukos kring en jästcellspopulation hastigt sänks och responsen i form av koncentrationen SUC2 mäts över tid. En matematisk modell över hur uttrycket av SUC2 varierar sätts upp och passas till datan från jästcellsexperimentet.

För att kunna använda den framtagna modellen till att prediktera hur signalvägen beter sig, till exempel om glukoskoncentrationen höjs istället för sänks, måste det gå att unikt bestämma värdena hos modellens parametrar, det vill säga parametrarna måste vara

identifierbara. Därför utförs identifierbarhetsanalyser.

Genom att skapa och validera en matematisk modell över uttrycket av SUC2 kan förståelsen för signalvägen ökas. Detta skulle i förlängningen leda till större förståelse för funktionen hos Snf1, och därmed även dess mänskliga analog AMPK.

2 Syfte

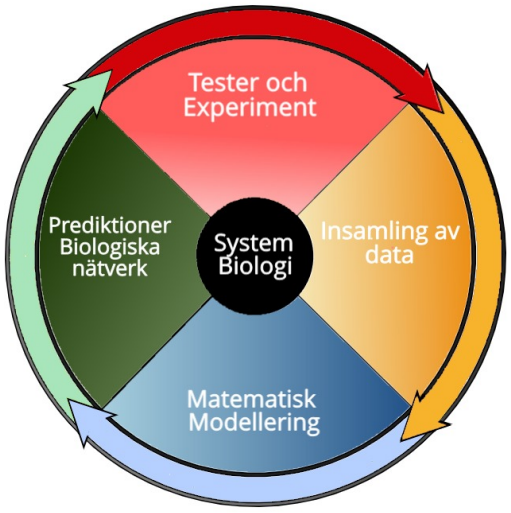
I denna rapport undersöks med hjälp av matematisk modellering hur uttrycket av SUC2 i *Saccharomyces cerevisiae* förändras över tid när den yttre glukoskoncentrationen ändras från 4 % till 0,1 %. De framtagna modellerna valideras genom att anpassas till experimentella data och modellparametrarnas identifierbarhet undersöks.

3 Teori

Följande kapitel innehåller den teoretiska bakgrunden till rapporten. Den inleds med en biologisk bakgrund om systembiologi och det system som ska modelleras och fortsätter med att generellt beskriva de verktyg som används.

3.1 Systembiologins grunder

Biologiska signalvägar är ofta för komplexa för att endast kunna förklaras och undersökas med experiment där varje enskild komponent i signalvägen studeras. För att undersöka större nätverk kan experimenten kombineras med matematisk modellering Detta görs inom systembiologin som kännetecknas av att flera olika discipliner som biologi, matematik och datavetenskap kombineras för att få en djupare förståelse för hur biologiska nätverk fungerar [8]. Den systembiologiska processen är ofta cyklisk (Figur 1). Processen inleds med att göra tester på den organism som ska undersökas. Experimentell data insamlas och därefter sätts matematiska modeller upp. Om modellen kan beskriva datans beteende kan prediktioner göras om det biologiska systemet [8]. Prediktioner kan därefter bekräftas eller förkastas genom nya experiment, och cykeln börjar om på nytt.

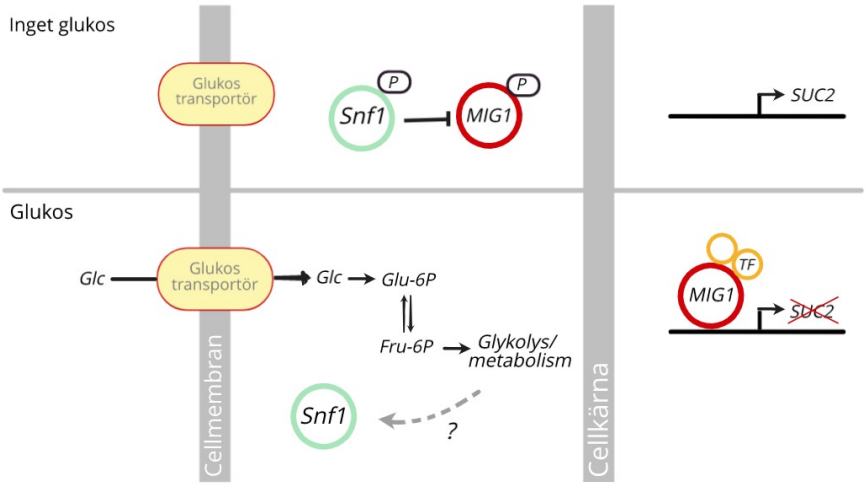


Figur 1: Schematisk bild över den systembiologiska cykeln.

3.2 Snf1-signalvägen

Snf1-signalvägen är en av flera signalvägar som ingår i det stora nätverket av näringssignalering hos *S. cerevisiae* [7]. Snf1 har visat sig vara aktivt, alltså i fosforylerad form, då glukoshalten i cellens omgivning är låg. Proteinet har en stor roll i reglering av många olika gener [9]. En av generna, SUC2, regleras av så kallad glukosrepression. Detta innebär att under glukosfattiga förhållanden kommer genen att uttryckas i stor mängd, men då glukoshalten i omgivningen är hög kommer genen att vara avstängd.

Mer detaljerat är uttrycket av SUC2 huvudsakligen reglerat av Snf1 och Mig1. Då inget glukos finns i jästens omgivning är Snf1 fosforylerat. Fosforylerat Snf1 kommer i sin tur att fosforylera transkriptionsfaktorn Mig1. I ickefosforylerad form inhiberar Mig1 uttrycket av SUC2, men då Mig1 fosforyleras lossnar det från DNA:t och SUC2-genen kan uttryckas fritt. Om glukos istället finns tillgängligt i omgivningen är Snf1 inte fosforylerat. Detta gör att Snf1 inte kan fosforylera Mig1 som då stannar på DNA:t och inhiberar uttrycket av SUC2 (Figur 2) [7].



Figur 2: Schematisk bild över Snf1-signalvägen. När glukos finns närvarande kan cellen utvinna energi genom att bryta ned den energirika molekylen. Mig1 inhiberar uttrycket av SUC2 med hjälp av andra transkriptionsfaktorer (TF). Om inget glukos finns närvarande aktiveras Snf1 som fosforylerar Mig1. Detta medför att SUC2 kan uttryckas fritt.

3.3 Generella metoder för modellering av biologiska system

För att beskriva ett biologiskt nätverk som förändras över tid är ordinära differentialekvationer, ODE:er, ett vanligt verktyg inom systembiologin [10]. Vanligen utnyttjas dessutom enzymatisk kinetik och massverkans lag för att beskriva de biokemiska reaktionerna [11]. Koncentrationen av ett protein, A, i ett slutet system beror då av hur mycket som skapas respektive bryts ned enligt

$$\frac{dA}{dt} = \{\text{genuttrycket av } A\} - \{\text{nedbrytning av } A\} = r_1 - r_2,$$

där r_i är hastigheten med vilken genuttrycket respektive nedbrytningen sker. Om A antas bildas och brytas ned enligt

$$\xrightarrow{k_1} A \xrightarrow{k_2} \emptyset$$

där $\emptyset$ symboliserar nedbrytningen, kan nedbrytningshastigheten beskrivas som en första ordningens reaktion på formen

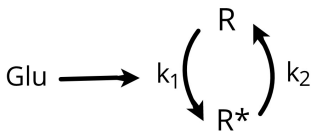
$$r_2 = k_2 \cdot A \tag{1}$$

där A är koncentrationen av proteinet [10][s. 171].

Ett biologiskt nätverk kan bestå av flera sammankopplade delar, så som en signalväg som leder till ett genuttryck. Fosforylering av ett protein i en signalväg kan illustreras som en ”loop” där proteinet har två stadier som det kan växla mellan: fosforylerad och ickefosforylerad form (Figur 3) [10]. Ett sätt att matematiskt beskriva detta är

$$\frac{dR^*}{dt} = k_1 \cdot S \cdot (R_{tot} - R^*) - k_2 \cdot R^*, \tag{2}$$

där R^* är mängden fosforylerat protein, R_{tot} den konstanta totala mängden protein, S den tidigare komponenten i signalvägen och k_1 och k_2 hastighetskonstanter [10, s. 218]. Här är alltså $r_1 = k_1 \cdot S \cdot (R_{tot} - R^*)$ och r_2 är på samma form som i ekvation (1). Ekvationen beskriver balansen mellan fosforylering och defosforylering.



Figur 3: Schematisk bild över fosforyleringen av R påverkad av S.

Genuttryck är en starkt reglerad process [10]. Om uttrycket regleras genom inhibering kan bildningshastigheten för uttrycket av en gen inhiberat av en transkriptionsfaktor B beskrivas med

$$r_1 = \frac{V_{max}}{K_B + B^n} \tag{3}$$

där V_{max} är maxhastigheten med vilken den inhiberande genen uttrycks, n är Hill-koefficienten och K_B är en inhiberingskonstant [10, s. 268].

3.3.1 Strukturell identifierbarhet

För att kunna använda en skapad modell till att göra prediktioner om det biologiska systemet behöver modellens parametrar vara strukturellt identifierbara. Detta innebär att de med hjälp av en viss uppmätt utdata, till exempel koncentrationen av ett protein, kan bestämmas entydigt [12][s.437-442]. Att testa om en modell är strukturellt identifierbar utgör en viktig del av själva modellutformningen och detta bör göras innan modellen anpassas till experimentella data. Den strukturella identifierbarheten beror endast av strukturen hos modellen givet en viss utdata som ska mätas, och inte av den faktiska experimentella datan. I fortsättningen av rapporten benämns en modell som strukturellt identifierbar då dess parametrar är strukturellt identifierbara givet en specifik utdata (i detta fall SUC2).

3.3.2 Modellreducering

För att göra modellerna identifierbara kan de reduceras vilket leder till att antalet parametrar minskas. Detta kan åstadkommas genom antaganden eller logiska resonemang.

Ett sätt är att betrakta en parameter som konstant, exempelvis med hjälp av värden från tidigare studier. Detta sätt att reducera modellen kräver att parametern representerar någonting mät- eller skattbart. Antagandet behöver därför göras först i processen innan andra antaganden görs, eftersom de andra antagandena kan förändra parametern, vilket gör den omätbar.

Ett annat vanligt antagande är antagandet att systemet var i ett ”steady state” innan t_0 då experimentet startades. Att ett visst genuttryck i en cell nått steady state innebär att cellen har anpassat sig efter ett konstant förhållande och koncentrationen av proteinet är därför konstant, det vill säga derivatan är 0.

En tredje metod för reducering är icke-dimensionalisering, det vill säga att variabler skalas om så att deras enheter försvinner [13, kap. 6]. Med denna metod kan icke-observerade variabler skalas om så att deras initialvärden blir 1. Detta är i många fall enkelt att göra men har nackdelen att parametrarna skalas om så att de inte längre motsvarar en biologisk egenskap.

3.4 Parameteruppskattning

En lösning till differentialekvationerna i en ODE-modell kan erhållas förutsatt att vissa parametrar såsom initialvärden och reaktionshastigheter är givna. För att hitta de värden på parametrar som bäst stämmer överens med verkligheten görs en parameteruppskattning, där målet är att hitta lösningen som minimerar kostnadsfunktionen f, det vill säga

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^n} f(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta}\in\mathbb{R}^n} \sum_{i=1}^{m} \|y_i - \hat{y}_i(\boldsymbol{\theta})\|^2 = \min_{\boldsymbol{\theta}\in\mathbb{R}^n} \sum_{i=1}^{m} \|r_i(\boldsymbol{\theta})\|^2 \tag{4}$$

där y_i är den uppmätta proteinkoncentrationen i tidpunkt i, $\hat{y}_i(\boldsymbol{\theta})$ är den simulerade proteinkoncentrationen givet parametrarna $\boldsymbol{\theta}$ i samma tidpunkt och $r_i(\boldsymbol{\theta})$ är residualerna, det vill säga skillnaderna mellan de äkta och de simulerade värdena. Genom att minimera summan av kvadraten på residualerna får vi de parametrar som har störst sannolikhet att ge upphov till datan givet att residualerna är normalfördelade med medelvärde 0 och okänd konstant varians [14, kap. 8.5].

För att göra själva minimeringen används Kvasi-Newton-metoden Broyden–Fletcher–Goldfarb–Shanno (BFGS) som från en startpunkt tar steg i en nedåtstigande riktning tills den hittar ett lokalt minimum [15, kap. 11.2.2].

3.5 Numerisk identifierbarhet

För att undersöka med vilken statistisk säkerhet parametrarna i en modell kan uppskattas genomförs en numerisk identifierbarhetsanalys. Precis som den strukturella identifierbarheten beror den numeriska identifierbarheten av modellstrukturen. Skillnaden är dock att den numeriska identifierbarheten även beror på den experimentella datan. För optimala parametrar $\hat{\boldsymbol{\theta}}$ vid ett lokalt minima i kostnadsfunktionen är en parameter $\hat{\theta}_k$ numeriskt identifierbar om parameterns uppskattade varians $s^2(\hat{\theta}_k)$ och kovarians $cov(\hat{\theta}_k, \hat{\theta}_l)$ har låga värden [12]. Kovariansen beskriver i hur stor grad två parametrar samvarierar. Uppskattade värden på varians och kovarians hos parametrarna är samlade i en kovariansmatris $COV(\hat{\boldsymbol{\theta}})$, där parametrarnas varians befinner sig på diagonalen. Ett sätt att uppskatta en lägre gräns för kovariansmatrisen är att använda Cramer-Raos sats som säger att

$$COV(\hat{\boldsymbol{\theta}}) \geq F^{-1}(\hat{\boldsymbol{\theta}})$$

där $F(\hat{\boldsymbol{\theta}})$ är en matris som kallas ”Fisher information matrix” [12]. I optimeringsmetoder som bygger på Newtons metod, så som BFGS och Gauss-Newton, beräknas en approximation av kostnadsfunktionens Hessian [15]. För kostnadsfunktionen i ekvation (4) går det att approximera Fisher information matrix med hjälp av kostnadsfunktionens Hessian [12]. Denna matris härleds med hjälp av approximationen av Hessianen som beräknas i optimeringsmetoden Gauss-Newton [15] [12] och hur detta görs beskrivs i appendix B. I Kvasi-Newton-metoden BFGS [15], som används i projektet, approximeras Hessianen $H_{\text{BFGS}}(\hat{\boldsymbol{\theta}})$ i den optimala punkten enligt en annan formel. Trots detta används i projektet samma ekvation att approximera Fisher information matrix som den som används för Gauss-Newton. En lägre gräns för kovariansmatrisen för parametrarna uppskattas därmed som

$$COV(\hat{\boldsymbol{\theta}}) \geq s^2 H^{-1}_{\text{BFGS}}(\hat{\boldsymbol{\theta}}),$$

där den skattade variansen hos residualerna ges enligt

$$s^2 = \frac{\sum_{i=1}^{m} (r_i(\hat{\boldsymbol{\theta}}))^2}{m - n} \tag{5}$$

där n är antalet strukturellt identifierbara parametrar [12].

3.6 Residualanalys

I parameteruppskattningen antas att residualerna är normalfördelade med medelvärde noll och okänd konstant varians. För att undersöka om dessa antaganden håller analyseras de standardiserade residualerna $r_{s,i} = (y_i - \hat{y}_i)/s$ där variansen s ges av ekvation (5). De standardiserade residualerna bör vara standardnormalfördelade och det antagandet kan kontrolleras genom att generera en QQ-plot, där Q står för kvantil (”quantile”). I en QQ-plot jämförs fördelningen hos residualerna mot standardnormalfördelningen. Om residualerna är standardnormalfördelade bör punkterna i grafen bilda en rät linje genom origo med lutning 1 [16]. För att ytterligare analysera residualerna kan de plottas mot tiden. Residualerna bör vara slumpmässigt placerade och inte följa någon särskild trend i en sådan graf. Det skulle i så fall kunna tyda på att modellen inte är korrekt formulerad [17].

4 Metod och modellering

Följande kapitel presenterar modellerna som skapats i projektet samt beskriver hur de togs fram. I kapitlet beskrivs även de metoder för parameteruppskattning och validering som användes i projektet.

4.1 Data

Datan i arbetet kommer från jästceller som levt i ett medium med 4 % glukos, där mediet vid t_0 bytts till att innehålla 0,1 % glukos. Mängden SUC2 hos individuella celler har sedan uppmätts med 5-minutersintervall i 97 tidpunkter. För att kunna följa uttrycket av SUC2 i cellerna har det fluorescerande proteinet Yellow Fluorescent Protein (YFP) transformerats in i cellernas DNA vid genen för SUC2. Därmed kommer YFP att uttryckas tillsammans med SUC2. När YFP belyses med UV-ljus kommer det att fluorescera i gult, och det blir därför enkelt att detektera SUC2. En högre koncentration SUC2 ger därmed ett starkare fluorescent ljus. Mer information om framställandet av datan finns i appendix A.

4.2 Modellering av uttrycket av SUC2

Den första modellen gjordes så enkel som möjligt för att sedan omformuleras till mer komplexa modeller med fler parametrar som bättre kunde representera systemet och beskriva den data som insamlats. Baserat på den biologiska bakgrunden formulerades följande krav på modellerna:

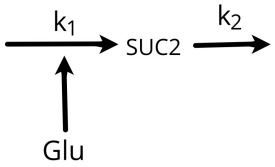
- När glukoskoncentrationen är låg kommer SUC2-koncentrationen att vara hög
- När glukoskoncentrationen är hög kommer koncentrationen av SUC2 att vara låg
- När glukoskoncentrationen är hög kommer inte Snf1 att fosforylera Mig1

3 modeller skapades och namngavs modell 0, 1 respektive X. En modellreducering gjordes för X-modellen vilket gav upphov till två förenklade modeller. Det biologiska systemet antogs i alla modeller vara slutet och lyda massverkans lag.

4.2.1 Modell 0

I modell 0 inkluderades enbart glukoskoncentrationen och SUC2, och inte övriga proteiner i signalvägen (Figur 4). Baserat på systemets villkor att genuttrycket bör vara högt då glukoshalten är låg och vice versa antogs bildningshastigheten av SUC2 vara omvänt proportionell mot glukoshalten. Med hjälp av detta och ekvation (1) blev modellen

$$\frac{d\text{SUC2}}{dt} = \frac{k_1}{\text{Glu}} - k_2 \cdot \text{SUC2}(t) \tag{6}$$



Figur 4: Schematisk bild över hur variablerna i modell 0 påverkar varandra. Här antas bildningshastigheten av SUC2 bero enbart av koncentationen extracellulärt glukos.

där Glu är den yttre glukoskoncentrationen. Modellens initialvillkor är

$$\mathrm{SUC2}(0) = \mathrm{SUC2}_0 \tag{7}$$

där $\mathrm{SUC2}_0$ tas från datan. Glu sattes till 0.001 eftersom koncentrationen glukos i början av experimentet ändrades til 0,1 %.

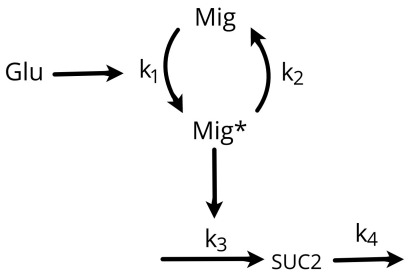
4.2.2 Modell 1

Med modell 1 gjordes ett försök att modellera en större del av Snf1-signalvägen. För detta användes ekvation (2) tillsammans med ekvation (1) vilket med Snf1, Mig1 och SUC2 som tillståndsvariabler gav ekvationerna

$$\begin{aligned}
\frac{d\mathrm{Snf1}^*}{dt} &= k_1 \cdot \frac{1}{\mathrm{Glu}} \cdot \mathrm{Snf1}(t) - k_2 \cdot \mathrm{Snf1}(t)^* \\
\frac{d\mathrm{Mig1}^*}{dt} &= k_3 \cdot \mathrm{Snf1}(t)^* \cdot \mathrm{Mig1}(t) - k_4 \cdot \mathrm{Mig1}(t)^* \\
\frac{d\mathrm{SUC2}}{dt} &= k_3 \cdot \mathrm{Mig1}(t)^* - k_4 \cdot \mathrm{SUC2}(t)
\end{aligned}$$

där $\mathrm{Snf1}^*$ och $\mathrm{Mig1}^*$ är de fosforylerade versionerna av proteinerna, Snf1 och Mig1 de ickefosforylerade versionerna och $\mathrm{Snf1}_{tot} = \mathrm{Snf1}^* + \mathrm{Snf1}$, $\mathrm{Mig1}_{tot} = \mathrm{Mig1}^* + \mathrm{Mig1}$. Signalen S från ekvation (2) representeras här av 1/Glu. I ekvationen för Mig1 utgörs S av fosforylerat Snf1, alltså $\mathrm{Snf1}^*$ och i ekvationen för SUC2 utgörs S av $\mathrm{Mig1}^*$.

För att minska antalet okända parametrar och variabler betraktades sedan mellansteget Snf1 som en svart låda och ersattes av den inversa glukoskoncentrationen (Figur 5).



Figur 5: Schematisk bild över hur variablerna i modell 1 påverkar varandra. Här illustreras fosforyleringen av Mig1 som en loop som påverkas av extracellulärt glukos. Fosforylerat Mig1 påverkar i sin tur bildningshastigheten av SUC2.

Då Mig1_{tot} antogs vara konstant sattes $\text{Mig1}_{tot} = 1$ vilket ger att $\text{Mig1} = 1 - \text{Mig1}^*$. Modell 1 blev därmed

$$\begin{aligned} \frac{d\text{Mig1}^*}{dt} &= k_1 \cdot \frac{1}{Glu} \cdot (1 - \text{Mig1}^*(t)) - k_2 \cdot \text{Mig1}^*(t) \\ \frac{d\text{SUC2}}{dt} &= k_3 \cdot \text{Mig1}^*(t) - k_4 \cdot \text{SUC2}(t) \end{aligned} \tag{8}$$

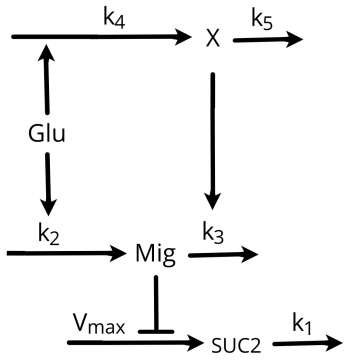
med initialvillkoren

$$\begin{aligned} \text{SUC2}(0) &= \text{SUC2}_0 \\ \text{Mig1}^*(0) &= \text{Mig1}_0^*, \end{aligned} \tag{9}$$

där Mig1_0^* är en parameter.

4.2.3 X-modellen

Då varken modell 0 eller modell 1 passade till den uppmätta datan konstruerades ännu en modell, där en ny tillståndsvariabel introducerades med inspiration från en tidigare studie [18]. Modellen består av tre tillståndsvariabler: SUC2, Mig1 som ger upphov till inhibering utav SUC2, samt en representation av övriga okända påverkningar orsakade av extracellulärt glukos samlade i den nya variabeln X (Figur 6). I denna modell beror ickefosforylerat Mig1 av den nya variabeln X.



Figur 6: Schematisk bild över hur variablerna i X-modellen påverkar varandra. Här visas att Mig1 defosforyleras av extracellulärt glukos och inhiberar SUC2. En okänd faktor X har lagts till som påverkas utav extracellulärt glukos samt påverkar fosforyleringen av Mig1.

Då Mig1 inhiberar SUC2 användes ekvation (3) istället för den loop-beskrivande ekvation (2) som använts i modell 1. ekvation (1) användes för nedbrytningen av X och SUC2. För nedbrytningen av Mig1 användes en modifierad version av ekvationen.

Den kompletta modellen blev därmed

$$\begin{aligned}\frac{d\text{SUC2}}{dt} &= \frac{V_{max}}{K_{\text{Mig1}} + \text{Mig1}(t)^n} - k_1 \cdot \text{SUC2}(t) \\ \frac{d\text{Mig1}}{dt} &= k_2 \cdot \text{Glu} - k_3 \cdot \text{Mig1}(t) \cdot X(t) \\ \frac{dX}{dt} &= k_4 \cdot \text{Glu} - k_5 \cdot X(t)\end{aligned} \tag{10}$$

med initialvillkoren

$$\begin{aligned}\text{SUC2}(0) &= \text{SUC2}_0 \\ \text{Mig1}(0) &= \text{Mig1}_0 \\ X(0) &= X_0\end{aligned} \tag{11}$$

där Mig1_0, X_0 är parametrar.

4.2.4 Reducering av X-modellen

För att göra modellen både strukturellt och numeriskt identifierbar gjordes två reducerade versioner.

Den första ”svagt” reducerade modellen reducerades med två metoder. Först antogs att Hill-koefficienten n i ekvation (10) var konstant. Detta antagande gjordes eftersom Hill-koefficienten för inhiberingen av SUC2 beror av antalet bindningssäten [19], och antagandet görs att detta inte varierar. n valdes till 3 eftersom detta visade sig ge bra stabilitet i optimeringen. Efter det gjordes en icke-dimensionalisering av Mig1 och X. Med antagandet att

$$\text{Mig1}(t) = \widetilde{\text{Mig1}}(t) \cdot \text{Mig1}_0 \tag{12}$$

$$X(t) = \widetilde{X}(t) \cdot X_0 \tag{13}$$

fick vi att

$$\begin{aligned}\frac{d\widetilde{\text{Mig1}}}{dt} &= \frac{k_2}{\text{Mig1}_0} \cdot \text{Glu} - k_3 X_0 \cdot \widetilde{\text{Mig1}}(t) \cdot \widetilde{X}(t) \\ &= \widetilde{k_2} \cdot \text{Glu} - \widetilde{k_3} \cdot \widetilde{\text{Mig1}}(t) \cdot \widetilde{X}(t)\end{aligned} \tag{14}$$

där $\widetilde{k_2} = k_2/\text{Mig1}_0$ och $\widetilde{k_3} = k_3 X_0$, och

$$\begin{aligned}\frac{d\widetilde{X}}{dt} &= \frac{k_4}{X_0} \cdot \text{Glu} - k_5 \cdot \widetilde{X}(t) \\ &= \widetilde{k_4} \cdot \text{Glu} - k_5 \cdot \widetilde{X}(t)\end{aligned} \tag{15}$$

där $\widetilde{k_4} = k_4/X_0$. Den svagt reducerade modellen blev alltså

$$\begin{aligned} \frac{d\text{SUC2}}{dt} &= \frac{\widetilde{V}_{max}}{\widetilde{K}_{\text{Mig1}} + \widetilde{\text{Mig1}}(t)^n} - k_1 \cdot \text{SUC2}(t) \\ \frac{d\widetilde{\text{Mig1}}}{dt} &= \widetilde{k_2} \cdot \text{Glu} - \widetilde{k_3} \cdot \widetilde{\text{Mig1}}(t) \cdot \widetilde{X}(t) \\ \frac{d\widetilde{X}}{dt} &= \widetilde{k_4} \cdot \text{Glu} - k_5 \cdot \widetilde{X}(t) \end{aligned} \tag{16}$$

där $\widetilde{V}_{max} = V_{max}/\text{Mig1}_0^n$ och $\widetilde{K}_{\text{Mig1}} = K_{\text{Mig1}}/\text{Mig1}_0^n$, med initialvillkoren

$$\begin{aligned} \text{SUC2}(0) &= \text{SUC2}_0 \\ \widetilde{\text{Mig1}}(0) &= 1 \\ \widetilde{X}(0) &= 1. \end{aligned} \tag{17}$$

Modellen har sju parametrar $(\widetilde{V}_{max}, \widetilde{K}_{\text{Mig1}}, k_1, \widetilde{k_2}, k_3, \widetilde{k_4}, k_5)$.

Den ”tungt” reducerade modellen togs fram på liknande sätt, men dessutom gjordes ett antagande om steady state för alla tre variabler. Först gjordes samma antagande om n som i den svagt reducerade modellen följt av antagandet i ekvation (13). Sedan gjordes antagandet att

$$\frac{d\widetilde{X}}{dt}(0^-) = 0,$$

det vill säga att X var konstant fram tills att glukoshalten minskades. Detta gav att

$$\frac{k_4}{X_0}\text{Glu}_{0^-} = k_5\widetilde{X}(0^-) = k_5$$

där Glu_{0^-} innebär glukoshalten innan minskning i koncentration. Med samma notation som i ekvation (15) fick vi då att

$$\frac{d\widetilde{X}}{dt} = \widetilde{k_4} \cdot \text{Glu} - \widetilde{k_4} \cdot \text{Glu}_{0^-} \cdot \widetilde{X}(t).$$

På samma sätt icke-dimensionaliserades Mig1 med antagandet från ekvation (12) och sedan gjordes antagandet

$$\frac{d\widetilde{\text{Mig1}}}{dt}(0^-) = 0.$$

Med notationen från ekvation (14) gav det på samma sätt som för $\widetilde{X}$ att

$$\frac{d\widetilde{\text{Mig1}}}{dt} = \widetilde{k_2} \cdot \text{Glu} - \widetilde{k_2} \cdot \text{Glu}_{0^-} \cdot \widetilde{\text{Mig1}}(t) \cdot \widetilde{X}(t).$$

Sist gjordes ett antagande att

$$\frac{d\widetilde{\text{SUC2}}}{dt}(0^-) = 0,$$

vilket gav att

$$k_1 \cdot \mathrm{SUC2}_0 = \frac{V_{max}}{K_{\mathrm{Mig1}} + \mathrm{Mig1}_0^n} = \frac{\widetilde{V}_{max} \cdot \mathrm{Mig1}_0^n}{\widetilde{K}_{\mathrm{Mig1}} \cdot \mathrm{Mig1}_0^n + \mathrm{Mig1}_0^n} = \frac{\widetilde{V}_{max}}{\widetilde{K}_{\mathrm{Mig1}} + 1}.$$

Med samma notation som i ekvation (16) blev den tungt reducerade modellen då

$$\begin{aligned} \frac{d\mathrm{SUC2}}{dt} &= \frac{\widetilde{V}_{max}}{\widetilde{K}_{\mathrm{Mig1}} + \widetilde{\mathrm{Mig1}}(t)^n} - \frac{\widetilde{V}_{max}}{(\widetilde{K}_{\mathrm{Mig1}} + 1)\mathrm{SUC2}_0}\mathrm{SUC2}(t) \\ \frac{d\widetilde{\mathrm{Mig1}}}{dt} &= \widetilde{k_2} \cdot \mathrm{Glu} - \widetilde{k_2} \cdot \mathrm{Glu}_{0^-} \cdot \widetilde{\mathrm{Mig1}}(t) \cdot \widetilde{X}(t) \\ \frac{d\widetilde{X}}{dt} &= \widetilde{k_4} \cdot \mathrm{Glu} - \widetilde{k_4} \cdot \mathrm{Glu}_{0^-} \cdot \widetilde{X}(t) \end{aligned} \tag{18}$$

med initialvilkor

$$\begin{aligned} \mathrm{SUC2}(0) &= \mathrm{SUC2}_0 \\ \widetilde{\mathrm{Mig1}}(0) &= 1 \\ \widetilde{X}(0) &= 1. \end{aligned} \tag{19}$$

Den tungt reducerade modellen har därmed 4 parametrar ($\widetilde{V}_{max}, \widetilde{K}_{\mathrm{Mig1}}, \widetilde{k_2}, \widetilde{k_4}$).

4.3 Strukturell identifierbarhet

För att undersöka strukturell identifierbarhet hos de olika reduktionerna av X-modellen användes det färdiga Wolfram Mathematica-paketet [20] IdentifiabilityAnalysis [21].

4.4 Parameteruppskattning

Parameteruppskattningen gjordes med MATLAB R2018a [22], ”Optimization Toolbox” [23] och ”Statistics and Machine Learning Toolbox” [24]. Här presenteras en övergripande bild över koden som användes för parameteruppskattningen. För detaljer och resten av koden se appendix E.

4.4.1 Modellformulering

För att kunna optimera parametrarna hos modellerna formulerades modellerna som MATLAB-funktioner. Denna formulering genomgick flera iterationer, men landade på en funktionsframställning (se t.ex. `XModel`, appendix E.1.3). Det gjorde att modellerna lätt kunde skickas som `function handle`:s [25, kap. 13]. I grunden tar modellfunktionen parametrar till modellen och ger tillbaka ett `function handle` med högerledet i modellens ODE och initialvilkor till ODE:n. För att öka läsbarheten gavs även `map`:s över reaktanterna och parametrarna i modellen tillbaka av modell-funktionen. Initialvärdet hos reaktanten det finns data för (dvs. SUC2) fördes även in som ett eget argument, vilket motiverades från en simuleringsståndpunkt.

4.4.2 Modellsimulering

Modellerna simulerades med en ODE-lösare i MATLAB. En modell kan få olika kvalitativa beteenden beroende på parametrarna, vilket gjorde att olika lösare behövde användas eftersom lösarvalet beror på ODE:ns beteende [26, kap. 11]. Därför användes både `ode45` och `ode15s`. Ett exempel på denna simulering finns i `simulate_model` (appendix E.2.1). `ode45` användes först då den har högre hastighet och om problemet visade sig vara svårlöst på ett sätt som `ode45` inte kunde hantera användes istället `ode15s`.

För att minimera antalet parametrar som behövde uppskattas användes den första datapunkten i tidsserien som initialvärde för SUC2. Den erhållna lösningen på modellen interpolerades även till de tidpunkter som uppmätts i datan för att göra jämförelser så enkla som möjligt. Dessa två krav gör att alla modellsimuleringar kräver att data finns. I praktiken skapade detta inte några problem, eftersom simuleringarna bara är intressanta i relation till datan.

4.4.3 Parameteroptimering

För att testa modellen användes minstakvadratmetoden på jästpopulationens medelvärde för att ta fram de optimala parametrarna. Modellen simulerades givet vissa parametrar och resultatet jämfördes med datan för att ta fram residualerna. Minimeringsproblemet löstes med `fminunc`, med Kvasi-Newton-metoden BFGS vald som lösningsmetod. Ett exempel på detta finns i `optimize_model_params_average` (appendix E.3.1). Eftersom modellen simulerades flera gånger per optimeringssteg automatiserades ODE-lösarbytet.

4.5 Residualanalys

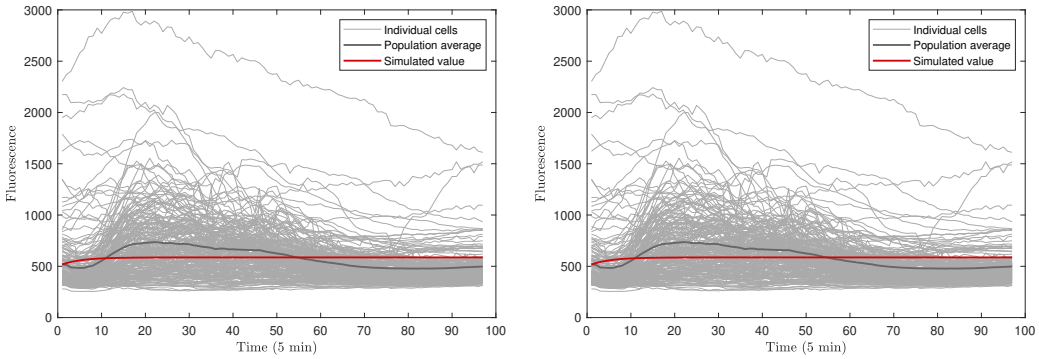
Residualanalys gjordes i MATLAB [22] genom att generera QQ-plottar och plotta residualerna över tid. Mer detaljer om uträkningarna kan ses i appendix E.6.1.

5 Resultat från parameteruppskattningar

Nedan presenteras resultaten från parameteruppskattningarna för de olika modellerna. Där det är av värde presenteras även resultaten från identifierbarhets- och residualanalysen. Fullständiga resultat finns i appendix D. Den exakta koden som använts för att ta fram figurerna och tabellerna finns i `main.m` i appendix E.6.2.

5.1 Modell 0 & 1

Modell 0 ges av ekvation (6) och (7) och modell 1 ges av ekvation (8) och (9). Resultatet av simuleringen visar att modellerna inte beskriver den experimentellt observerade datan och modellerna analyserades därför inte vidare (Figurerna 7a och 7b).



(a) Modell 0 (b) Modell 1

Figur 7: Simulering av modell 0 och 1. Experimentella fluorescensmätningar för 198 individuella celler över tid. Medelvärdet hos populationen (svart kurva) och simulerade värden (röd kurva) för modell 0 (del a) och modell 1 (del b).

5.2 X-modellen

X-modellen ges av ekvation (10) och (11). Modellen visade sig inte vara strukturellt identifierbar (appendix C.1).

5.3 Den svagt reducerade X-modellen

Den svagt reducerade X-modellen som beskrivs av ekvation (16) och (17) är strukturellt identifierbar (appendix C.2). Modellen har 7 okända parametrar

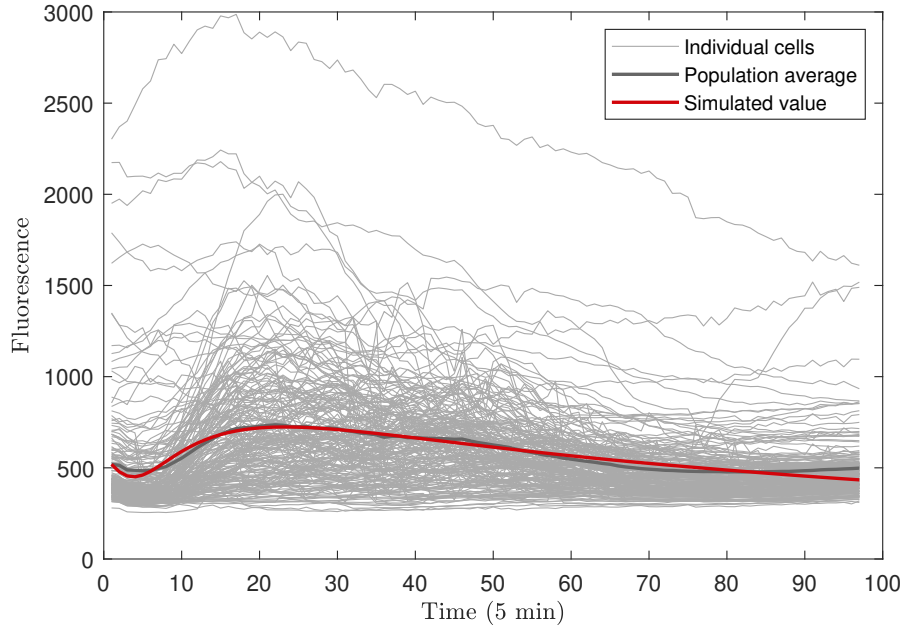
$$\boldsymbol{\theta} = \left[\widetilde{V_{max}}, \widetilde{K_{\text{Mig1}}}, k_1, \widetilde{k_2}, k_3, \widetilde{k_4}, k_5\right].$$

Startgissningar på dessa finns beskrivna i appendix D.3 och de optimala parametrarna ges i Tabell 1.

Tabell 1: Optimala parametrar för den lätt reducerade X-modellen.

Vmax	Kmig	k1	k2	k3	k4	k5
10.1811	-0.1268	0.1175	2.0104	0.2619	3.2223	0.5196

Resultatet av simuleringen visar att modellen beskriver den experimentellt observerade datan (Figur 8).



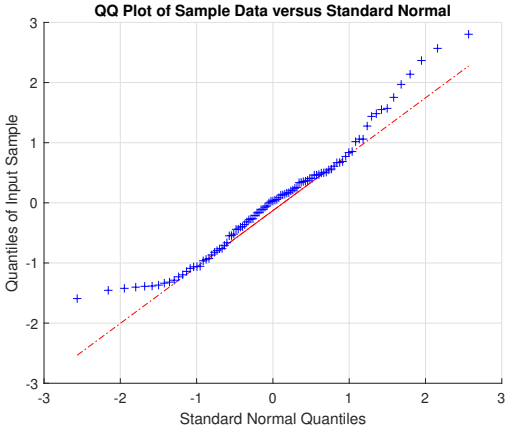
Figur 8: Simulering av den svagt reducerade X-modellen. Experimentella fluorescensmätningar för 198 individuella celler över tid. Medelvärdet hos populationen (svart kurva) och simulerade värden (röd kurva) för den svagt reducerade X-modellen.

5.3.1 Numerisk identifierbarhet

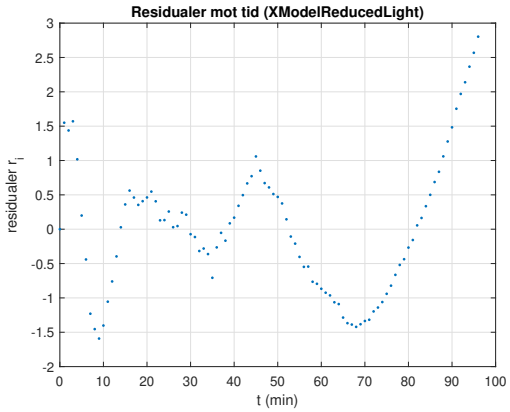
Inversen av den erhållna approximerade Hessianen innehåller negativa värden på diagonalen (appendix D.3). Det innebär att kovariansmatrisen för de uppskattade parametrarna inte kan beräknas och det går inte att avgöra om modellen är numeriskt identifierbar.

5.3.2 Residualanalys

Residualanalysen visar att de största och minsta residualerna avviker från en rät linje på ett sätt som implicerar att deras fördelning är skev åt höger (”positively skewed”) (Figur 9). Det betyder att antagandet om att residualerna är normalfördelade, som görs i parameteruppskattningen, inte är uppfyllt. Ytterligare ges att residualerna inte är slumpmässiga utan istället följer en trend och residualerna har inte heller konstant varians (Figur 10).



Figur 9: QQ-plot av de standardiserade residualerna för den svagt reducerade X-modellen. Residualerna beskrivs på y-axeln och x-axeln beskriver de förväntade värdena hos residualerna om de skulle vara standardnormalfördelade.



Figur 10: De standardiserade residualerna för den svagt reducerade X-modellen som funktion av tiden.

5.4 Den tungt reducerade X-modellen

Den tungt reducerade X-modellen som beskrivs av ekvation (18) och (19) är strukturellt identifierbar (appendix C.3). Modellen har 4 okända parametrar

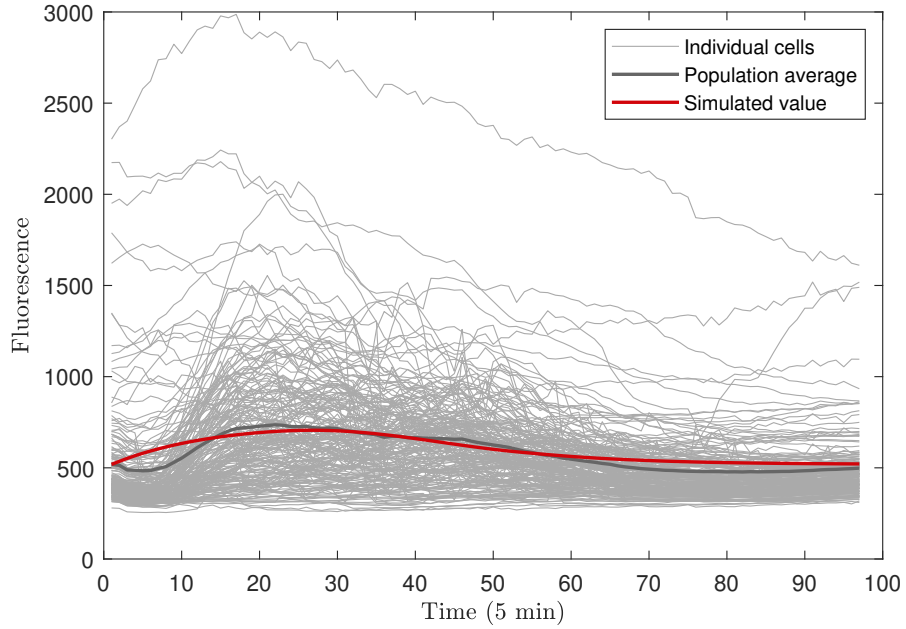
$$\boldsymbol{\theta} = \left[\widetilde{V}_{max}, \widetilde{K}_{\text{Mig1}}, \widetilde{k_2}, \widetilde{k_4}\right].$$

Startgissningar på dessa finns beskrivna i appendix D.4 och de optimala parametrarna ges i Tabell 2.

Tabell 2: Optimala parametrar för den tungt reducerade X-modellen.

Vmax	Kmig	k2	k4
133.1231	2.2794	361.7944	3.7772

Resultatet av simuleringen visar att modellen beskriver den experimentellt observerade datan (Figur 11).



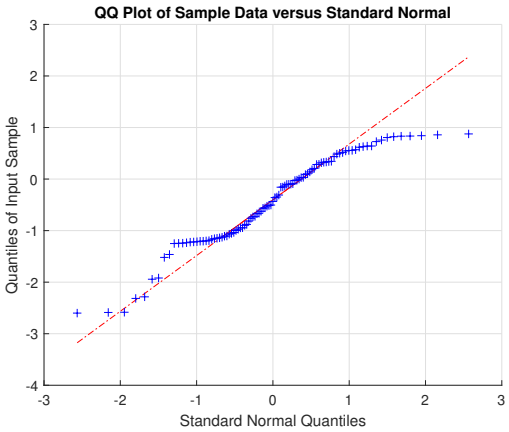
Figur 11: Simulering av den tungt reducerade X-modellen. Experimentella fluorescensmätningar för 198 individuella celler över tid. Medelvärdet hos populationen (svart kurva) och simulerade värden (röd kurva) för den tungt reducerade X-modellen.

5.4.1 Numerisk identifierbarhet

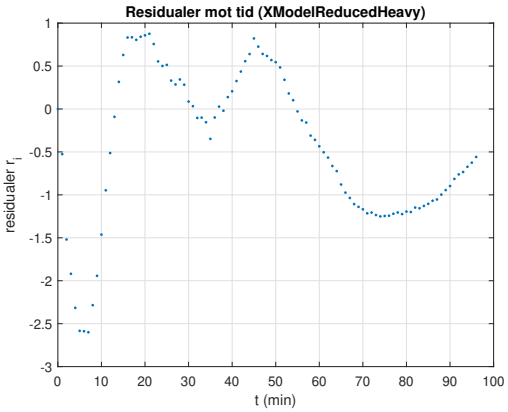
Inversen av den erhållna approximerade Hessianen innehåller positiva värden på diagonalen (appendix D.4) men eftersom negativa värden erhölls för den svagt reducerade var det inte intressant att undersöka huruvida den tungt reducerade X-modellen är numeriskt identifierbar.

5.4.2 Residualanalys

Residualanalysen visar att residualerna avviker från en rät linje på ett sätt som implicerar att deras fördelning har färre extrema värden än förväntat (”light tailed”) (Figur 12). Det betyder att antagandet om att residualerna är normalfördelade, som görs i parameteruppskattningen, inte är uppfyllt. Ytterligare ges att residualerna inte är slumpmässiga utan istället följer en trend och residualerna har inte heller konstant varians (Figur 13).



Figur 12: QQ-plot av de standardiserade residualerna för den tungt reducerade X-modellen. Residualerna beskrivs på y-axeln och x-axeln beskriver de förväntade värdena hos residualerna om de skulle vara standardnormalfördelade.



Figur 13: De standardiserade residualerna för den tungt reducerade X-modellen som funktion av tiden.

6 Diskussion

I detta projekt undersöktes genuttrycket av SUC2 i jäst med hjälp av matematisk modellering. Två reducerade varianter av X-modellen skapades som till skillnad från modell 0 och 1 lyckas beskriva den exprimentellt obseverade datan. När resultaten från simuleringarna av de reducerade X-modellerna jämförs kan det konstateras att modellerna lyckas fånga olika aspekter av hur koncentrationen av SUC2 varierar under experimentet. Den svagt reducerade X-modellen fångar koncentrationsförändringen i början av experimentet men lyckas inte beskriva hur uttrycket av SUC2 stabiliseras i slutet (Figur 8). Den tungt reducerade X-modellen däremot lyckas beskriva stabiliseringen av SUC2 men fångar inte beteendet hos cellerna i början av experimentet (Figur 11). Att den tungt reducerade modellen fångar stabiliseringsbeteendet var förväntat eftersom modellen reducerades med ett antagande om att systemet var stabilt innan. Däremot är det inte klart vad hos steady-stateantagandet som gör att den tungt reducerade modellen inte lyckas beskriva datans beteende i den första fasen.

Residualanalysen av de reducerade X-modellerna ger liknande resultat för båda modellerna. Slutsatsen som kan dras är att modellerna har en del brister. Analysen gav att antagandet om normalfördelade residualer inte är uppfyllt samt att residualerna inte är slumpmässiga utan följer en trend. (Figurerna 9, 10, 12 och 13).

För den svagt reducerade X-modellen hade inversen av den approximerade Hessianen negativa värden på diagonalen. Därmed kunde inte den lägre gränsen för kovariansmatrisen bestämmas eftersom det skulle innebära att parametrarna hade en skattad varians som var negativ. Det går alltså inte att avgöra om parametrarna är numeriskt identifierbara. För den tungt reducerade X-modellen erhölls positiva värden på diagonalen hos inversen av Hessianen. På grund av resultatet från den svagt reducerade X-modellen finns dock en osäkerhet i om Hessianen kan användas för att beräkna kovariansmatrisen. Därför

gjordes inte en numerisk identifierbarhetsanalys hos den tungt reducerade X-modellen heller.

Att den erhållna Hessianen innehöll negativa värden på diagonalen skulle kunna tyda på att kovariansmatrisen borde beräknas med en annan metod. Det framgår inte i [12] om endast den approximation av Hessianen som används i Gauss-Newton-metoden är lämplig att använda för att uppskatta kovariansmatrisen eller om andra approximationer också fungerar bra. Huruvida approximationen av Hessianen som används i Kvasi-Newton-metoden BFGS är lämplig att använda för att approximera en undre begränsning på kovariansmatrisen är därmed inte tydligt. Då både BFGS-Hessianen och Hessianen i Gauss-Newton-metoden används i samma syfte i optimeringsalgoritmerna finns det anledning att tro att BFGS-Hessianen går att använda. Om approximationen av Hessianen som används i BFGS däremot inte är lika lämplig, hade kovariansmatrisen kunnat uppskattas på ett annat sätt. Ett sätt skulle vara att använda en optimeringsalgoritm som uppskattar kovariansmatrisen med hjälp av den approximation av Hessianen som används i Gauss-Newton-metoden.

Vid modellformuleringen gjordes antagandet att Hill-koefficienten n var konstant och hade värdet 3. Värdet som tilldelades n är inte motiverat på annat sätt än att den resulterande modellen gav numeriskt stabila simuleringar. Det exakta värdet n ska fixeras vid bör undersökas vidare. Vilket värde n tilldelas påverkar dock inte det kvalitativa beteendet hos modellen, så detta är främst av biologiskt intresse.

Det genomsnittliga genuttrycket av SUC2 för populationen minskar i början för att sedan öka och nå maxpunkten vid cirka $t = 100$ minuter. Därefter avtar kurvan igen för att stabiliseras efter cirka 140 minuter (Figur 8). Ökningen kort efter glukosskiftet (från 4 % till 0,1 %) är ett förväntat resultat eftersom tillverkning av SUC2 hjälper cellerna vid låg glukoshalt. Därefter förväntades uttrycket av invertaset stabiliseras och hållas konstant eftersom glukoshalten är fortsatt låg (0,1 %) under resten av experimenttiden. Istället sker en minskning av SUC2 efter maximat, vilket båda X-modellerna fångar. Att genuttrycket av SUC2 minskar efter maximat kan bero på att cellen känner av att det inte finns någon sackaros i dess omgivning (glukoshalten i cellen ökar inte igen) och agerar därefter. Om sackaros inte finns tillgängligt behöver cellen utnyttja alternativa kolkällor och dämpar kanske därför uttrycket av SUC2 för att lägga energi på uttrycket av andra gener.

I X-modellen användes tillståndsvariabeln X som en matematisk representation av alla de okända komponenter som kan påverka genuttrycket av SUC2. Det är oklart vad X skulle kunna ha för biologisk betydelse. Då bildningshastigheten av X är direkt proportionell mot glukoskoncentrationen kan X inte representera det fosforylerade Snf1*. Det är heller inte rimligt att X står för det ickefosforylerade Snf1 då nedbrytningshastigheten av Mig1 är direkt proportionell mot X.

Trots att X-modellerna beskriver datan väl kan de inte användas för att göra prediktioner om det biologiska systemet. Detta beror på att det inte kunde avgöras huruvida modellernas parametrar är numeriskt identifierbara. Här finns stor utvecklingspotential.

Ytterligare arbete behövs, kanske i form av andra sätt att bestämma identifierbara parametrar, för att kunna göra prediktion och sluta den systembiologiska cykeln (Figur 1).

För att vidare kunna dra slutsatser om hur signalvägen fungerar kan cellvariabiliteten hos populationen studeras. Då datan som användes i projektet består av mätningar för individuella celler skulle det gå att använda datan för att uppskatta individuella parametrar för cellerna och undersöka skillnader mellan dem [7] [18]. Detta kunde inte göras i projektet eftersom parameteruppskattningen för en del av de individuella cellerna inte kunde genomföras. Det beror på att uttrycket hos vissa celler skiljer sig mycket från populationens medelvärde, och en mer flexibel parameteruppskattningsalgoritm skulle behövas.

7 Slutsats

De två X-modellerna lyckas beskriva olika aspekter av hur uttrycket av SUC2 varierar med tiden. Parametrarna i båda modellerna var strukturellt identifierbara givet SUC2 som utdata men det gick inte att dra några slutsatser om den numeriska identifierbarheten. Därmed går det inte att använda modellerna för att göra prediktioner av det biologiska systemet. Det kan konstateras att det för den aktuella signalvägen fungerade bättre att använda en inhiberingsterm för att beskriva bildningshastigheten av SUC2, som gjordes i X-modellerna, än att använda den ”loop-beskrivande” ekvation som användes i modell 1 eller det omvänt proportionella förhållandet till glukoskoncentrationen som antogs i modell 0.

Referenser

[1] Ljung, Per E and Bornestaf, Cecilia . Användning av försöksdjur i Sverige under 2016; 2018.

[2] Zaky AS, Greetham D, Louis EJ, Tucker GA, Du C. A New Isolation and Evaluation Method for Marine-Derived Yeast spp. with Potential Applications in Industrial Biotechnology. Journal of Microbiology and Biotechnology. 2016 Nov;26(11):1891–1907. Tillgänglig från: https://doi.org/10.4014/jmb.1605.05074.

[3] Bendrioua L, Smedh M, Almquist J, Cvijovic M, Jirstrand M, Goksör M, et al. Yeast AMP-activated Protein Kinase Monitors Glucose Concentration Changes and Absolute Glucose Levels. Journal of Biological Chemistry. 2014 Mar;289(18):12863–12875. Tillgänglig från: https://doi.org/10.1074/jbc.m114.547976.

[4] Salminen A, Kaarniranta K. AMP-activated protein kinase (AMPK) controls the aging process via an integrated signaling network. Ageing Research Reviews. 2012 Apr;11(2):230–241. Tillgänglig från: https://doi.org/10.1016/j.arr.2011.12.005.

[5] Hardie DG. AMPK: A Target for Drugs and Natural Products With Effects on Both Diabetes and Cancer. Diabetes. 2013 Jun;62(7):2164–2172. Tillgänglig från: https://doi.org/10.2337/db13-0368.

[6] Vega M, Riera A, Fernández-Cid A, Herrero P, Moreno F. Hexokinase 2 Is an Intracellular Glucose Sensor of Yeast Cells That Maintains the Structure and Activity of Mig1 Protein Repressor Complex. Journal of Biological Chemistry. 2016 Feb;291(14):7267–7285. Tillgänglig från: https://doi.org/10.1074/jbc.m115.711408.

[7] Welkenhuysen N, Borgqvist J, Backman M, Bendrioua L, Goksör M, Adiels CB, et al. Single-cell study links metabolism with nutrient signaling and reveals sources of variability. BMC Systems Biology. 2017 Jun;11(1). Tillgänglig från: https://doi.org/10.1186/s12918-017-0435-z.

[8] Katze MG. Systems Biology. No. v. 363 in Current Topics in Microbiology and Immunology. Springer; 2013.

[9] Conrad M, Schothorst J, Kankipati HN, Zeebroeck GV, Rubio-Texeira M, Thevelein JM. Nutrient sensing and signaling in the yeastSaccharomyces cerevisiae. FEMS Microbiology Reviews. 2014 Mar;38(2):254–299. Tillgänglig från: https://doi.org/10.1111/1574-6976.12065.

[10] Klipp E. Systems biology in practice: concepts, implementation and application.[Internet]. Wiley-VCH; 2006.

[11] Cheong R, Paliwal S, Levchenko A. Models at the single cell level. WIREs: Systems Biology & Medicine. 2010;2(1):34.

[12] DiStefano III J. Dynamic Systems Biology Modeling and Simulation. Academic Press; 2015.

[13] Murray JD. Mathematical Biology. 3rd ed. Interdisciplinary Applied Mathematics. Springer-Verlag New York; 2002.

[14] Rice JA. Mathematical statistics and data analysis. Duxbury advanced series. Thomson Brooks/Cole; 2007.

[15] Patriksson M, Andréasson N, Evgrafov A, Nedělková Z, Cheong Sou K, Önnheim M. An Introduction to Continuous Optimization - Foundations and Fundamental Algorithms. 3rd ed. Studentlitteratur; 2016.

[16] DeGroot MH, Schervish MJ. Probability and statistics. 4th ed. Pearson Education; 2012.

[17] Rasmuson A, Andersson B, Olsson L, Andersson R. Mathematical modeling in chemical engineering. Cambridge University Press; 2014.

[18] Almquist J, Bendrioua L, Adiels CB, Goksör M, Hohmann S, Jirstrand M. A Nonlinear Mixed Effects Approach for Modeling the Cell-To-Cell Variability of Mig1 Dynamics in Yeast. PLOS ONE. 2015 Apr;10(4):e0124050. Tillgänglig från: `https://doi.org/10.1371/journal.pone.0124050`.

[19] Zeiser S, Liebscher HV, Tiedemann H, Rubio-Aliaga I, Przemeck GK, de Angelis MH, et al. Number of active transcription factor binding sites is essential for the Hes7 oscillator. Theoretical Biology and Medical Modelling. 2006 Feb;3(1). Tillgänglig från: `https://doi.org/10.1186/1742-4682-3-11`.

[20] Inc WR. Mathematica, Version 11.3;. Champaign, IL, 2018.

[21] Karlsson J, Anguelova M, Jirstrand M. An Efficient Method for Structural Identifiability Analysis of Large Dynamic Systems*. IFAC Proceedings Volumes. 2012 Jul;45(16):941–946. Tillgänglig från: `https://doi.org/10.3182/20120711-3-be-2027.00381`.

[22] The MathWorks Inc. MATLAB, version 9.4 (R2018a) [programvara]. 1 Apple Hill Drive Natick, MA; 2018.

[23] The MathWorks Inc. MATLAB Optimization Toolbox, version 8.1 [programvara]. 1 Apple Hill Drive Natick, MA; 2018.

[24] The MathWorks Inc. MATLAB Statistics and Machine Learning Toolbox, version 11.3 [programvara]. 1 Apple Hill Drive Natick, MA; 2018.

[25] MATLAB Programming Fundamentals. 1 Apple Hill Drive Natick, MA; 2019.

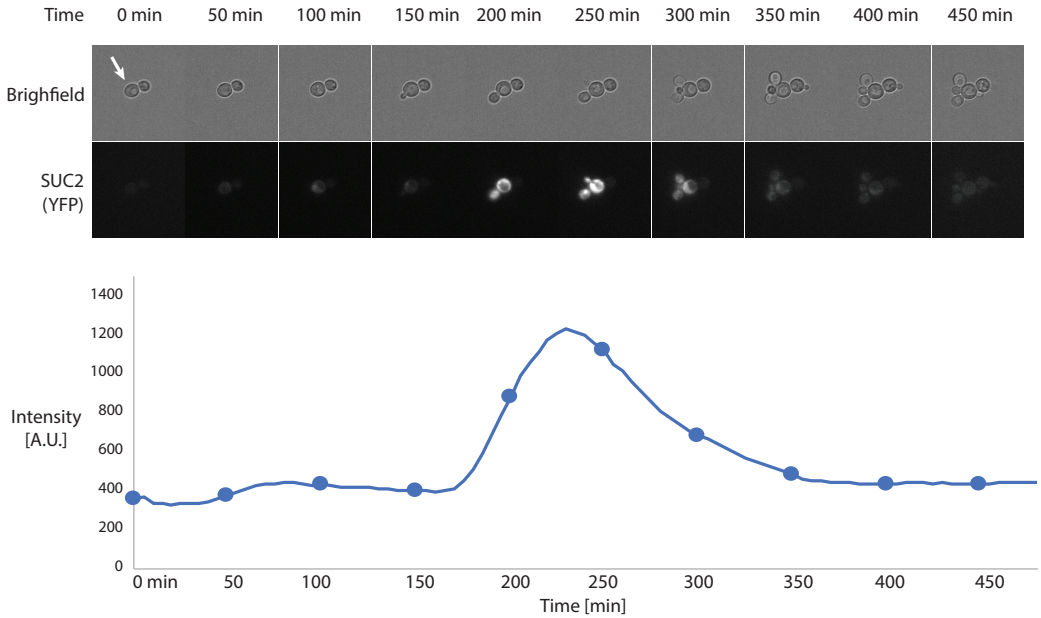
[26] MATLAB Mathematics. 1 Apple Hill Drive Natick, MA; 2019.

[27] Hansen AS, Hao N, O'Shea EK. High-throughput microfluidics to control and measure signaling dynamics in single yeast cells. Nature Protocols. 2015 Jul;10(8):1181–1197. Tillgänglig från: `https://doi.org/10.1038/nprot.2015.079`.

A Dataframställning

Datan som används i projektet är framställd av Niek Welkenhuysen. Datan beskriver hur fluorescensen av YFP förändras över tid i individuella jästceller och är framställd med hjälp av fluorescensmikroskopi och ett mikrofluidiksystem. Ett mikrofluidiksystem består av utrustning som möjliggör att celler kan hållas på plats samtidigt som näringsmedium kan tillföras kontinuerligt genom ett in- och utlopp [27]. Det medför att cellerna kan analyseras genom mikroskopi och att den yttre koncentrationen av glukos (0,1%) kan hållas konstant.

I dataframställningen togs bilder av cellerna var femte minut i 97 tidpunkter. Bilderna togs vid två platser som totalt innehöll 200 individuella celler. Med hjälp av bildanalys bestämdes sedan fluorescensintensiteten av YFP i cellerna. Cellerna kategoriserades även efter cellstorlek mätt i pixlar vid experimentets start. I Figur 14 visas bilderna från några av cellerna vid vissa tidpunkter.



Figur 14: Övre panelen är bilder från ljusfält och fluorescensmikroskopi, där en cell (indikerad av den vita pilen) följs över tiden när den yttre glukoskoncentrationen ändras från 4 % till 0,1 % vid tidpunkt 0. Nedre panelen är spåret av uppmätt intensitet hos cellen. Blå markörer representerar mätningen från bilderna ovan.

Datan är tillgänglig vid förfrågan, niek@chalmers.se. I projektet utelämnades data från två celler eftersom den uppmätta cellstorleken hos dessa celler hade orimliga värden.

B Approximation av Hessianen

För vår kostnadsfunktion (ekvation (4)), där residualerna antas vara normalfördelade med konstant varians σ^2, ges i [12] att Fisher information matrix beskrivs enligt

$$F(\hat{\boldsymbol{\theta}}) = \frac{1}{\sigma^2}\Big(\frac{\partial \hat{\boldsymbol{y}}}{\partial \hat{\boldsymbol{\theta}}}\Big)^T\Big(\frac{\partial \hat{\boldsymbol{y}}}{\partial \hat{\boldsymbol{\theta}}}\Big) \tag{20}$$

$$\Big(\frac{\partial \hat{\boldsymbol{y}}}{\partial \hat{\boldsymbol{\theta}}}\Big) = \begin{pmatrix} \frac{\partial \hat{y}_1}{\partial \hat{\theta}_1} & \cdots & \frac{\partial \hat{y}_1}{\partial \hat{\theta}_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_m}{\partial \hat{\theta}_1} & \cdots & \frac{\partial \hat{y}_m}{\partial \hat{\theta}_n}. \end{pmatrix} \tag{21}$$

För den kostnadsfunktion $f(\hat{\boldsymbol{\theta}})$ som används projektet (ekvation (4)) härleds i [15] följande approximation av Hessianen

$$\nabla^2 f(\hat{\boldsymbol{\theta}}) \approx H(\hat{\boldsymbol{\theta}}) = \Big(\frac{\partial \boldsymbol{r}}{\partial \hat{\boldsymbol{\theta}}}\Big)^T\Big(\frac{\partial \boldsymbol{r}}{\partial \hat{\boldsymbol{\theta}}}\Big) = \Big(\frac{\partial \hat{\boldsymbol{y}}}{\partial \hat{\boldsymbol{\theta}}}\Big)^T\Big(\frac{\partial \hat{\boldsymbol{y}}}{\partial \hat{\boldsymbol{\theta}}}\Big) \tag{22}$$

under antagandet att residualerna är små. Det är därför möjligt att approximera Fisher information matrix och kovariansmatrisen med approximationen av Hessianen (ekvation (22)) [12].

C Strukturell identifierbarhet

Koden i detta appendix är från Wolfram Mathematica-paketet [20] IdentifiabilityAnalysis [21]. I alla modellerna är x_1 =SUC2, x_2 =Mig1 och x_3 =X. x_1 är satt som utdata och $x_1(0)$ är godtyckligt vald.

C.1 X-modellen

Resultat från strukturell identifierbarhetsanalys av X-modellen. Modellen är inte strukturellt identifierbar. Resultatet ges att $\{\theta_1, \theta_3, \theta_4, \theta_5, \theta_7, \theta_8, \theta_9\}$, vilka motsvaras av $\{V_{max}$, $k_2, k_3, k_4, Mig1_0, X_0, K_{Mig1}\}$, är ickeidentifierbara parametrar.

```
Needs["IdentifiabilityAnalysis"];

deq = {

x1'[t] == θ1 / (θ9 + x2[t]^3) - θ2 * x1[t], x1[0] == 0,

x2'[t] == θ3 * 0.001 - θ4 * x2[t] * x3[t], x2[0] == θ7,

x3'[t] == θ5 * 0.001 - θ6 * x3[t], x3[0] == θ8} ;

iad = IdentifiabilityAnalysis [{deq, x1[t]} , {x1, x2, x3} , Table [θi, {i, 9}] , t]

iad["NonIdentifiableParameters"]
```

IdentifiabilityAnalysisData[False, <>]

$\{\theta_1, \theta_3, \theta_4, \theta_5, \theta_7, \theta_8, \theta_9\}$

C.2 Den svagt reducerade X-modellen

Resultat från strukturell identifierbarhetsanalys av den svagt reducerade X-modellen. Modellen är strukturellt identifierbar.

```
Needs["IdentifiabilityAnalysis"];
deq = {
x1'[t] == θ1/ (θ2 + x2[t]^3) - θ3 * x1[t], x1[0] == 0,
x2'[t] == θ4 * 0.001 - θ5 * x2[t] * x3[t], x2[0] == 1,
x3'[t] == θ6 * 0.001 - θ7 * x3[t], x3[0] == 1} ;
iad = IdentifiabilityAnalysis [{deq, x1[t]} , {x1, x2, x3} , Table [θi, {i, 7}] , t]

iad["NonIdentifiableParameters"]
```

IdentifiabilityAnalysisData[True, <>]

{}

C.3 Den tungt reducerade X-modellen

Resultat från strukturell identifierbarhetsanalys av den tungt reducerade X-modellen. Modellen är strukturellt identifierbar.

```
Needs["IdentifiabilityAnalysis"];
deq = {
x1'[t] == θ1/ (θ2 + x2[t]^3) - ((θ1/ (0.02 * (θ2 + 1))) * x1[t]) , x1[0] == 0.02,
```

$$x_2'[t] == \theta_3 * 0.001 - \theta_3 * 0.04 * x_2[t] * x_3[t], x_2[0] == 1,$$

$$x_3'[t] == \theta_4 * 0.001 - \theta_4 * 0.04 * x_3[t], x_3[0] == 1\};$$

$$\text{iad} = \text{IdentifiabilityAnalysis}\left[\{\text{deq}, x_1[t]\}, \{x_1, x_2, x_3\}, \text{Table}\left[\theta_i, \{i, 4\}\right], t\right]$$

iad["NonIdentifiableParameters"]

IdentifiabilityAnalysisData[True, <>]

{}

D Fullständiga resultat från parameteruppskattning

Här presenteras de fullständiga resultaten från parameteruppskattningarna, inklusive de resultat som inte var relevanta för diskussionen.

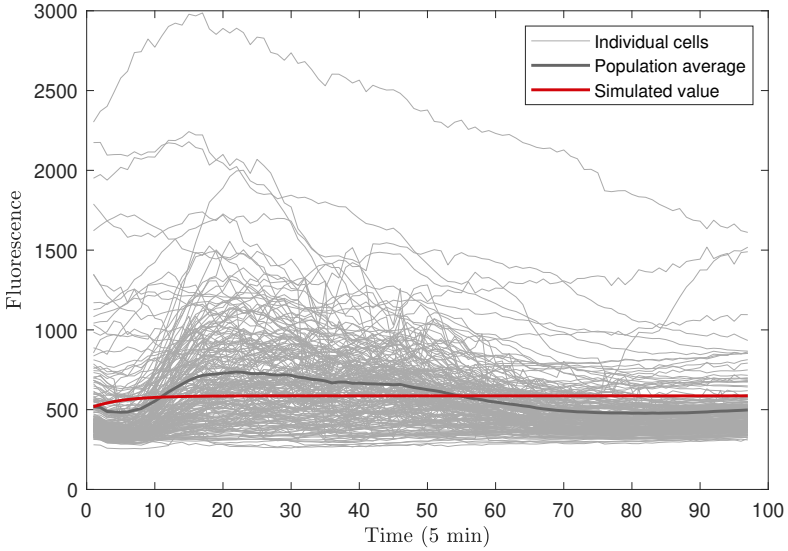
D.1 Modell 0

Tabell 3: Startparametrar för modell 0.

k1	k2
1.0000	1.0000

Tabell 4: Optimala parametrar för modell 0.

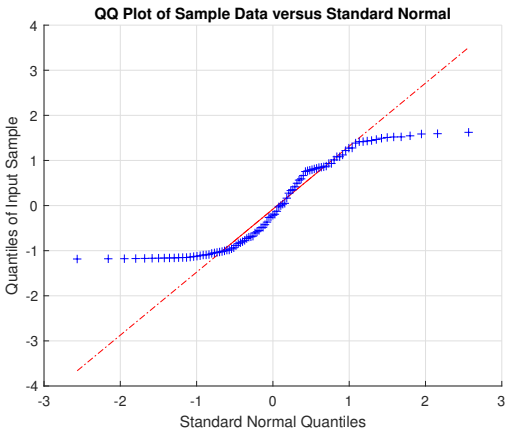
k1	k2
0.1248	0.2127



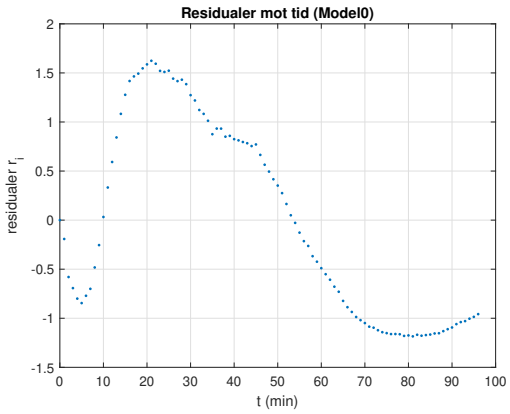
Figur 15: Simulering av modell 0. Experimentella fluorescensmätningar för 198 individuella celler över tid. Medelvärdet hos populationen (svart kurva) och simulerade värden (röd kurva) för den svagt reducerade X-modellen.

Inversen av Hessianen i den optimala punkten blev

$$H^{-1}_{\text{BFGS}}(\hat{\boldsymbol{\theta}}) = \begin{pmatrix} -6.99e-10 & -1.2e-09 \\ -1.2e-09 & -2.05e-09 \end{pmatrix}. \tag{23}$$



Figur 16: QQ-plot av de standardiserade residualerna för modell 0. Residualerna beskrivs på y-axeln och x-axeln beskriver de förväntade värdena hos residualerna om de skulle vara standardnormalfördelade.



Figur 17: De standardiserade residualerna för modell 0 som funktion av tiden.

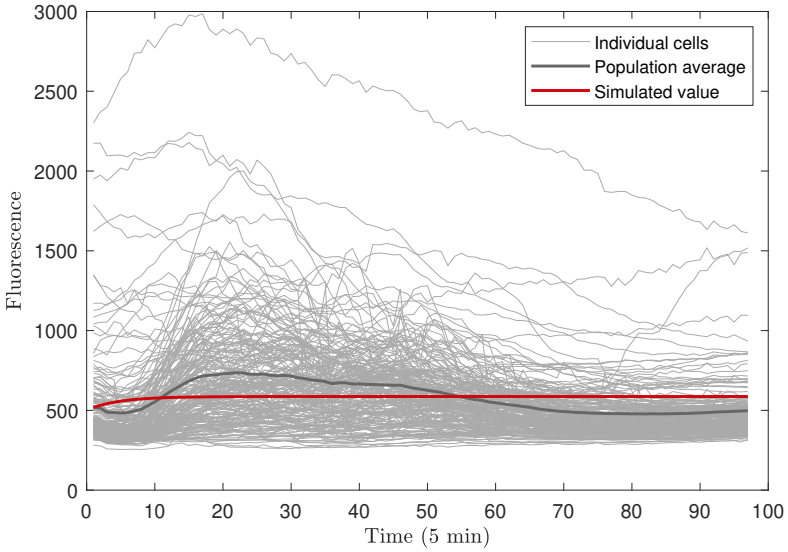
D.2 Modell 1

Tabell 5: Startparametrar för modell 1.

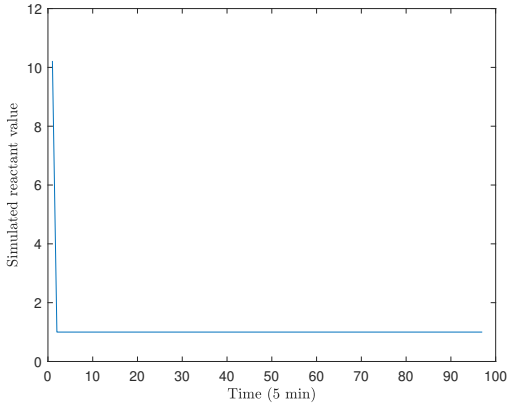
Mig1p0	k1	k2	k3	k4
0.5000	1.0000	1.0000	1.0000	1.0000

Tabell 6: Optimala parametrar för modell 1.

Mig1p0	k1	k2	k3	k4
10.2190	4.9191	0.5408	124.8731	0.2128



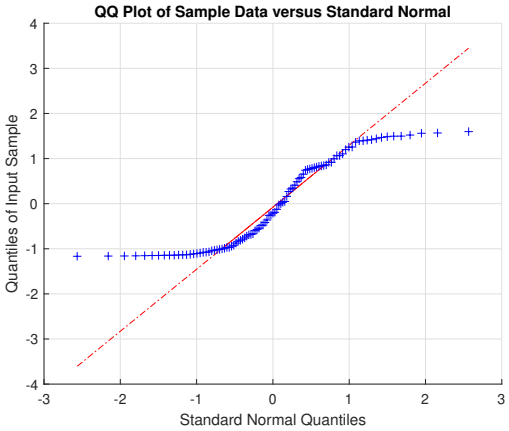
Figur 18: Simulering av modell 1. Experimentella fluorescensmätningar för 198 individuella celler över tid. Medelvärdet hos populationen (svart kurva) och simulerade värden (röd kurva) för den svagt reducerade X-modellen.



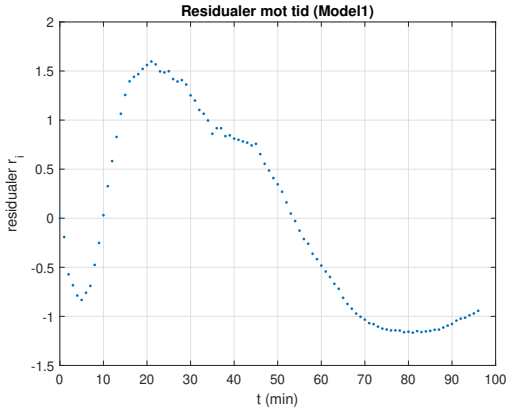
Figur 19: De simulerade värdena av Mig1 efter parameteruppskattning för modell 1.

Inversen av Hessianen i den optimala punkten blev

$$H^{-1}_{\text{BFGS}}(\hat{\boldsymbol{\theta}}) = \begin{pmatrix} 0.00042 & -0.000286 & 1.11e-05 & 0.000431 & 7.36e-07 \\ -0.000286 & 0.000288 & -2.75e-05 & -0.000174 & -2.96e-07 \\ 1.11e-05 & -2.75e-05 & 3.94e-06 & 3.41e-05 & 5.81e-08 \\ 0.000431 & -0.000174 & 3.41e-05 & -0.00439 & -7.52e-06 \\ 7.36e-07 & -2.96e-07 & 5.81e-08 & -7.52e-06 & -1.29e-08 \end{pmatrix}. \tag{24}$$



Figur 20: QQ-plot av de standardiserade residualerna för modell 1. Residualerna beskrivs på y-axeln och x-axeln beskriver de förväntade värdena hos residualerna om de skulle vara standardnormalfördelade.



Figur 21: De standardiserade residualerna för modell 1 som funktion av tiden.

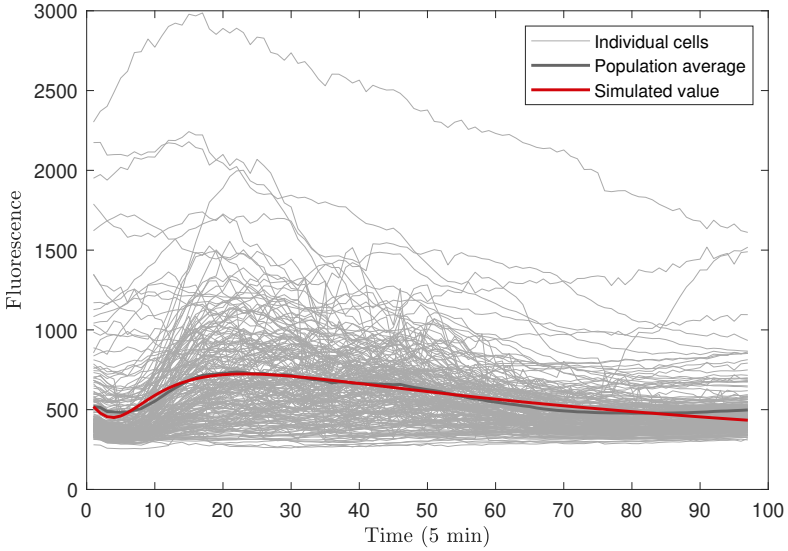
D.3 Den svagt reducerade X-modellen

Tabell 7: Startparametrar för den svagt reducerade X-modellen.

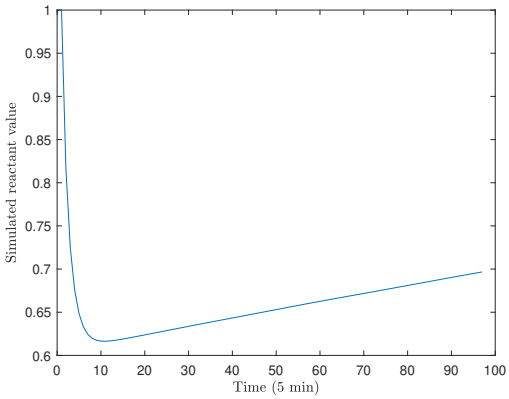
Vmax	Kmig	k1	k2	k3	k4	k5
10.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000

Tabell 8: Optimala parametrar för den svagt reducerade X-modellen.

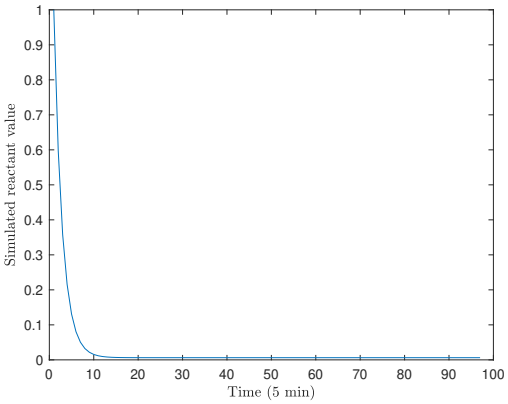
Vmax	Kmig	k1	k2	k3	k4	k5
10.1811	-0.1268	0.1175	2.0104	0.2619	3.2223	0.5196



Figur 22: Simulering av den svagt reducerade X-modellen. Experimentella fluorescensmätningar för 198 individuella celler över tid. Medelvärdet hos populationen (svart kurva) och simulerade värden (röd kurva) för den svagt reducerade X-modellen.



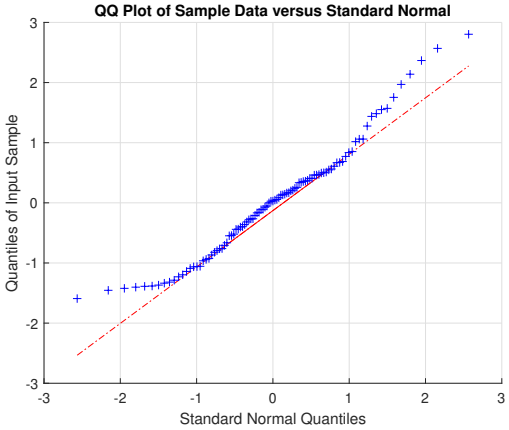
Figur 23: De simulerade värdena av Mig1 efter parameteruppskattning för den svagt reducerade X-modellen.



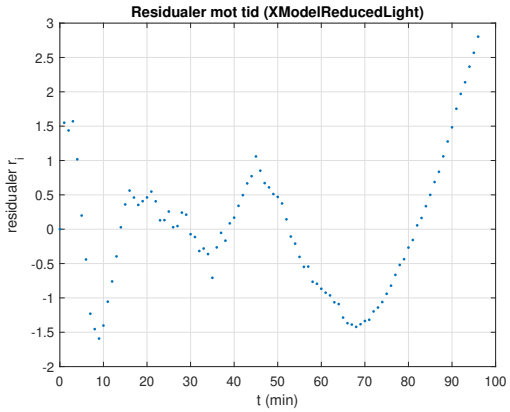
Figur 24: De simulerade värdena av X efter parameteruppskattning för den svagt reducerade X-modellen.

Inversen av Hessianen i den optimala punkten blev

$$H^{-1}_{\text{BFGS}}(\hat{\boldsymbol{\theta}}) = \begin{pmatrix} -7.78e-07 & -6.23e-08 & -9.16e-08 & 3.7e-06 & -7.06e-08 & 1.02e-05 & 3.88e- \\ -6.23e-08 & -1.64e-09 & -1.61e-09 & 1.69e-08 & -1.56e-09 & 4.52e-08 & 2.59e- \\ -9.16e-08 & -1.61e-09 & -2.1e-09 & 6.75e-08 & -1.12e-09 & 1.93e-07 & 1.08e- \\ 3.7e-06 & 1.69e-08 & 6.75e-08 & -2.24e-06 & 3.11e-08 & -5.95e-06 & 2.39e- \\ -7.06e-08 & -1.56e-09 & -1.12e-09 & 3.11e-08 & -1.22e-09 & 9.84e-08 & 5.66e- \\ 1.02e-05 & 4.52e-08 & 1.93e-07 & -5.95e-06 & 9.84e-08 & -1.55e-05 & 8.61e- \\ 3.88e-08 & 2.59e-10 & 1.08e-09 & 2.39e-08 & 5.66e-11 & 8.61e-08 & -1.07e \end{pmatrix} \tag{25}$$



Figur 25: QQ-plot av de standardiserade residualerna för den svagt reducerade X-modellen. Residualerna beskrivs på y-axeln och x-axeln beskriver de förväntade värdena hos residualerna om de skulle vara standardnormalfördelade.



Figur 26: De standardiserade residualerna för den svagt reducerade X-modellen som funktion av tiden.

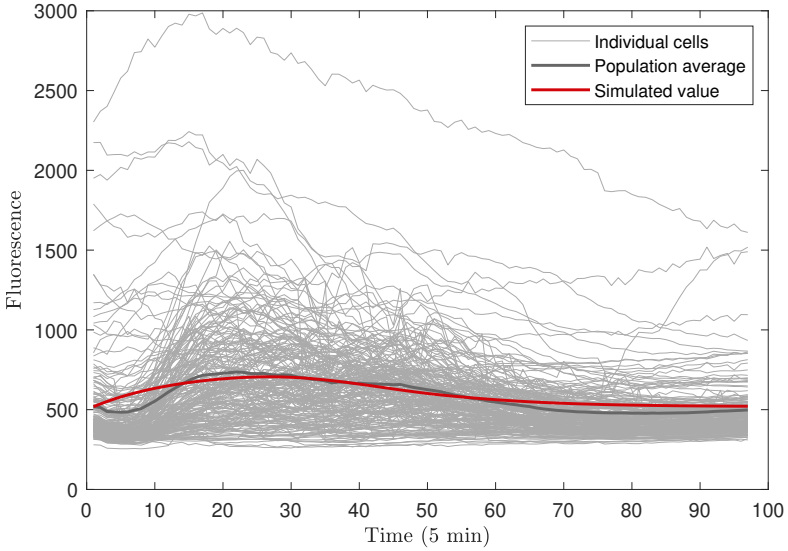
D.4 Den tungt reducerade X-modellen

Tabell 9: Startparametrar för den tungt reducerade X-modellen.

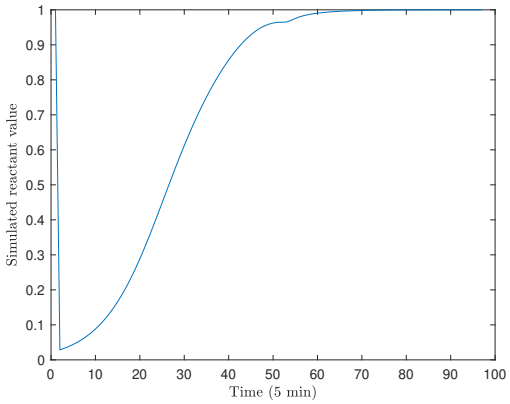
Vmax	Kmig	k2	k4
1.0000	1.0000	1.0000	1.0000

Tabell 10: Optimala parametrar för den tungt reducerade X-modellen.

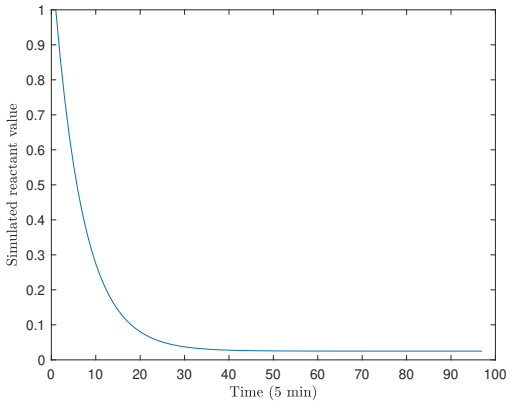
Vmax	Kmig	k2	k4
133.1231	2.2794	361.7944	3.7772



Figur 27: Simulering av den tung reducerade X-modellen. Experimentella fluorescensmätningar för 198 individuella celler över tid. Medelvärdet hos populationen (svart kurva) och simulerade värden (röd kurva) för den svagt reducerade X-modellen.



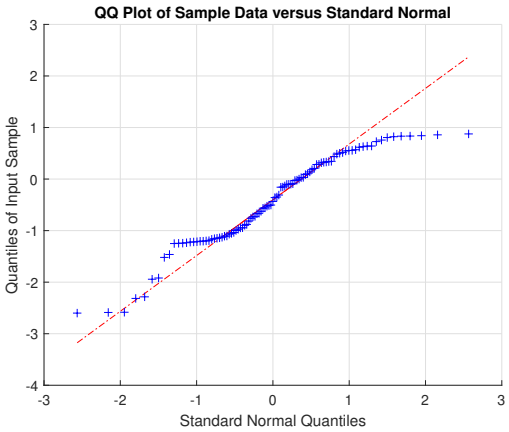
Figur 28: De simulerade värdena av Mig1 efter parameteruppskattning för den tungt reducerade X-modellen.



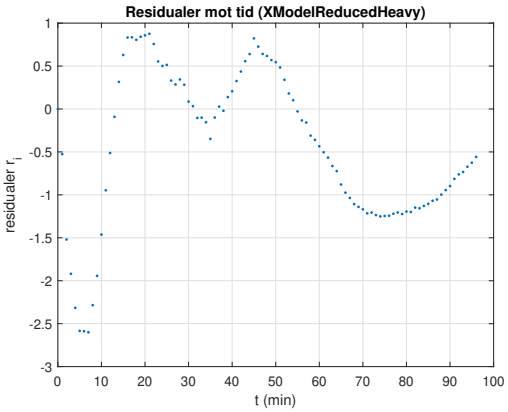
Figur 29: De simulerade värdena av X efter parameteruppskattning för den tungt reducerade X-modellen.

Inversen av Hessianen i den optimala punkten blev

$$H^{-1}_{\text{BFGS}}(\hat{\boldsymbol{\theta}}) = \begin{pmatrix} 0.000974 & 6.8e-06 & -0.000774 & -9.1e-06 \\ 6.8e-06 & 5.77e-08 & -5.54e-06 & -6.95e-08 \\ -0.000774 & -5.54e-06 & 0.0127 & 2.58e-07 \\ -9.1e-06 & -6.95e-08 & 2.58e-07 & 1.21e-07 \end{pmatrix}. \tag{26}$$



Figur 30: QQ-plot av de standardiserade residualerna för den tungt reducerade X-modellen. Residualerna beskrivs på y-axeln och x-axeln beskriver de förväntade värdena hos residualerna om de skulle vara standardnormalfördelade.



Figur 31: De standardiserade residualerna för den tungt reducerade X-modellen som funktion av tiden.

E MATLAB-kod

Här presenteras koden som använts i projektet. All kod finns dessutom tillgänglig på `https://github.com/cvijoviclab/BachelorProject2019`.

I koden hanteras ofta ”mean fluorescence” hos en cell eller population. På olika ställen benämns det som ”fluorescence” alternativt ”mean”, men båda ord är förkortningar för samma sak. Ordet ”average” har använts för att benämna populationens genomsnitt.

E.1 Modeller

Olika modeller som kan parameteruppskattas.

E.1.1 Model0

```
function [ODEfuncHandle, initCond, map, parameterNames] = Model0(parameters
    , Suc20)
%MODEL0 A first primitive model where SNF1 and Mig1 are blackboxed

    % The reactants that are used in the model.
    reactants = {'Suc2'};
    % This creates a map of reactants. Keep as is.
    map = containers.Map(reactants, 1:size(reactants,2));

    % This creates a map of parameters.
    parameterNames = {'k1', 'k2'};

    % Assign names to parameters
    numParams = numel(parameterNames);
    p = array2table(parameters(1:numParams), ...
        'VariableNames', parameterNames);
    Glu = 0.001;

    % The function to be returned
    function dydt = ODE_func(t, y)
```

```
        dydt = zeros(size(reactants, 2), 1);

        % Reactant values that should change in the model.
        dydt(map('Suc2')) = p.k1 ./ Glu -  p.k2 .* y(map('Suc2'));
    end

    ODEfuncHandle = @ODE_func;

    % Enter initial conditions
    initCond = zeros(size(reactants, 2), 1);
    initCond(map('Suc2')) = Suc20;
end
```

E.1.2 Model1

```
function [ODEfuncHandle, initCond, map, parameterNames] = Model1(parameters
    , Suc20)
%MODEL1 One of the first two models

    % The reactants that are used in the model.
    reactants = {'Suc2',...
                 'Mig1p'};
    % This creates a map of reactants. Keep as is.
    map = containers.Map(reactants, 1:size(reactants,2));

    % This creates a map of parameters.
    parameterNames = {'Mig1p0', 'k1', 'k2', 'k3', 'k4'};

    % Assign names to parameters
    numParams = numel(parameterNames);
    p = array2table(parameters(1:numParams), ...
        'VariableNames', parameterNames);
    Glu = 0.001;
    A = 1;

    % The function to be returned
    function dydt = ODE_func(t, y)
        dydt = zeros(size(reactants, 2), 1);

        % Reactant values that should change in the model.
        dydt(map('Mig1p')) = p.k1 ./ Glu .* (A - y(map('Mig1p')))...
            -  p.k2 .* y(map('Mig1p'));
        dydt(map('Suc2')) = p.k3 .* y(map('Mig1p')) - p.k4 .* y(map('Suc2')
            );
    end

    ODEfuncHandle = @ODE_func;

    % Enter initial conditions
    initCond = zeros(size(reactants, 2), 1);
    initCond(map('Suc2')) = Suc20; % Not backed by anything, use data's
        init value
    initCond(map('Mig1p')) = p.Mig1p0;
end
```

E.1.3 XModelReducedLight

```
function [ODEfuncHandle, initCond, map, parameterNames] =
    XModelReducedLight(parameters, Suc20)
%XMODELREDUCEDLIGHT A lightly reduced X-model with constant n
% Nondimensionlization was used to redcue model mildly

    % The reactants that are used in the model.
    reactants = {'Suc2',...
                 'Mig',...
                 'X'};
    % This creates a map of reactants. Keep as is.
    map = containers.Map(reactants, 1:size(reactants,2));

    % This creates a map of parameters.
    parameterNames = {'Vmax', 'Kmig', 'k1', 'k2', 'k3', 'k4', 'k5'};
```

```
14
15      % Assign names to parameters
16      numParams = numel(parameterNames);
17      p = array2table(parameters(1:numParams), ...
18          'VariableNames', parameterNames);
19      n = 3;
20      Mig0 = 1;
21      X0 = 1;
22      Glu = 0.001;
23
24      % The function to be returned
25      function dydt = ODE_func(t, y)
26          dydt = zeros(size(reactants, 2), 1);
27
28          % Reactant values that should change in the model.
29          dydt(map('Suc2')) = p.Vmax ./ (p.Kmig + y(map('Mig')).^n) ...
30              - p.k1 .* y(map('Suc2'));
31
32          dydt(map('Mig')) = p.k2 .* Glu - p.k3 .* y(map('Mig')) .* y(map('X'
               ));
33
34          dydt(map('X')) = p.k4 .* Glu - p.k5 .* y(map('X'));
35      end
36
37      ODEfuncHandle = @ODE_func;
38
39      % Enter initial conditions
40      initCond = zeros(size(reactants, 2), 1);
41      initCond(map('Suc2')) = Suc20;
42      initCond(map('Mig')) = Mig0;
43      initCond(map('X')) = X0;
44  end
```

E.1.4 XModelReducedHeavy

```
1   function [ODEfuncHandle, initCond, map, parameterNames] =
        XModelReducedHeavy(parameters, Suc20)
2   %XMODELREDUCEDHEAVY A heavyliy reduced X-model using steady state
3   % Steady state assumtions were made to reduce the model
4
5       % The reactants that are used in the model.
6       reactants = {'Suc2',...
7                    'Mig',...
8                    'X'};
9       % This creates a map of reactants. Keep as is.
10      map = containers.Map(reactants, 1:size(reactants,2));
11
12      % This creates a map of parameters.
13      parameterNames = {'Vmax', 'Kmig', 'k2', 'k4'};
14
15      % Assign names to parameters
16      numParams = numel(parameterNames);
17      p = array2table(parameters(1:numParams), ...
18          'VariableNames', parameterNames);
19      n = 3;
20      Mig0 = 1;
21      X0 = 1;
22      Glu = 0.001;
23      Glu0m = 0.04;
24
25      % The function to be returned
26      function dydt = ODE_func(t, y)
27          dydt = zeros(size(reactants, 2), 1);
28
29          % Reactant values that should change in the model.
30          dydt(map('Suc2')) = p.Vmax ./ (p.Kmig + y(map('Mig')).^n) ...
31              - p.Vmax ./ (Suc20 .* (p.Kmig + 1)) .* y(map('Suc2'));
32
33          dydt(map('Mig')) = p.k2 .* Glu ...
34              - p.k2 .* Glu0m .* y(map('Mig')) .* y(map('X'));
35
36          dydt(map('X')) = p.k4 .* Glu - p.k4 .* Glu0m .* y(map('X'));
```

```
    end

    ODEfuncHandle = @ODE_func;

    % Enter initial conditions
    initCond = zeros(size(reactants, 2), 1);
    initCond(map('Suc2')) = Suc20;
    initCond(map('Mig')) = Mig0;
    initCond(map('X')) = X0;
end
```

E.2 Simulering

E.2.1 simulate_model

```
function [simFluorescence, simReactants] = simulate_model(Model, ...
    params, measuredName, data)
%SIMULATE_MODEL Summary of this function goes here
%   Detailed explanation goes here

% Get measured fluorescence starting value from data
fluorescenceValues = plotable_fluorescence(data);
fluorescenceAverage = mean(fluorescenceValues, 2);

% Generate the model to use
[ModelODE, initCond, map] = Model(params, fluorescenceAverage(1));

% Use data to get timepoints
timePoints = 1:size(data{1}, 1);

% Simulate the model
[~, simConc] = ode15s(ModelODE, timePoints, initCond);
simFluorescence = simConc(:, map(measuredName));

simReactants = array2table(simConc, 'VariableNames', keys(map));
% Add non-measured reactant
for reactant = keys(map)
    simReactants.(reactant{1}) = simConc(:, map(reactant{1}));
end
end
```

E.3 Parameteruppskattning

E.3.1 optimize_model_params_average

```
function [optimalParameters, optInfo, optParamTable, invHessian] = ...
    optimize_model_params_average(model, initParams, data, meas_reac...
    , initOdeSolver)
%OPTIMIZE_MODEL_PARAMS Optimizes the parameters of a chemical model given
   data.
% This version has a fixed measured reactant in the model.
%
% Args:
%   model: A function representation of a model of the reactions. Should
%       return an ODE function handle, initial conditions and a map of
%       reactant names given some parameters.
%
%   init_params: The initial values of the parameters given to the model.
%
%   data: The data to fit the model to. Should consist of tables containing
%       a field 'Mean' to be compared to the model results in the given
%       reactant, with the tables lying in a cell array. The tables should
%       all have equally long time series.
%
%   meas_reac: The measured reactant from the model that should be compared
%       to the data.
%
```

```
%   initOdeSolver: An ode solver to start with.
%
% Returns:
%   optimal_parameters: The paramaters that gives the model the least
%       square error when compared to the data.
%
%   opt_info: Information about the optimization as given by fminunc.

% Time data is missing, so equal time distribution assumed. The data is
% asumed to have the same size for all time series.
time_points = 1:size(data{1}, 1);

% Get the average mean fluorescence at every time point
mean_average = mean(plotable_fluorescence(data), 2);

% Get the number of reactants
[~, tempInitCond, ~, paramNames] = model(initParams, mean_average(1));
numReac = size(tempInitCond, 2);

% Set options for the ODE solver
simOpt = odeset('NonNegative', ones(numReac, 1), 'Stats', 'off');

% See if odeSolver was given
if nargin < 5
    initOdeSolver = @ode45;
end

defaultSolvers = {@ode45, @ode15s};
startSolver = 1;
% Test if inital solver is in dafault solvers
solverInDefaults = false;
for j = 1:numel(defaultSolvers)
    if isequal(initOdeSolver, defaultSolvers{j})
        solverInDefaults = true;
        % Set so that we begin on inital solver
        startSolver = j;
        break;
    end
end
% Add unique init solver to solvers
if ~solverInDefaults
    solvers = {initOdeSolver, defaultSolvers};
else
    solvers = defaultSolvers;
end

    % A function returning the error of a model given some parameters. The
    % model is simulated once with the average data initial value as
    % initial value, but is compared to each individuals time series.
    function error = model_error(params)
        [ODE_func, init_cond, map] = model(params, mean_average(1));

        allSolversTried = false;
        curSolver = startSolver;
        while ~allSolversTried
            try
                [~, sim_data] = solvers{curSolver}(ODE_func, time_points, ...
                    init_cond, simOpt);

                % Calculate the square error compared to every time series and use
                % the sum as a measurement of the correctness of the parameters.
                error = 0;
                for i = 1:size(data, 2)
                    error = error ...
                        + sum(...
                        (sim_data(:, map(meas_reac)) - data{i}.Mean).^2);
                end

                return
            catch
                disp(func2str(solvers{curSolver}) + " failed");
                curSolver = mod(curSolver, numel(solvers)) + 1;
```

```
                % See if all solvers have been tested
                if curSolver == startSolver
                    allSolversTried = true; %This is unused, make mods safe
                    error = Inf;
                    return
                else
                    disp("Trying " + func2str(solvers{curSolver}));
                end
            end
        end
    end

estOpt = optimoptions(@fminunc, 'Algorithm','quasi-newton', ...
                                'Diagnostics', 'on',...
                                'Display', 'iter');

[optimalParameters, ~, ~, optInfo, ~, hes] = fminunc(@model_error,
    initParams, estOpt);
invHessian = inv(hes);
disp(invHessian);

numParams = numel(paramNames);
optParamTable = array2table(optimalParameters(1:numParams), 'VariableNames'
    , paramNames);
end
```

E.4 Visualisering

E.4.1 plotable_fluorescence

```
function fluorescenceValues = plotable_fluorescence(dataSeries)
%PLOTABLE_FLUORESCENCE Compiles fluorescence for plotting.

numTimePoints = size(dataSeries{1}.Mean, 1);
numTimeSeries = size(dataSeries, 2);

fluorescenceValues = zeros(numTimePoints, numTimeSeries);

for i=1:numTimeSeries
    % Add to sum at every time point
    fluorescenceValues(:, i) = dataSeries{i}.Mean;
end

end
```

E.4.2 visualize_mean

```
function [idvPlot, avgPlot] = visualize_mean(dataSeries, axis)
%VISUALIZE_MEAN

if ~exist('axis','var')
    % get dafault axis
    axis = gca;
end
% Vector to save sum of flourecence at different time points
fluorescenceValues = plotable_fluorescence(dataSeries);

% Define individual color
idvCStr = '#AAAAAA';
idvColor = sscanf(idvCStr(2:end),'%2x%2x%2x',[1 3])/255;

idvPlots = plot(fluorescenceValues, 'Color', idvColor);
idvPlot = idvPlots(1);

fluorescenceAverage = mean(fluorescenceValues, 2);

% Define average color
avgCStr = '#666666';
avgColor = sscanf(avgCStr(2:end),'%2x%2x%2x',[1 3])/255;


hold on
avgPlot = plot(axis, fluorescenceAverage, 'LineWidth', 1.5, ...
```

```
28      'Color', avgColor);
29  hold off
```

E.4.3 visualize_editable_model

```
1   function visualize_editable_model(Model, startParams, measuredName, data)
2   %VISUALIZE_EDITABLE_MODEL Summary of this function goes here
3   %   Detailed explanation goes here
4
5   curParams = startParams;
6   try
7       [~, ~, ~, parameterNames] = Model(startParams, 1); %Only use this for
            map
8       UIon = true;
9   catch ME
10      UIon = false;
11  end
12
13  drawEverything;
14
15      function drawEverything
16      %drawEverything Solves the ODEs for current values and plots them with
17      %controls
18          clf
19
20          % Create axis that wont colide with UI
21          axLeft = axes(gcf,'OuterPosition',[0 0 0.7 1]);
22
23          % Solve the ODE
24          simy = simulate_model(Model, curParams, measuredName, data);
25
26          % Plot the solution
27          visualize_model(simy, data, axLeft);
28          title(func2str(Model));
29
30          % Draw the UI
31          if UIon
32              drawUI;
33          end
34      end
35
36      function drawUI
37      %drawUI Draws the UI elements
38
39          uicontrol(gcf, 'Style', 'pushbutton', 'Callback', @reset, ...
40              'Units','normalized',...
41              'Position', [0.85 ((numel(parameterNames)+2)*0.05) 0.05
                    0.05],...
42              'String', 'Reset')
43          % Draw UI elements for every data field
44          for i = 1:numel(parameterNames)
45
46              % Get the field name
47              var = parameterNames{i};
48              % Height to draw elements at
49              h = (numel(parameterNames)-(i-1))*0.05;
50
51              % Draw title of value-slider
52              uicontrol(gcf,'Style','text','Units','normalized',...
53                  'Position',[0.7 h 0.05 0.05],'String', var + string('='));
54
55              % Draw editable textbox with field value
56              uicontrol(gcf,'Style','edit','Units','normalized',...
57                  'Position',[0.75 h 0.05 0.05],...
58                  'String', curParams(i), ...
59                  'Callback', @(s,e) update(s,e,var));
60
61              % Draw slider that controls field value
62              uicontrol(gcf,'Style', 'slider', 'Min',0,...
63                  'Max',6*startParams(i),...
64                  'Value',curParams(i),'Units','normalized',...
65                  'Position', [0.8 h 0.18 0.05],...
66                  'Callback', @(s,e) update(s,e,i) );
67          end
68
```

```
    end

    function update(source,~,var)
    %update Updates the parameter var with new value and redraws everything

        % Sets parameter to the updated value
        if (source.Style == "slider")
            curParams(var) = source.Value;
        elseif (source.Style == "edit")
            curParams(var) = str2double(source.String);
        end

        % Redraws the figures with new values
        drawEverything
    end

    function reset(src,event)
        curParams = startParams;

        drawEverything;
    end
end
```

E.4.4 visualize_model

```
function plots = visualize_model(simFluorescence, data, axis)
%VISUALIZE_MODEL Visualize model solution given parameters over data

if ~exist('axis','var')
    % get dafault axis
    axis = gca;
end

% Visualize the data
[idvPlot, avgPlot] = visualize_mean(data, axis);

% Define average color
simCStr = '#D2040A';
simColor = sscanf(simCStr(2:end),'%2x%2x%2x',[1 3])/255;

% Plot simulation of model
hold on
simPlot = plot(axis, simFluorescence, 'Color', simColor, 'LineWidth', 1.5);
hold off

xlabel("Time (5 min)",'Interpreter', 'latex');
ylabel("Fluorescence",'Interpreter', 'latex');

plots = [idvPlot, avgPlot, simPlot];
end
```

E.5 Latex-formatering

E.5.1 latexTable

```
function latex = latexTable(input)
% An easy to use function that generates a LaTeX table from a given MATLAB
% input struct containing numeric values. The LaTeX code is printed in the
% command window for quick copy&paste and given back as a cell array.
%
% Author:       Eli Duenisch
% Contributor:  Pascal E. Fortin
% Date:         April 20, 2016
% License:      This code is licensed using BSD 2 to maximize your freedom
    of using it :)
%   ------------------------------------------------------------------------
%   Copyright (c) 2016, Eli Duenisch
%   All rights reserved.
%
%   Redistribution and use in source and binary forms, with or without
```

```
15 %  modification, are permitted provided that the following conditions are
       met:
16 %
17 %  * Redistributions of source code must retain the above copyright notice,
        this
18 %    list of conditions and the following disclaimer.
19 %
20 %  * Redistributions in binary form must reproduce the above copyright
       notice,
21 %    this list of conditions and the following disclaimer in the
       documentation
22 %    and/or other materials provided with the distribution.
23 %
24 %  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS
       IS"
25 %  AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
       THE
26 %  IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
       PURPOSE ARE
27 %  DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE
       LIABLE
28 %  FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
       CONSEQUENTIAL
29 %  DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
       OR
30 %  SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
       HOWEVER
31 %  CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
       LIABILITY,
32 %  OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF
       THE USE
33 %  OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
34 %  ------------------------------------------------------------------------
35 %
36 % Input:
37 % input    struct containing your data and optional fields (details
       described below)
38 %
39 % Output:
40 % latex    cell array containing LaTex code
41 %
42 % Example and explanation of the input struct fields:
43 %
44 % % numeric values you want to tabulate:
45 % % this field has to be a matrix or MATLAB table datatype
46 % % missing values have to be NaN
47 % % in this example we use an array
48 % input.data = [1.12345 2.12345 3.12345; ...
49 %               4.12345 5.12345 6.12345; ...
50 %               7.12345 NaN 9.12345; ...
51 %               10.12345 11.12345 12.12345];
52 %
53 % % Optional fields (if not set default values will be used):
54 %
55 % % Set the position of the table in the LaTex document using h, t, p, b, H
       or !
56 % input.tablePositioning = 'h';
57 %
58 % % Set column labels (use empty string for no label):
59 % input.tableColLabels = {'col1','col2','col3'};
60 % % Set row labels (use empty string for no label):
61 % input.tableRowLabels = {'row1','row2','','row4'};
62 %
63 % % Switch transposing/pivoting your table:
64 % input.transposeTable = 0;
65 %
66 % % Determine whether input.dataFormat is applied column or row based:
67 % input.dataFormatMode = 'column'; % use 'column' or 'row'. if not set '
       column' is used
```

```
%
% % Formatting-string to set the precision of the table values:
% % For using different formats in different rows use a cell array like
% % {myFormatString1,numberOfValues1,myFormatString2,numberOfValues2, ... }
% % where myFormatString_ are formatting-strings and numberOfValues_ are the
% % number of table columns or rows that the preceding formatting-string applies.
% % Please make sure the sum of numberOfValues_ matches the number of columns or
% % rows in input.tableData!
% %
% % input.dataFormat = {'%.3f'}; % uses three digit precision floating point for all data values
% input.dataFormat = {'%.3f',2,'%.1f',1}; % three digits precision for first two columns, one digit for the last
%
% % Define how NaN values in input.tableData should be printed in the LaTex table:
% input.dataNanString = '-';
%
% % Column alignment in Latex table ('l'=left-justified, 'c'=centered,'r'=right-justified):
% input.tableColumnAlignment = 'c';
%
% % Switch table borders on/off:
% input.tableBorders = 1;
%
% % Switch table booktabs on/off:
% input.booktabs = 1;
%
% % LaTex table caption:
% input.tableCaption = 'MyTableCaption';
%
% % LaTex table label:
% input.tableLabel = 'MyTableLabel';
%
% % Switch to generate a complete LaTex document or just a table:
% input.makeCompleteLatexDocument = 1;
%
% % % Now call the function to generate LaTex code:
% latex = latexTable(input);

%%%%%%%%%%%%%%%%%%%%%%%%% Default settings %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% These settings are used if the corresponding optional inputs are not given.
%
% Placement of the table in LaTex document
if isfield(input,'tablePlacement') && (length(input.tablePlacement)>0)
    input.tablePlacement = ['[',input.tablePlacement,']'];
else
    input.tablePlacement = '';
end
% Pivoting of the input data switched off per default:
if ~isfield(input,'transposeTable'),input.transposeTable = 0;end
% Default mode for applying input.tableDataFormat:
if ~isfield(input,'dataFormatMode'),input.dataFormatMode = 'column';end
% Sets the default display format of numeric values in the LaTeX table to '%.4f'
% (4 digits floating point precision).
if ~isfield(input,'dataFormat'),input.dataFormat = {'%.4f'};end
% Define what should happen with NaN values in input.tableData:
if ~isfield(input,'dataNanString'),input.dataNanString = '-';end
% Specify the alignment of the columns:
% 'l' for left-justified, 'c' for centered, 'r' for right-justified
if ~isfield(input,'tableColumnAlignment'),input.tableColumnAlignment = 'c';
    end
```

```
% Specify whether the table has borders:
% 0 for no borders, 1 for borders
if ~isfield(input,'tableBorders'),input.tableBorders = 1;end
% Specify whether to use booktabs formatting or regular table formatting:
if ~isfield(input,'booktabs')
    input.booktabs = 0;
else
    if input.booktabs
        input.tableBorders = 0;
    end
end
% Other optional fields:
if ~isfield(input,'tableCaption'),input.tableCaption = 'MyTableCaption';end
if ~isfield(input,'tableLabel'),input.tableLabel = 'MyTableLabel';end
if ~isfield(input,'makeCompleteLatexDocument'),input.
   makeCompleteLatexDocument = 0;end
if ~isfield(input, 'matrix')
    input.matrix = 0;
else
    if input.matrix
        input.tableBorders = 0;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% process table datatype
if isa(input.data,'table')
  if(~isempty(input.data.Properties.RowNames))
    input.tableRowLabels = input.data.Properties.RowNames';
  end
  if(~isempty(input.data.Properties.VariableNames))
    input.tableColLabels = input.data.Properties.VariableNames';
  end
    input.data = table2array(input.data);
end

% get size of data
numberDataRows = size(input.data,1);
numberDataCols = size(input.data,2);

% obtain cell array for the table data and labels
colLabelsExist = isfield(input,'tableColLabels');
rowLabelsExist = isfield(input,'tableRowLabels');
cellSize = [numberDataRows+colLabelsExist,numberDataCols+rowLabelsExist];
C = cell(cellSize);
C(1+colLabelsExist:end,1+rowLabelsExist:end) = num2cell(input.data);
if rowLabelsExist
    C(1+colLabelsExist:end,1)=input.tableRowLabels';
end
if colLabelsExist
    C(1,1+rowLabelsExist:end)=input.tableColLabels;
end

% obtain cell array for the format
lengthDataFormat = length(input.dataFormat);
if lengthDataFormat==1
    tmp = repmat(input.dataFormat(1),numberDataRows,numberDataCols);
else
    dataFormatList={};
    for i=1:2:lengthDataFormat
        dataFormatList(end+1:end+input.dataFormat{i+1},1) = repmat(input.
           dataFormat(i),input.dataFormat{i+1},1);
    end
    if strcmp(input.dataFormatMode,'column')
        tmp = repmat(dataFormatList',numberDataRows,1);
    end
    if strcmp(input.dataFormatMode,'row')
        tmp = repmat(dataFormatList,1,numberDataCols);
    end
end
if ~isequal(size(tmp),size(input.data))
```

```
194         error(['Please check your values in input.dataFormat:'...
195             'The sum of the numbers of fields must match the number of columns
                    OR rows '...
196             '(depending on input.dataFormatMode)!']);
197     end
198     dataFormatArray = cell(cellSize);
199     dataFormatArray(1+colLabelsExist:end,1+rowLabelsExist:end) = tmp;
200
201     % transpose table (if this switched on)
202     if input.transposeTable
203         C = C';
204         dataFormatArray = dataFormatArray';
205     end
206
207     % make table header lines:
208     if ~input.matrix
209         hLine = '\hline';
210         if input.tableBorders
211             header = ['\begin{tabular}','{|',repmat([input.tableColumnAlignment
                        ,'|'],1,size(C,2)),'}'];
212         else
213             header = ['\begin{tabular}','{',repmat(input.tableColumnAlignment
                        ,1,size(C,2)),'}'];
214         end
215         latex = {header};
216
217         % generate table
218         if input.booktabs
219             latex(end+1) = {'\toprule'};
220         end
221     else
222         latex = {'\begin{pmatrix}'};
223     end
224
225     for i=1:size(C,1)
226         if i==2 && input.booktabs
227             latex(end+1) = {'\midrule'};
228         end
229         if input.tableBorders
230             latex(end+1) = {hLine};
231         end
232         rowStr = '';
233         for j=1:size(C,2)
234             dataValue = C{i,j};
235             if iscell(dataValue)
236               dataValue = dataValue{:};
237             elseif isnan(dataValue)
238               dataValue = input.dataNanString;
239             elseif isnumeric(dataValue)
240               dataValue = num2str(dataValue,dataFormatArray{i,j});
241             end
242             if j==1
243                 rowStr = dataValue;
244             else
245                 rowStr = [rowStr,' & ',dataValue];
246             end
247         end
248         latex(end+1) = {[rowStr,' \\']};
249     end
250
251     if ~input.matrix
252         if input.booktabs
253             latex(end+1) = {'\bottomrule'};
254         end
255
256         % make footer lines for table:
257         tableFooter = {'\end{tabular}'};
258         if input.tableBorders
259             latex = [latex {hLine} tableFooter];
260         else
261             latex = [latex tableFooter];
262         end
263     else
264         tableFooter = {'\end{pmatrix}'};
265         latex = [latex tableFooter];
```

```
end
% add code if a complete latex document should be created:
if input.makeCompleteLatexDocument
    % document header
    latexHeader = {'\documentclass[a4paper,10pt]{article}'};
    if input.booktabs
        latexHeader(end+1) = {'\usepackage{booktabs}'};
    end
    latexHeader(end+1) = {'\begin{document}'};
    % document footer
    latexFooter = {'\end{document}'};
    latex = [latexHeader';latex;latexFooter];
end
% print latex code to console:
disp(char(latex));

end
```

E.5.2 saveLatexFile

```
function saveLatexFile(latexCode, fileName)
%SAVELATEXFILE Saves code from latexTable into fileName
fid=fopen(fileName,'w');
[nrows,ncols] = size(latexCode);
for row = 1:nrows
    fprintf(fid,'%s\n',latexCode{row,:});
end
fclose(fid);
fprintf("\n... your LaTex code has been saved as '" + fileName + "' in your
    working directory\n");

end
```

E.6 Huvudprogram

E.6.1 estimate_model_average

```
function estimate_model_average(Model, data, initParams, initODEsolver)
%ESTIMATE_MODEL_AVERAGE Estimate the model given data and present and save
%results.
measuredName = 'Suc2';

% Optimize parameters
[optParams, ~, optParamTable, invHessian] = optimize_model_params_average
    (...
    Model, initParams, data, measuredName, initODEsolver);

% Plot the optimal solution of the model
[simy, simReac] = simulate_model(Model, optParams, measuredName, data);
figure(1)
visualize_editable_model(Model, optParams, measuredName, data)
title(func2str(Model));

% Make plot to save
averagePlot = figure(2);
modelPlots = visualize_model(simy, data, gca);
legend(modelPlots, {'Individual cells', 'Population average', 'Simulated
    value'}, 'Location', 'northeast')
averagePlot.Position = [400, 300, 600, 400];

% Save plot
filename = func2str(Model) + "_param_opt";
saveas(averagePlot, "Slutrapport/img/" + filename,'epsc')

% Save optimal parameters
latexOptParams.data = optParamTable;
saveLatexFile(latexTable(latexOptParams), ...
```

```
    "Slutrapport/var/" + filename + ".tex")

% Save initial parameters
filename = func2str(Model) + "_param_init";
initParamTable = array2table(initParams, ...
    'VariableNames', optParamTable.Properties.VariableNames);
latexInitParams.data = initParamTable;
saveLatexFile(latexTable(latexInitParams), ...
    "Slutrapport/var/" + filename + ".tex")

% Save inverted hessian
filename = func2str(Model) + "_inv_hessian";
latexInvHessian.data = invHessian;
latexInvHessian.matrix = 1;
latexInvHessian.dataFormat = {'%.3g'};
saveLatexFile(latexTable(latexInvHessian), ...
    "Slutrapport/var/" + filename + ".tex")

% Plot and save the other reactants over time
for reactantCell = simReac.Properties.VariableNames
    reactant = reactantCell{1};
    if ~ strcmp(reactant, 'Suc2')
        reacPlot = figure(6);
        plot(gca, simReac.(reactant));

        xlabel("Time (5 min)", 'Interpreter', 'latex');
        ylabel("Simulated reactant value", 'Interpreter', 'latex');

        % Save plot
        filename = func2str(Model) + "_param_opt_" + reactant;
        saveas(reacPlot, "Slutrapport/img/" + filename, 'epsc')
    end
end

% Generate residuals
fluorescenceValues = plotable_fluorescence(data);
fluorescenceAverage = mean(fluorescenceValues, 2);
y = fluorescenceAverage;

% Standardized residuals
e = y - simy;
e2 = e .^ 2;
numDataPoints = numel(e);

sse = sum(e2);
p = numel(optParams); % SI-parameters in the model. this ex;
    HillmodelReducedLight
s2 = sse / (numDataPoints - p); % estimate of the variance of residuals
s = sqrt(s2);

es = e / s;

% Plot residuals against time.
residualPlot = figure(3);
t = 0:(numDataPoints-1); % (1 time unit corresponds to 5 minutes)
plot(t, es, '.'), grid on
xlabel('t (min)'), ylabel('residualer r_i')
title("Residualer mot tid (" + func2str(Model) + ")")
filename = func2str(Model) + "_res";
saveas(residualPlot, "Slutrapport/img/" + filename,'epsc')

% Plot residuals against yhat
figure(4)
plot(simy,es,'.'), grid on
xlabel('yhat'), ylabel('residualer r_i')
title('Residualer mot yhat')

% Histogram of residuals
figure(5)
histogram(es, numDataPoints)

% Plot qq-plot
```

```
qqPlot = figure(6);
h = qqplot(es);
grid on
filename = func2str(Model) + "_qq";
saveas(qqPlot, "Slutrapport/img/" + filename,'epsc')

% Calculate correlation coefficient from QQ-plot
%figure(6)
X = h.XData;
Y = h.YData;
r = corrcoef(X,Y);

end
```

E.6.2 main

```
% Load data
dataName = 'Suc2_0d1Glc_outlierbyhand.mat';
datastruct = load(dataName);
data = datastruct.Suc2_0d1Glc;

%% Model 0

initParams0 = [1,...    k1
               1 ...    k2
               ];
estimate_model_average(@Model0, data, initParams0, @ode45);

%% Model 1

initParams1 = [0.5,...  Mig1p0
               1,...    k1
               1,...    k2
               1,...    k3
               1 ...    k4
               ];
estimate_model_average(@Model1, data, initParams1, @ode15s);

%% Hill Model Reduced Light

initParamsXl = [10,...  Vmax
                1,...   Kmig
                1,...   k1
                1,...   k2
                1,...   k3
                1,...   k4
                1 ...   k5
                ];
estimate_model_average(@XModelReducedLight, data, initParamsXl, @ode45);

%% Hill Model Reduced Heavy

initParamsXh = [1,...   Vmax
                1,...   Kmig
                1,...   k2
                1 ...   k4
                ];
estimate_model_average(@XModelReducedHeavy, data, initParamsXh, @ode45);
```