Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

Ludvig Stangeland Sæle

Parametrisk usikkerhet i modellbasert regulering

Masteroppgave i Kybernetikk & Robotikk
Veileder: Anders Lyngvi Fougner. Biveileder: Sebastien Gros
Desember 2019

NTNU
Kunnskap for en bedre verden

Ludvig Stangeland Sæle

Parametrisk usikkerhet i modellbasert regulering

Masteroppgave i Kybernetikk & Robotikk
Veileder: Anders Lyngvi Fougner. Biveileder: Sebastien Gros
Desember 2019

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for teknisk kybernetikk

NTNU
Kunnskap for en bedre verden

NTNU
Norges teknisk-naturvitenskapelige universitet

Fakultet for informasjonsteknologi, matematikk og elektroteknikk
Institutt for teknisk kybernetikk

MASTEROPPGAVE

Kandidatens navn: Ludvig Stangeland Sæle

Fag: TTK4900 Teknisk Kybernetikk, Masteroppgave

Oppgavens tittel (norsk): **Parametrisk usikkerhet i modellbasert regulering.**

Oppgavens tittel (engelsk): **Parametric uncertainty in model-based control.**

Oppgavens tekst:

This thesis is affiliated with Artificial Pancreas Trondheim (APT).
In model-based control it is well known that uncertain parameter estimates will give an uncertain model prediction and thereby reduced control performance. Knowledge of this uncertainty can potentially give "worst-case" and "best-case" estimates which can make it possible to design a safer and more robust control system.
In this project, the student will analyse how parametric uncertainty affects the model based calculation of meal insulin in an artificial pancreas (closed-loop glucose control in patients with diabetes).
Specifically, the student will perform the following tasks:

1. Literature review on model based control taking parametric uncertainty into account.
2. For the system to control (i.e. glucose control in diabetes), find nominal parameter values and an appropriate description of their uncertainty. Through model simulation or sensitivity analysis, analyse the impact of these parametric uncertainties.
3. Evaluate the suitability of the methods found in task 1 and implement one or more of these methods.
4. Evaluate the results through testing it on a simulator.

Oppstart: 1. august 2019

Innleveringsfrist: 19. desember 2019

Utført ved Institutt for teknisk kybernetikk

Veileder: Anders Lyngvi Fougner
Biveileder: Sebastien Gros

Trondheim, 1. august 2019

Forord

Denne masteroppgaven er skrevet som en av del av mastergrad innen Kybernetikk & Robotikk ved Norges Teknisk-naturvitenskapelige Universitet, NTNU.

Oppgavebeskrivelsen er tildelt av forskningsgruppen Artificial Pancreas Trondheim, APT, og handler om å finne ut om parametrisk usikkerhet i modellbasert regulering. Denne masteroppgaven er basert på tidligere arbeid fra studenter som har skrevet sine masteroppgaver for APT. Deriblant Hans Erik Frøyen (2014), Nicolay Erlbeck (2016) og Karl Arthur Unstad (2017).
APT-gruppen sørget for å gi meg tidligere relevant MATLAB-kode fra tidligere studenter. Denne MATLAB-koden er en simulator av glukose-/insulin-/glukagon-dynamikken hos diabetikere. APT-simulatoren kan kjøres med 30 "virtuelle pasienter" med parametersett hentet fra en kommersiell simulator fra forskningsmiljøene i Virginia (UVa) og Padova.
Institutt for Teknisk Kybernetikk har sørget for å gi meg en arbeidsplass med tilhørende datamaskin gjennom hele perioden av min masteroppgave.

Min veileder, Anders Lyngvi Fougner, har veiledet meg gjennom semesteret med ukentlige møter. Der relevant bakgrunnslitteratur har blitt gitt i starten av semesteret, samt hans kunnskap om diabetes, og hvilke parametere som antas å være usikre. I tillegg har han bidratt til å finne relevant medveileder, Sebastien Gros. Han har bidratt med sin kunnskap om optimalisering, og hvordan bygge en simulator med verktøyet CasADi.

Med hjelp fra Sebastien Gros ble en ny simulator, samt en sensitivitetsanalyse implementert i MATLAB med utgangspunkt i verktøyet CasADi.

Jeg vil gjerne takke Anders Lyngvi Fougner for hans veiledning gjennom semesteret, og hans forståelse for til tider frustrasjon med oppgaven. Jeg vil også takke Sebastien Gros for hans tålmodighet med utallige besøk og spørsmål rettet mot tekniske problemer og bruk av CasADi.

Ludvig Stangeland Sæle
Trondheim, Desember 2019

Abstract

This master thesis considers the work done with the APT-simulator which Hans Erik Frøyen started on in his master thesis. Nicolay Erlbeck and Karl Arthur Unstad developed this simulator further in their thesis.
This thesis discusses parametric uncertainty in the glucose-insulin model based on existing literature. Together with this it is performed a sensitivity analysis to be able to tell which parameters that can have an affect on the model, and further be defined as uncertain.
The simulator works with a dataset of 30 "virtual patients", and the simulator is further developed such that in the future it can be possible to apply an algorithm for optimization. The further development is based on the optimization tool CasADi. With this tool it is implemented two methods for simulating the system, first the Runge-Kutta method of 4th order, and second the Implicit Runge-Kutta method. In this thesis it was not achieved complete closed loop optimization for the system, however, a description of how to implement this is given.

This master thesis is divided into 10 chapters. Chapter 1 describes the aim of the study and motivation for the study. Former studies, as well as other methods that will be used later is described in Chapter 2. In Chapter 3, a thorough explanation is given of the parameters in the model of this thesis. How the model is designed, and build is described in Chapter 4. In Chapter 5 and 6 is how to perform a sensitivity analysis and the method for the CasADi-simulator described respectively. The results of this master thesis are displayed consecutively to be able to make decisions of further process. Other results and observations that does not influence decisions will be displayed in Chapter 7.
Finally, in Chapter 8 and 9 will alternative solutions together with what that has been achieved be discussed. While Chapter 10 will give an indication of what that needs to be done in future work.
All code that has been used to achieve the functionality in this thesis is appended as a *.zip*-file.

Sammendrag

Denne oppgaven tar i betraktning arbeidet utført på APT-simulatoren som Hans Erik Frøyen startet på i sin masteroppgave. Nicolay Erlbeck og Karl Arthur Unstad videreutviklet denne simulatoren i sine oppgaver.
Oppgaven drøfter parametrisk usikkerhet i glukose-insulin modellen basert på eksisterende litteratur. Samtidig er det utført en sensitivitetsanalyse for å kunne si noe om hvilke parametere som kan ha innvirkning på modellen, og som derfor må bli definert som usikre.
Simulatoren fungerer over et datasett med 30 "virtuelle pasienter", og simulatoren er videreutviklet slik at det i framtiden skal være mulig å utføre en optimaliseringsalgoritme. Videreutviklingen er basert på at optimaliseringsverktøyet CasADi er tatt i bruk. Med dette verktøyet er det blitt utført to metoder for å simulere systemet, først Runge-Kutta 4.ordens metode, og videre Implisitt Runge-Kutta metode. Det ble ikke oppnådd full lukket sløyfe optimalisering for systemet, men en beskrivelse av mulig utførelse er blitt gjort.

Denne oppgaven er delt inn i 10 kapitler. Kapittel 1 beskriver oppgavens mål og formål. Tidligere forskning samt metoder som vil bli brukt senere er beskrevet i Kapittel 2. I Kapittel 3 er det grundigere forklart fokuset rundt parameterne i denne oppgaven. Modellens oppbygning er beskrevet i Kapittel 4. I Kapittel 5 og 6 er henholdsvis metoden for å utføre sensitivitetsanalyse og metoden for CasADi-simulatoren beskrevet. Resultatene i denne oppgaven er presentert fortløpende for å kunne utføre valg om videre prosess. I tillegg vil resultater som ikke påvirker videre beslutninger lagt fram i Kapittel 7.
Tilslutt vil Kapittel 8 og 9 drøfte alternative løsninger samt hva som er fått til i denne oppgaven. Mens Kapittel 10 vil gi en indikasjon på hva som bør gjøres i videre arbeid.
All kode som er brukt for å få fram funksjonalitet i denne oppgaven er vedlagt i en egen *.zip*-fil.

Contents

List of Figures

List of Tables

Nomenclature

AD	Algorithmic differentiation
DAE	Differential Algebraic Equations
MPC	Model Predictive Controller (Modellbasert Prediksjonsregulator)
ODE	Ordinary Differential Equations
PID	Proporsjonal Integrasjon Derivasjon
CasADi	Optimaliseringsverktøy
in silico	Simulering utført på "virtuelle pasienter"
in vivo	Forsøk utført på levende organismer
IRK	Implisitt Runge-Kutta
NLP	Nonlinear Programming (Ulineær Programmering)
RSV	Right Singular Value (Høyre singulærverdi)
SVD	Singular Value Decomposition (Singulærverdi dekomposisjon)
SV	Singular Value (Singulærverdi)
UVa/Padova-simulator	Kommersiell simulator utviklet av forskningsmiljøene i Virginia (UVa) og Padova

1 Introduksjon

Ifølge International Diabetes Federation IDF (2019), [15], er det estimert 463 millioner voksne (20-79 år) med diabetes. Dette har blitt en av de vanligste sykdommene i dagens samfunn. Selve sykdomsbildet og symptomene kan variere fra person til person, som igjen gjør at diabetes er veldig komplisert å behandle. Det finnes flere ulike former for diabetes. De to vanligste formene er Type 1 diabetes, hvor kroppen har problemer med utskilling av insulin, og Type 2 diabetes hvor insulinproduksjonen er delvis nedsatt. I denne oppgaven vil man bare ta utgangspunkt i den medfødte, kroniske versjonen av diabetes, Type 1. Fellesnevner for alle de ulike typene av diabetes er at den naturlig reguleringen av blodsukkernivået slutter å fungere. Blodsukkeret svinger, og man innfører da kunstige metoder for å bistå med reguleringen av blodsukkernivået.
En Artificial Pancreas, *AP*, eller en kunstig bukspyttkjertel, har siden 1960-tallet vært med på å bidra til økt livskvalitet blant pasienter som har fått påvist diabetes. Videre forskning har gjort metodene og teknologien bedre, slik at færre utslag av hyperglykemi eller hypoglykemi forekommer. Dette er de medisinske uttrykkene som beskriver henholdsvis for høyt eller for lavt blodsukker. En kunstig bukspyttkjertel består av tre deler: sensor, insulinpumpe og en reguleringsalgoritme. En undersøkelse, [23], viser at Norge brukte i 2011 rundt 4 milliarder kroner på diabetes. Det er et ønske om å redusere kostnadene rettet mot diabetes, men likevel komme med nye løsninger for å gi en bedre og tryggere hverdag for pasienter med denne sykdommen.
Artificial Pancreas Trondheim, *APT*, har siden 2013 arbeidet i Trondheim, Norge, med et tverrfaglig team av forskere for å oppnå en robust lukket-sløyfe glukoseregulator. En slik regulator vil kunne kontrollere det komplekse systemet i kroppen basert på parametere som en idag ikke har full oversikt over. For å klare å få en bedre oversikt over parameterne i modellen, kan en usikker modell av prosessen bli etablert. Det at reguleringsmetoden håndterer denne usikkerheten vil gjøre metoden robust.

1.1 Hvorfor bytte ut allerede metoder?

Idag finnes mange ulike metoder for å håndtere diabetes, enten det er via insulinpenn eller insulinpumpe. Dette er metoder som fungerer, men som fortsatt krever en manuell regulering av blodsukkeret ved injeksjoner subkutant. En insulinpenn vil kunne føre til ujevn dosering av insulin, men også kjente hudkomplikasjoner som lipohypertrofi, [28], kan finne sted. Dagens insulinpumper kan også føre til komplikasjoner med infeksjon og fortetninger der insulinnålen er satt. Alle pasienter med diabetes må ha individuelt tilpasset behandling, og det vil være vanskelig med dagens metode å regulere nivået helt korrekt. Metoder som ikke er basert på en kunstig bukspyttkjertel vil baserer seg på

pasientens evne til å kontrollere sitt eget behov for insulin. I visse sammenhenger vil kroppen ha en forsinket reaksjon på insulininjeksjonen. Dette kan føre til at pasienter tilfører mer insulin enn det de har behov for, siden de ikke fysisk kjenner umiddelbar effekt, og pasienten kan gå fra ene ytterpunktet til det andre. Slike menneskelige feilkilder kan unngås ved at man innfører algoritmer som kan ha oversikt over de ulike parameterne i kroppen, og regulere glukosenivået deretter. Disse parameterne vil kunne variere både i tid, og fra individ til individ. Med en robust metode for å løse denne problemstillingen vil en forhåpentligvis nærme seg en kunstig bukspyttkjertel til kommersiell bruk med ønsket terapeutisk effekt.

1.2 Tidligere arbeid i APT

APT har siden sin oppstart hatt flere studenter fra Norges teknisk-naturvitenskapelige universitet (NTNU) som har utført sin masteroppgave rettet mot utfordringene rundt diabetes. En av studentene var Hans Erik Frøyen, [12], som utarbeidet en simulator for en glukose-insulin modell. Denne simulatoren baserte seg på en kommersiell simulator utviklet av forskningsmiljøene i Virginia (UVa) og Padova, (UVa/Padova). Denne forskningsgruppen brukte insulin og målinger av glukose via intraperitonale metoder. Oppgaven til Frøyen sammenlignet to ulike regulatorer: en modellbasert regulator *model-predictive-controller*, *MPC* og en modelløs regulator *proporsjonal-integral-derivasjons*, *PID*-regulator. Disse regulatorene vil i denne oppgaven bli undersøkt for videre forbedringer.

Student ved NTNU, Nicolay Erlbeck, videreutviklet arbeidet til Frøyen i sin masteroppgave [9] ved å oppdatere simulatoren. Erlbeck flyttet simulatoren som var implementert i Simulink over til MATLAB for så å oppdatere med de nyeste modellikningene fra UVa/Padova. Mens Frøyen brukte parametere som ikke var like velutviklet, brukte Erlbeck UVa/Padova sine store datasett av parametere. APT sin simulator blir så sammenlignet med UVa/Padova sin simulator. Simuleringer blir gjort av virtuelle pasienter med diabetes Type 1 via et åpent-sløyfe scenario med subkutan injeksjon av insulin.
Erlbeck implementerte en egen APT-simulator i MATLAB. Scriptet som ble kjørt kan kjøre både APT-simulatoren og UVa/Padova simulatoren samtidig.

I 2017 fikk studenten fra NTNU Karl Arthur Unstad, [25], i oppgave å implementere et feilestimat for å ta hånd om feilkilder fra utstyr. Disse feilene inngår i pumpene og sensorene. APT-simulatoren kan nå kjøres med mer reelle verdier.

1.3 Mål med oppgaven

Målet med denne masteroppgaven er å ta steget videre fra oppgaven til Frøyen, via det Erlbeck fant i sin oppgave, og deretter analysere metodens funksjonalitet. Slik at metoden som blir gjennomført er i henhold til det som står beskrevet i oppgaveteksten. I oppgaven til Erlbeck ble det ikke beskrevet en regulator til å administrere insulininjeksjonen, men det blir nevnt som en mulighet for videre utvikling. Denne oppgaven skal ta for seg en robust modellbasert regulator som vil si at den kan håndtere usikre parametere til en usikker modell. Med kunnskap om disse usikkerhetene vil en kunne gjøre regulatoren mer robust basert på "worst case" og "best-case" estimater. Administrasjon av injeksjon ønskes å gjøre så enkelt som mulig, og i vårt tilfelle subkutan injeksjon.

1.4 Motivasjon for studie

Diabetes er en av de vanligste forekommende sykdommer i verden, og hverdagen til pasienten endrer seg vesentlig ved påvisning av sykdommen. Sykdommen er dødelig og ekstrem nøyaktighet kreves av pasienten for å opprettholde et normalt blodsukkernivå. Pasienter pålegges ansvar i administrering av sykdommen som ofte kan oppleves som en belastning og en begrensning i hverdagen.
En kunstig bukspyttkjertel vil overføre store deler av reguleringen av blodsukkernivået til algoritmer og tilhørende utstyr. Dette gjør det enklere for pasienten, og samtidig øke individets livskvalitet. I tillegg vil en kunstig bukspyttkjertel med de rette verdiene gi rom for større trygghet for å sikre rett blodsukkernivå.
Ved utvikling av en robust modell må man også ta hensyn til parametrisk usikkerhet. Ved å undersøke parametrisk usikkerhet kan vi tillate større variasjon og forstyrrelser i modellen uten at dette skal ha innvirkning på en simulator eller en eventuell regulator.
Hovedmotivasjon for utvikling av en velfungerende regulator er å redusere ansvaret for administrering av insulin til pasienten og samtidig øke livskvaliteten til pasienter med diabetes. Utfordring med parametrisk usikkerhet i en regulator kan være flere og samtidig individuelle. Blant annet kan estimering av insulin føre til utfordringer. Underestimering kan føre til at glukosenivået stiger, men at det fortsatt holder seg innenfor kravene, og ingen stor risiko oppstår. Dersom insulinpådraget er overestimert kan det potensielt føre til insulinsjokk og for lavt blodsukker. Reguleringen ønsker å holde seg på en referanse av glukosekonsentrasjon, gitt av APT til å være $4.5mmol/L$. Intervallet skal ideelt holde seg innenfor $4 - 8mmol/L$. Basert på referansen vil dette tyde på at parametrisk usikkerhet kan ha en større risiko for å føre glukosenivået under intervallet enn over intervallet. Derfor bør det bli tatt i betraktning at parametere kan ha "strenge" begrensninger for usikkerhet i en retning, og eventuelt ha

"mykere" begrensning i annen retning.

2 Teori

2.1 Diabetes

I store medisinske leksikon, [3], blir Diabetes Type 1 beskrevet som en kronisk sykdom hvor de insulinproduserende β-cellene i bukspyttkjertelen slutter å virke. Pasienten trenger da tilførsel av hormonet insulin for å kunne opprettholde blodsukkernivået og normal funksjon av kroppens fysiologiske prosesser. I figuren nedenfor, Figur 1, vises den biologiske prosessen som forekommer ved naturlig regulering av kroppens sukkernivå i blodet. Viktigste organene som er med i prosessen er bukspyttkjertel og lever, men også andre organer som hjerne, muskler og vev er med. I tillegg er forholdet mellom hormonene insulin og glukagon en viktig del av produksjonen av glukose, som er sett på som drivstoffet i menneskekroppen.

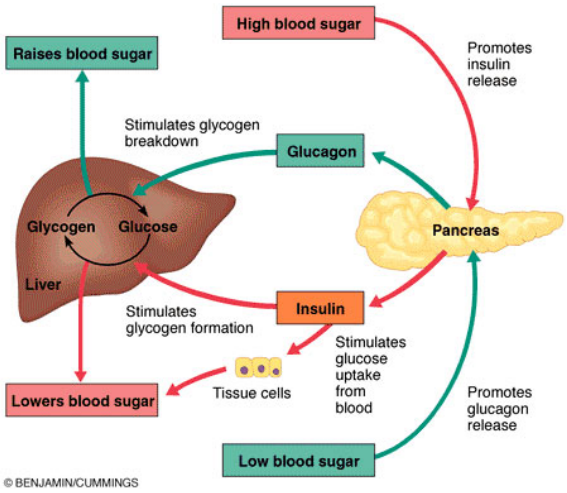


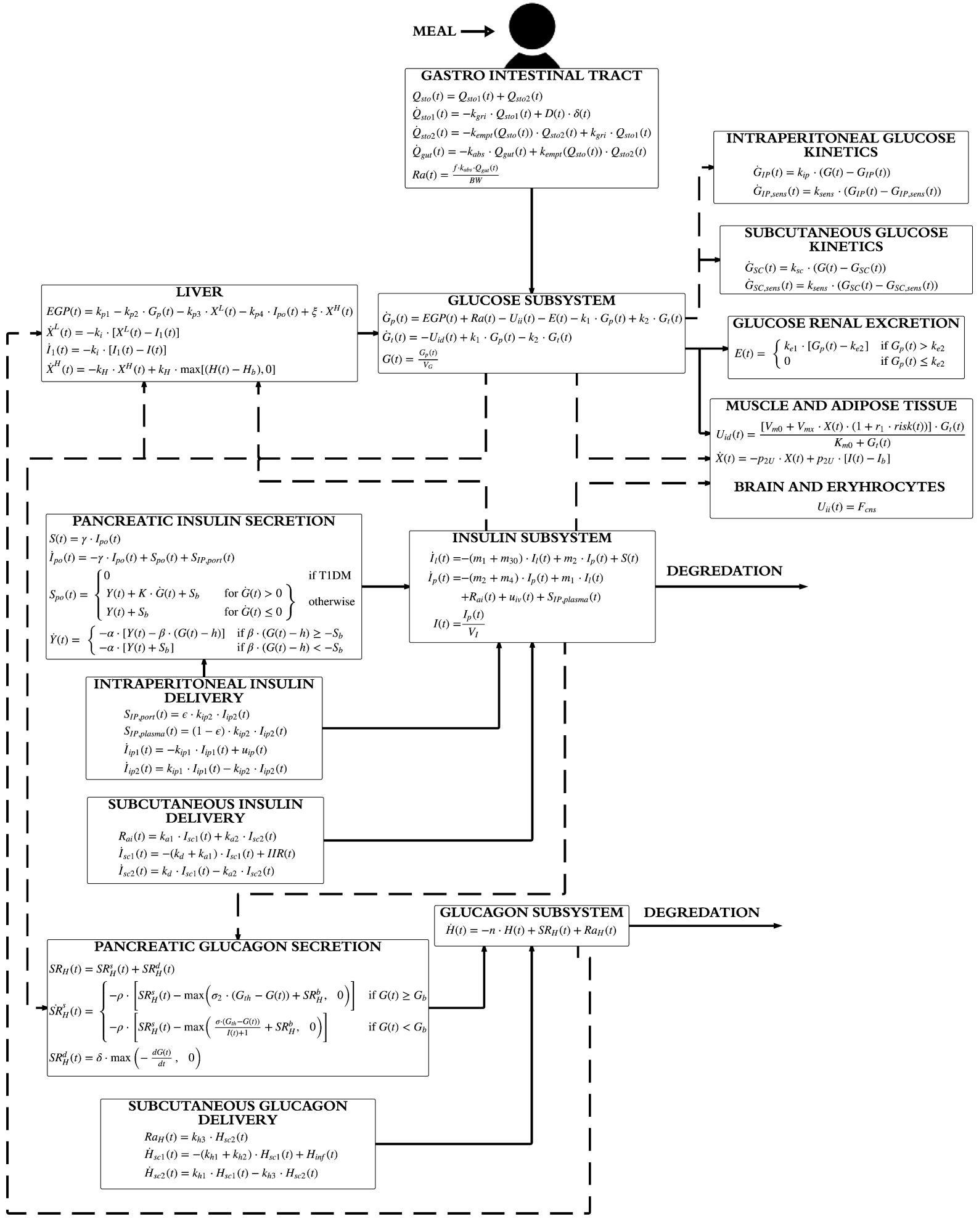
Figure 1: Grunnleggende mekanisme for regulering av glukosenivå i bukspyttkjertel, [1]

Bukspyttkjertelens oppgave er å registrere sukkernivået i blodet, og deretter gi impuls videre til leveren om å senke eller øke blodsukkeret. Systemet er som en lukket sløyfe hvor bukspyttkjertelen fungerer som en regulator. Når blodsukkeret er høyt, fremmer dette produksjon av insulin i de Langeranske øyer i bukspyttkjertelen, og frigivelse av insulin videre til leveren. Når sukkernivået i blodet er for lavt frigis glukagon som fanges opp av bukspyttkjertelen som sender impuls til leveren om å øke sukkernivå. Basert på disse mekanismene klarer systemet å regulere seg selv under normale forhold. Når bukspyttkjertelen ikke lenger produserer insulin vil blodsukkernivå øke ukontrollert, og pasienten får diagnosen diabetes. Diabetes er en dødelig sykdom, men kan kontrolleres ved at pasienten får tilført ekstern insulin eller glukagon for å op-

prettholde normoglykemi. Riktig kosthold er en viktig faktor, men reguleringen må oftest foregå med intravenøst, subkutant eller intraperitonalt tilførsel av insulin.

2.2 Metabolsk modell

I forrige avsnitt ble den naturlige reguleringsprosessen for insulin og glukose beskrevet. Dersom en skal ta i bruk en kunstig regulering av systemet, må flere prosesser i kroppen tas hensyn til. Dette gjelder prosesser i ulike organer som påvirker sammenhengen mellom forbruk og opptak av glukose og insulin. Lever, muskler, hjerne eller ulike vev er blant viktige deler av dette systemet. For å få en komplett oversikt over disse delene lagde Erlbeck en oversiktsskisse over en menneskelig metabolsk modell i sin masteroppgave, [9]. Denne oversiktsskissen er seksjonert inn i deler, der en seksjon kan for eksempel være insulintilførsel, mens en annen gir informasjon om leverens bidrag til den metabolske modellen. Skissen viser hvilke variabler som er avhengig av hverandre, eller overføring fra en tilstand til en annen. Den er i hovedsak delt inn i tre deler. En del som som har ansvar for input til modellen, enten ved insulininjeksjon eller måltid. Andre del består av undersystemene for insulin, glukose og glukagon. Her blir bidrag fra lever, muskler, osv. samlet. Tredje del representerer det som går ut av systemet eller der det forekommer målinger, som blant annet *degradation/nedbrytning* eller *Glukose kinetikk*. En metabolsk modell for insulin og glukose er et nyttig verktøy ved utvikling av medisinsk utstyr og tilretteleggelse av en individualisert behandlingsplan rettet mot pasienter med diabetes. I denne oppgaven blir modellen sammen med alle likningene brukt til å lage en simulator som videre kan bli brukt for å teste regulatoren. Den metabolske modellen består i sin helhet av 25 tilstander sammen med 59 ulike parametere.


MEAL
GASTRO INTESTINAL TRACT
$Q_{sto}(t) = Q_{sto1}(t) + Q_{sto2}(t)$
$\dot{Q}_{sto1}(t) = -k_{gri} \cdot Q_{sto1}(t) + D(t) \cdot \delta(t)$
$\dot{Q}_{sto2}(t) = -k_{empt}(Q_{sto}(t)) \cdot Q_{sto2}(t) + k_{gri} \cdot Q_{sto1}(t)$
$\dot{Q}_{gut}(t) = -k_{abs} \cdot Q_{gut}(t) + k_{empt}(Q_{sto}(t)) \cdot Q_{sto2}(t)$
$Ra(t) = \frac{f \cdot k_{abs} \cdot Q_{gut}(t)}{BW}$
INTRAPERITONEAL GLUCOSE KINETICS
$\dot{G}_{IP}(t) = k_{ip} \cdot (G(t) - G_{IP}(t))$
$\dot{G}_{IP,sens}(t) = k_{sens} \cdot (G_{IP}(t) - G_{IP,sens}(t))$
SUBCUTANEOUS GLUCOSE KINETICS
$\dot{G}_{SC}(t) = k_{sc} \cdot (G(t) - G_{SC}(t))$
$\dot{G}_{SC,sens}(t) = k_{sens} \cdot (G_{SC}(t) - G_{SC,sens}(t))$
LIVER
$EGP(t) = k_{p1} - k_{p2} \cdot G_p(t) - k_{p3} \cdot X^L(t) - k_{p4} \cdot I_{po}(t) + \xi \cdot X^H(t)$
$\dot{X}^L(t) = -k_i \cdot [X^L(t) - I_1(t)]$
$\dot{I}_1(t) = -k_i \cdot [I_1(t) - I(t)]$
$\dot{X}^H(t) = -k_H \cdot X^H(t) + k_H \cdot \max[(H(t) - H_b), 0]$
GLUCOSE SUBSYSTEM
$\dot{G}_p(t) = EGP(t) + Ra(t) - U_{ii}(t) - E(t) - k_1 \cdot G_p(t) + k_2 \cdot G_t(t)$
$\dot{G}_t(t) = -U_{id}(t) + k_1 \cdot G_p(t) - k_2 \cdot G_t(t)$
$G(t) = \frac{G_p(t)}{V_G}$
GLUCOSE RENAL EXCRETION
$E(t) = \begin{cases} k_{e1} \cdot [G_p(t) - k_{e2}] & \text{if } G_p(t) > k_{e2} \\ 0 & \text{if } G_p(t) \le k_{e2} \end{cases}$
MUSCLE AND ADIPOSE TISSUE
$U_{id}(t) = \frac{[V_{m0} + V_{mx} \cdot X(t) \cdot (1 + r_1 \cdot risk(t))] \cdot G_t(t)}{K_{m0} + G_t(t)}$
$\dot{X}(t) = -p_{2U} \cdot X(t) + p_{2U} \cdot [I(t) - I_b]$
BRAIN AND ERYHROCYTES
$U_{ii}(t) = F_{cns}$
PANCREATIC INSULIN SECRETION
$S(t) = \gamma \cdot I_{po}(t)$
$\dot{I}_{po}(t) = -\gamma \cdot I_{po}(t) + S_{po}(t) + S_{IP,port}(t)$
$S_{po}(t) = \begin{cases} 0 & \text{if T1DM} \\ \left.\begin{cases} Y(t) + K \cdot \dot{G}(t) + S_b & \text{for } \dot{G}(t) > 0 \\ Y(t) + S_b & \text{for } \dot{G}(t) \le 0 \end{cases}\right\} & \text{otherwise} \end{cases}$
$\dot{Y}(t) = \begin{cases} -\alpha \cdot [Y(t) - \beta \cdot (G(t) - h)] & \text{if } \beta \cdot (G(t) - h) \ge -S_b \\ -\alpha \cdot [Y(t) + S_b] & \text{if } \beta \cdot (G(t) - h) < -S_b \end{cases}$
INSULIN SUBSYSTEM
$\dot{I}_l(t) = -(m_1 + m_{30}) \cdot I_l(t) + m_2 \cdot I_p(t) + S(t)$
$\dot{I}_p(t) = -(m_2 + m_4) \cdot I_p(t) + m_1 \cdot I_l(t) + R_{ai}(t) + u_{iv}(t) + S_{IP,plasma}(t)$
$I(t) = \frac{I_p(t)}{V_I}$
DEGREDATION
INTRAPERITONEAL INSULIN DELIVERY
$S_{IP,port}(t) = \epsilon \cdot k_{ip2} \cdot I_{ip2}(t)$
$S_{IP,plasma}(t) = (1 - \epsilon) \cdot k_{ip2} \cdot I_{ip2}(t)$
$\dot{I}_{ip1}(t) = -k_{ip1} \cdot I_{ip1}(t) + u_{ip}(t)$
$\dot{I}_{ip2}(t) = k_{ip1} \cdot I_{ip1}(t) - k_{ip2} \cdot I_{ip2}(t)$
SUBCUTANEOUS INSULIN DELIVERY
$R_{ai}(t) = k_{a1} \cdot I_{sc1}(t) + k_{a2} \cdot I_{sc2}(t)$
$\dot{I}_{sc1}(t) = -(k_d + k_{a1}) \cdot I_{sc1}(t) + IIR(t)$
$\dot{I}_{sc2}(t) = k_d \cdot I_{sc1}(t) - k_{a2} \cdot I_{sc2}(t)$
GLUCAGON SUBSYSTEM
$\dot{H}(t) = -n \cdot H(t) + SR_H(t) + Ra_H(t)$
DEGREDATION
PANCREATIC GLUCAGON SECRETION
$SR_H(t) = SR^s_H(t) + SR^d_H(t)$
$\dot{SR}^s_H(t) = \begin{cases} -\rho \cdot \left[SR^s_H(t) - \max\left(\sigma_2 \cdot (G_{th} - G(t)) + SR^b_H,\ 0\right)\right] & \text{if } G(t) \ge G_b \\ -\rho \cdot \left[SR^s_H(t) - \max\left(\frac{\sigma \cdot (G_{th} - G(t))}{I(t)+1} + SR^b_H,\ 0\right)\right] & \text{if } G(t) < G_b \end{cases}$
$SR^d_H(t) = \delta \cdot \max\left(-\frac{dG(t)}{dt},\ 0\right)$
SUBCUTANEOUS GLUCAGON DELIVERY
$Ra_H(t) = k_{h3} \cdot H_{sc2}(t)$
$\dot{H}_{sc1}(t) = -(k_{h1} + k_{h2}) \cdot H_{sc1}(t) + H_{inf}(t)$
$\dot{H}_{sc2}(t) = k_{h1} \cdot H_{sc1}(t) - k_{h3} \cdot H_{sc2}(t)$

Denne skissen er hentet direkte fra Erlbeck sin masteroppgave, [9], og hvis en ønsker forklaring på de ulike undersystemene, er dette beskrevet grundig der. Det som kan være viktig og relevant for denne oppgaven er å se på hvilke ulineariteter som er tilstede i modellen. Ulineariteter kan forekomme basert på listen under:

- if-setninger
- max-funksjoner
- risk-funksjonen
- tilstander multiplisert med hverandre

If-setninger finnes i *Pancreatic glucagon secretion*, *Glucose renal excretion* og *Pancreatic insulin secretion*. Max-funksjoner finnes i modellen i *Liver* og *Pancreatic glucagon secretion*. Risk-funksjonen kommer til syne i *Muscle and Adipose tissue*-delen. Tilstander som er multiplisert er blant annet i *Muscle and Adipose tissue*. Denne modellen kan virke noe uoversiktlig, men i Appendix B er det forsøkt å visualisere modellen på en enklere måte. Den nye modellen i Appendix B er delt inn i tre deler, hvor alle likningene er fjernet. Bare blokker med navn og hvilke tilstander eller variabler som flyter mellom ulike delene i modellen er beskrevet. Det er gir også en enklere visualisering på hvor i systemet det kommer pådrag av enten insulin eller måltid, i tillegg til hvilke nivå en ønsker å gjennomføre målinger.

2.3 CGM

En kunstig bukspyttkjertel er som nevnt basert på tre komponenter: Sensor, pumpe og en algoritme. Sensorens oppgave er å ta målinger av glukosenivået i blodet, for så å gi en tilbakemelding til resten av systemet, om hva som bør gjøres. En slik sensor er basert på kontinuerlige målinger med frekvens på rundt 5 minutter. Denne type sensor kalles en "Contious Glucose Monitor", *CGM*. En slik målemetode medfører at pasienten bærer rundt på måleinstrumentet til enhver tid. Alternativet til dette er at pasienten måler blodsukkernivået med et fingerstikk. Fingerstikkmetoden gir ikke pasienten hyppig nok målinger for blodsukkeret, og konseptet kunstig bukspyttkjertel vil ikke lenger være oppnåelig.
Siden denne oppgaven ikke dreier seg spesifikt om denne målemetoden, vil ikke *CGM* bli mer diskutert. Det kan forøvrig legges merke til at det vil kunne oppstå en støykilde i måleinstrumentet, men dette er ikke en usikkerhet som vil være av stor interesse.

2.4 Eksisterende reguleringsmetoder

I løpet av denne oppgaven skal en regulator bli valgt for så å forsøke å implementere den valgte regulatoren med hensyn til usikre parametere. I tidligere oppgaver som i Frøyen sin oppgave, [12], blir en enkel PID og MPC regulator valgt. I litteratur ser man flere ulike regulatorer bli tatt i bruk, og flere ulike versjoner av disse regulatorene. Når en skal velge regulator må en se på hvor avansert modellen er og hvor avansert regulatoren har behov for å være. Jo mer avansert regulatoren er, jo større datakraft kreves og lengre tid for å gjennomføre kalkuleringer.
Prosessen rundt diabetes og dens modell er ganske kompleks, og det trengs mer avanserte metoder for å regulere blodsukkeret. Ikke alle metoder vil bli evaluert, men et utvalg av metoder tidligere forskning vil bli diskutert i denne oppgaven. Dette gjelder reguleringsdesign som PID, MPC og H_∞. En rekke andre metoder finnes også, som for eksempel *fuzzy based controller* [6] eller *Linear Matrix Inequality approach* [29], men vil ikke bli diskutert i denne oppgaven.

2.4.1 Proportional-integrator-derivator, *PID*

En *Proporsjonal-integrasjon-derivasjon*, PID-regulator, er blant de enkleste regulatorene en kan velge. Denne metoden ønsker å redusere feilen som er gitt av referansen og en feedback sløyfe. Ytterligere beskrivelser vil ikke bli utført her siden det er nøye beskrevet i Frøyen sin oppgave, [12]. Han implementerte denne metoden og sammenlignet den opp mot andre metoder. Når systemet blir mer komplekst og det er ønskelig å introdusere robusthet, vil ikke en enkel PID regulator være tilstrekkelig.

2.4.2 Model Predictive Control, *MPC*

PID eller MPC er kanskje de mest brukte reguleringsmetodene innen dagens prosessindustri. For enkle systemer holder det ofte med en PID-regulator, men når systemet blir mer komplekst kan det være smart å introdusere MPC for å unngå store tidsforsinkelser.
En Model Predictive Controller, MPC, er en metode som gir ut en optimal løsning på et dynamisk optimaliseringsproblem for hvert tidsskritt. Basert på kjennskap til modellen og dens input og begrensinger, klarer en MPC å oppnå en ny sekvens av input. Denne sekvensen blir brukt til å forutse tilstandenes forløp, og hvordan regulere det. En typisk framstilling av optimaliseringsproblemet er gitt under:

$$\min_{w \in R^n} J(w) = \frac{1}{2} \sum_{t=0}^{N-1} x_{t+1}^T Q x_{t+1} + u_t^T R u_t + \epsilon_t^T S \epsilon_t \tag{1a}$$

subject to

$$x_{t+1} = f(x_t, u_t) \tag{1b}$$
$$x_0 = gitt \tag{1c}$$
$$u_0 = gitt \tag{1d}$$
$$x^{min} - \epsilon_t \leq x_t \leq x^{max} + \epsilon_t \tag{1e}$$
$$u^{min} \leq u_t \leq u^{max} \tag{1f}$$
$$\epsilon_t \geq 0 \tag{1g}$$
$$Q > 0 \tag{1h}$$
$$R > 0 \tag{1i}$$
$$S > 0 \tag{1j}$$

Der $J(w)$ er kostfunksjonen som en ønsker å minimalisere. Videre er alle begrensningene til problemet listet fra (1b-1j). Der x og u er henholdsvis tilstandene og input en ønsker å minimalisere. ϵ er slakkvariabler som forteller hvor mye brudd på begrensningene systemet har. Den brukes for å myke opp begrensningene, slik at en optimalisering blir realiserbar. w er en variabel som vil bli beskrevet mer senere, men som samler alle variablene til en vektor. Q, R, S er hvor mye en skal straffe henholdsvis tilstandene, pådrag og slakkvariabler. En kan da tune seg inn til ønsket optimalisering ved å straffe ulikt. Typisk vil S være satt høy og Q lav, for så å tune ved å endre på R.
I Figur 2 ser man hvordan dynamisk optimalisering fungerer. Her er tidshorisonten for optimaliseringen gitt av $t = 10min$, med antall reguleringsintervall på $N = 20$. Dette betyr at for hvert $30.$ sekund gjør modellen en optimalisering. Dette gir $x1$ som et estimat av tilstanden innenfor tidshorisonten, mens u er inputen visualisert med trappetrinnsformat. Inputen er da uforandret i løpet av 30 sekund, og etter det regnes det ut ny input.

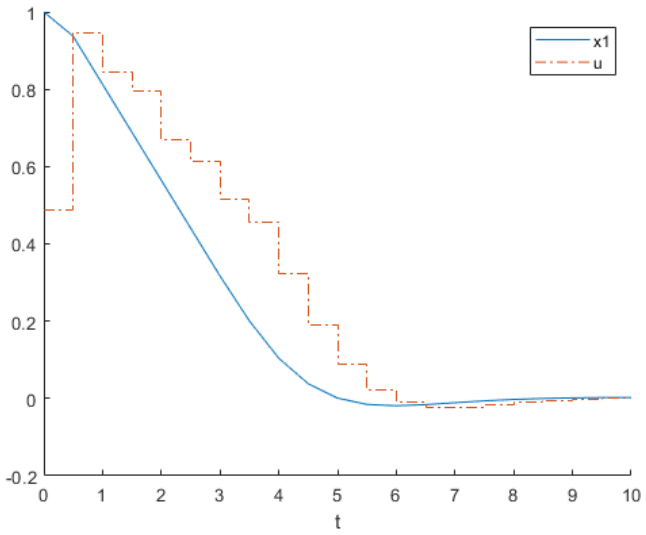


Figure 2: Dynamisk optimalisering, MPC

Figur 2 viser et åpent sløyfe optimaliseringsproblem. For at dette skal bli en MPC må en gjøre dette i lukket sløyfe ved hjelp av tilbakekoblet sløyfe. Det vil si at for hvert tidssteg i modellen, oppnår MPC en ny løsning på et dynamisk optimaliseringsproblem. Dette beskrives av Foss og Heirung, [10], fra 2016 som at MPC har en *moving horizon*. Altså ved at prediksjonshorisonten endrer seg fra $t, ..., t+N$ til $t+1, ..., t+N+1$, osv. I Figur 3 ser man hvordan optimaliseringsproblemet er visualisert via et blokkdiagram i lukket sløyfe.

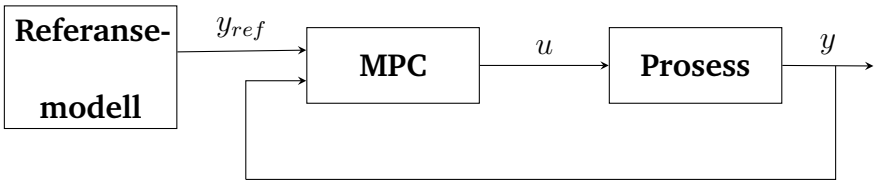


Figure 3: Blokkdiagram for MPC i lukket sløyfe

En MPC er en praktisk regulator som kan fungere i mange ulike sammenhenger. Variasjoner til MPC er stor, og baserer seg på det miljøet det er satt opp mot. Hvis systemet er lineært uten spesielle komplikasjoner, vil en vanlig og enkel MPC fungere. Dersom ulinearitet forekommer, vil funksjonaliteten ekspanderes ytterligere og en vil ha en Nonlinear MPC. Videre har man flere varianter av MPC som tar for seg parametrisk usikkerhet. En Robust MPC er en mulighet, men også litt spesielle og kompliserte varianter som Tubebasert Robust Nonlinear MPC [19]. Dette er versjoner som alle gir god regulering dersom systemet

følger omgivelsene og de begrensningene som regulatoren definerer. Hvorvidt en trenger kompliserte varianter som sistnevnte, vil være basert på modellen og de begrensninger en må jobbe med i hvert enkelt tilfelle.

2.4.3 H_∞-regulering

H_∞ er en metode brukt for regulering som ser på reguleringsproblemet som et matematisk optimaliseringsproblem, for så å finne en regulator til å løse problemet. Denne metoden er fordelaktig når en har systemer med multivariabler som krysskobles, men det krever desto bedre matematisk forståelse for å ta i bruk metoden. Det kreves også en ryddig og god modell for systemet som skal reguleres.
H_∞-metoden baserer seg på at en må sette systemet sitt opp som en standard lukket sløyfe som vist under.

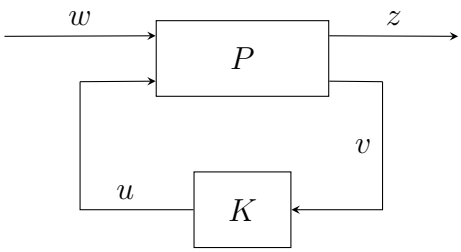


Figure 4: Blokkdiagram for H_∞-kontroll

Der P er modellen, og K er en regulator som optimaliserer outputen. u er da manipulerte input til modellen. Basert på blokkdiagrammet kan en se at H_∞ er en *state feedback* regulator der en har avhengigheten, $u = Kv$. Denne metoden for regulering vil ikke bli gått mer i dybden rundt det matematiske, siden det krever en del utredning og dypere forståelse. Denne forståelsen kan en få ved å undersøke Khalil sin litteratur om *Nonlinear Systems*, [16]. Det som er viktig å legge merke til er at denne metoden kan bli brukt til høyst ulineære systemer, og kan løses med tunge matematiske metoder. I tillegg er H_∞ også kompatibel innenfor robust regulering som beskrives i neste avsnitt, 2.5. Kienitz mener i sitt arbeid, [17], at H_∞ er godt egnet for parametrisk usikkerhet rettet mot diabetes.

2.5 Robust regulering

Robust regulering er fremgangsmåter som tar hånd om usikkerhet, enten i form av usikre parametere eller forstyrrelser, [27].
Når man tar i betraktning usikkerhet i en modell gir man rom for å takle små forskjeller mellom den nominelle og den faktiske modellen. En regulator som

fungerer godt i omgivelser med usikre parametere eller forstyrrelser vil antas å være robust. Dette gjør at små endringer i systemet, kanskje feilmåling eller forstyrrelser ikke vil gi utslag på det helhetlige systemet. Der parametere vil kunne variere med tiden eller fra pasient til pasient vil det være fornuftig å ta i bruk robust regulering. Systemet blir definert med endel begrensninger eller antagelser som en regulator må tilfredsstille. Dette kan medføre lavere helhetlig utførelse og nøyaktighet, men en får tilfredsstilt begrensningene, slik at systemet tåler mer.
For å etablere disse begrensingene eller antagelsene, blir det i litteraturen sett på ulike metoder for å oppnå best mulig resultat. En ønsker å definere et intervall hvor parameteren er gyldig. For ikke å risikere at en kalkulerer over for stort område, er det ønskelig med kortest mulig intervall.
Det er i hovedtrekk 3 ulike metoder for å ta i betraktning usikkerhet og definere et slikt intervall:

- Nominell deterministisk kontroll
- Sannsynlighets-modell
- Robust optimaliseringsmetode

Listen er hentet fra Pusche et al, [21], hvor de beskriver hvordan en skal definere usikre parameter.
Ved *robust optimaliseringsmetode* brukes algoritmer for eksempel min-max formulering, for å gjøre en "worst-case"-analyse til en parameter. Parameteren får da intervallet begrenset av "worst-case" verdien samt "best-case". Disse metodene er ofte krevende, og det er ikke alltid at metoden resulterer i de beste mulige intervallene.
Ved *sannsynlighets-modell* vil en ta den nominelle verdien til parameteren og kalkulere en prosentverdi, for så å lage en øvre og nedre grense på intervallet. Denne metoden vil være relevant dersom en robust optimaliseringsmetode blir for krevende i henhold til datakraft og tidsbegrensing i beregningene.
Det første punktet som nevner *nominell deterministisk kontroll* vil ikke være like relevant, da den ignorer usikkerheten i parameteren. Dette passer kun når systemet ikke er sensitivt til usikre parametere.

2.6 Ordinary Differential Equations, ODE

Når et system er skrevet som en ODE, er likningene i systemet ordnet som i Likning 2. Dette betyr at de algebraiske funksjonene i systemet, y, er uavhengig av tidsderiverte av modellen, $\frac{dy}{dt}$. Dersom disse skulle vært avhengig av hverandre, vil en ende opp med et system av Differential Algebraic Equations, DAE. Ved DAE er det betraktelig vanskeligere å oppnå en numerisk løsning. ODE

løses typisk med numeriske metoder som for eksempel Runge-Kutta eller Eulermetode, [20].

$$f(\frac{dy}{dt}, y, t) = 0 \tag{2}$$

2.6.1 Løse ODE numerisk

Runge-Kutta 4. ordens metode

Runge-Kutta er metode for å regne ut tilnærmete løsninger av differensiallikninger. De to mest anvendte metodene av Runge-Kutta er andre orden og fjerde orden. Ordenen defineres av hvor mange approksimasjoner som skal utføres. Nedenfor vises hvordan en Runge-Kutta 4. orden blir implementert.

$$k1 = f(t_n, y_n) \tag{3a}$$
$$k2 = f(t_n + h/2, y_n + (h/2) \cdot k1) \tag{3b}$$
$$k3 = f(t_n + h/2, y_n + (h/2) \cdot k2) \tag{3c}$$
$$k4 = f(t_n + h, y_n + h \cdot k3) \tag{3d}$$
$$y_n + (h/6) \cdot (k1 + 2 \cdot k2 + 2 \cdot k3 + k4) \tag{3e}$$

Konstantene, $(k1, k2, k3, k4)$, blir definert og brukes i Taylor-utvidelsen i siste likning, (3e). Taylor-utvidelsen regner ut og gir ut neste verdi og løsning av Runge-Kutta basert på forover integrering. $f(t_n, y_n)$ definerer den deriverte funksjonen med variabler y_n. t_n er tiden, mens h er skrittlengden til metoden.

Implisitt Runge-Kutta

Implisitt Runge-Kutta, IRK, er en avansert form for Runge-Kutta, og passer spesielt til systemer som er *stiff*, Seksjon 2.7. En slik metode approksimerer løsningen på differensiallikningene numerisk ved å legge til algebraiske likninger. En matematisk framstilling av Implisitt Runge-Kutta vil ikke bli utledet grundig her, men en kort beskrivelse av *Collocation method*, [13], vil bli gitt. Grunnen til dette er at *Collocation*-metoden er en metode som i praksis er en form for IRK. Det vil også være lettere å forklare teorien bak denne metoden. *Collocation* baserer seg på en approksimasjon gitt av et 3. ordens polynom.

$$x = C_0 + C_1 s + C_2 s^2 + C_3 s^3 \tag{4}$$

Der C_i er koeffisienter, mens s er *Collocation*-punkter. Disse *Collocation*-punktene ligger innenfor intervallet, $[0, 1]$. Punktene sammen med polynomet er ment til å gi kandidater til mulige løsninger av systemet, for så å velge ut riktig løsning ved de gitte punktene. Et polynom av grad 3 er det som virker å bli brukt mest i *Collocation*-metoden.

2.7 Stiff systems

Stiffness er et vanskelig begrep å definere teoretisk, men er mer et praktisk fenomen som oppstår i et system. Shampine prøver å forklare fenomenet i sin artikkel, [22]. Ved at et system inneholder et slikt fenomen, vil vanlige numeriske metoder for å løse systemet være utilstrekkelig. En må da ta i bruk metoder som egner seg godt for *stiffness*, som blant annet Implisitt Runge-Kutta, 2.6.1.
Det som kan kjennetegne et slikt system, er at komponenter innad i systemet endrer seg mye hurtigere enn andre. Et annet tegn på at systemet kan være *stiff*, er dersom komponenter eller tilstander varierer hurtig periodevis, og langsomt i andre perioder. Ofte kan dette undersøkes ved å se på hvor egenverdiene til systemet er plassert i forhold til hverandre.

2.8 Verktøy for simulering og optimalisering

Det finnes flere verktøy for simulering og optimalisering. Blant annet har man det universelle programmeringsverktøyet MATLAB, eller andre verktøy som er kompatibel med MATLAB. Simulink er et grafisk simuleringsverktøy basert på MATLAB, som er laget for å simulere på en oversiktlig måte. De to neste verktøyene, YALMIP og CasADi, er rettet mot optimalisering av mer kompliserte problem, disse vil bli beskrevet litt grundigere.

2.8.1 YALMIP

YALMIP er et optimaliseringsverktøy laget for å løse kompliserte og ulineære problem. YALMIP er en MATLAB Toolbox skrevet av J. Löfberg, [18], i 2004. Dette er et verktøy som er rettet spesifikt mot optimalisering og robust regulering. Derfor inneholder det også mange viktige funksjoner som kan gjøre det lettere å utføre kompliserte optimaliseringsproblem. Den består blant annet av funksjoner som "robustify" og "optimize", som hjelper til med å gjøre systemet robust. Det som kanskje kan være ulempen med YALMIP, er at hele systemet, ikke bare modellen, må reimplementeres for at det skal fungere i praksis.

2.8.2 CasADi

Det andre optimaliseringsverktøyet er CasADi. Dette brukes for å løse ODE numerisk, og der skal utføres kompliserte kalkulasjoner, men at det fortsatt kreves hurtig utregning. CasADi er et *open-source* verktøy utviklet av Andersson, [5]. Dette verktøyet gir en rask implementasjon av ulike metoder for numerisk optimal regulering. CasADi tilbyr kompatibilitet med MATLAB, som gjør at man kan laste ned og importere CasADi inn i MATLAB-script som man allerede bruker. Litt endring i MATLAB-kode er nødvendig for at det skal samsvare med syntaks

for en CasADi implementering. CasADi er basert på syntaks lånt fra *computer algebra systems* for å brukes til *algorithmic differentiation*, (AD). AD er fortsatt en av hovedfunksjonaliteten til CasADi, men verktøyet har utviklet seg til å kunne løse problemer innenfor mange ulike områder. MPC, ODE, DAE og robust optimization er eksempler på ulike områder. CasADi vil være et fornuftig valg av implementeringsverktøy. Det er dog bare et verktøy for å gi brukeren byggeklosser for å løse optimaliseringsproblem, og derfor ikke et verktøy som løser problemet fullstendig. Derfor kreves litt programmeringsinnsats for å løse problemet. Desverre er CasADi sin feilmeldingshåndtering ikke så intuitiv, så det krever at en setter seg godt inn i verktøyet for å kunne ta det i bruk. Basert på at medveileder Sebastien Gros har god kjennskap til dette verktøyet, vil det være enklere å ta i bruk medveileder når det kreves forklaring av syntaks. Derfor vil det være naturlig å bruke CasADi for implementering av optimaliseringsproblemet.
CasADi håndterer ikke ulineariteter direkte, slik som if-setninger, og det kreves derfor ekstra matematisk forståelse for å endre på disse.

Hvordan gjøre ulineariteter "myke"?

For å håndtere ulineariteter i modellen kan trigonometriske funksjoner brukes som erstatning. I teorien kan trigonometriske funksjoner si hvor hurtig en funksjon kan endres til en annen funksjon. Dette foregår innenfor et intervall på $[0, 1]$, hvor 1 er 100% representasjon av funksjonen. Desto nærmere 0, jo "mykere" er ulineariteten. En ønsker å gjøre dette for de delene av modellen som inneholder if-setninger, max-funksjoner og for risk-funksjonen. Oversikt over disse finnes i den metabolske modellen i Seksjon 2.2. Basert på matematisk forståelse kan en gå fra if-setningen under til likningen i (5b).

```
1 if condition >= 0;
2     func = S1;
3 else
4     func = S2;
5 end
6
```

$$switch = \frac{1}{2}tanh(\sigma c) + \frac{1}{2} \tag{5a}$$

$$func = switch \cdot S1 + (1 - switch) \cdot S2 \tag{5b}$$

På denne måten kan en oppnå en kontinuerlig glatt kurve basert på en if-setning. Jo høyere σ er, desto brattere blir kurven. Hvis den er høy nok vil den tilsvare et hopp eller en uendelig stigning slik som en if-setning gjør når en

forutsetning blir sann. I Likning (5b) blir det vist hvordan de to if-forutsetningene blir vektet mellom hverandre. Dersom if-setningen inneholder flere forutsetninger må metoden utvides ytterligere, men fortsatt basert på samme prinsipp.

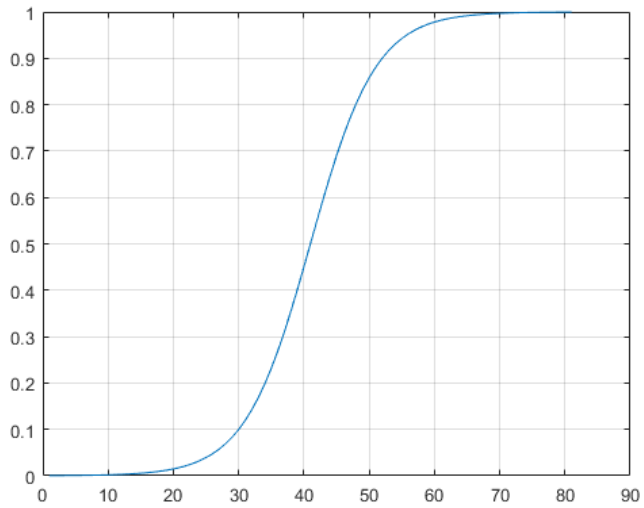


Figure 5: Kontinuerlig glatt kurve basert på switch-formelen

3 Parametere i modellen

I en modell er det ikke alltid like enkelt å definere parametere til en eksakt verdi. Staal beskriver glukose-insulin modellen i sin doktoravhandling, [24]. Han nevner at det er vanskelig å definere eksakte verdier når modellen inneholder ulinearitеter, og der enkelte parametere er personavhengig. Derfor kan det være interessant å se nærmere på parametere som en har kunnskap om, og hvor mye de varierer. En full oversikt over alle parametere er gitt i Appendix A. Basert på kunnskapen om parameterens variasjon ønskes det å definere et intervall som parameteren har muligheten til å bevege seg innen. Fra litteraturen er det ikke utført nok studie som tilsier hvilke parametere som eventuelt kan være usikre. Stort sett er det studier som tar i betraktning parametrisk usikkerhet i form av ytre forstyrrelser. Dette kan være alt fra fysisk aktivitet til matinntak. Disse forstyrrelsene vil ikke være like interessante når fokuset skal være på hvilke parametere i kroppen som kan variere fra person til person, og hvilket tidspunkt det er på døgnet.

3.1 Nominelle parameter

Det å finne nominelle parametere til en modell vil si å finne de ideelle eller teoretiske parametere som en kan ta utgangspunkt i. Videre kan en bruke disse nominelle verdiene til å definere ett sett med usikre parametere.
Det har vært vanskelig å finne god litteratur på parametere i glukose-insulin modellen, så det som vil bli brukt videre er en gjennomsnittspasient. Denne pasienten er generert av *subject*-filene fra UVa/Padova beskrevet i Seksjon 4.1.1. Dette vil gi en indikasjon på hvilke verdier disse parameterene vil ha. I Tabell 1 er parameterverdiene for*subject*-fil *adult#average.mat* generert. Som nevnt over kan disse også finnes med nærmere beskrivelse og enhet i Appendix A.

Parameter					
k1	0.0731	**kcounter**	0.0087	**Gb**	119.5897
k2	0.1076	**ki**	0.0109	**SRHd**	7.6559
Vg	1.8479	**kXGn**	0.0991	**kGSRd**	0.3999
m1	0.1849	**Gnb**	56.9834	**SQgluc_k1**	0.0101
m2	0.4840	**Fcns**	1	**SQgluc_kc1**	0.1192
m30	0.2774	**Vm0**	4.4739	**SQgluc_k2**	0.0257
m4	0.1940	**Vmx**	0.0715	**ke1**	0.0005
Vi	0.0511	**r1**	0.8109	**ke2**	339
kgri	0.0429	**r3**	1.4551	**epsilon**	0.5000
kabs	0.2062	**Km0**	226.6100	**gamma_param**	0.5000
f	0.9000	**p2u**	0.0470	**alpha_param**	0.0500
BW	69.7097	**Ib**	106.0437	**beta_param**	0.1100
kmin	0.0141	**ka1**	0.0038	**Sb**	0
kmax	0.0429	**ka2**	0.0177	**K**	2.3000
b	0.7611	**kd**	0.0161	**h**	119.5897
c	0.1372	**ksc**	0.1053	**kip**	0.1790
kp1	4.9865	**k01g**	0.1343	**kip1**	1
kp2	0.0054	**rho**	0.8000	**kip2**	0.06
kp3	0.0105	**kGSRs**	0.6462	**ksens**	0.5
kp4	0.0618	**Gth**	119.5897		

Table 1: Nominelle parameter

3.2 Usikre parameter

I denne oppgaven skal det bli definert ett sett med usikre parametere. Det som gjør en parameter usikker kan være basert på flere aspekter, og kan derfor være vanskelig å identifisere gjennom forsøk. En parameter kan for eksempel variere med tiden, men kan være definert som konstant i vår modellen. De usikre parameterne vil enten være basert på funn i litteraturen eller hvilke parametere som "antas" å være usikre. Antagelsene er gjort i samråd med veileder og hans kunnskap om de ulike parameterne. Nedenfor er det gitt en liste over antatte usikre parametere. Punkt nr. 2 er beskrevet grundigere under med tilhørende parametere.

- Insulinsensitivitet, V_{mx}
- Tidsforsinkelse/tidskonstanter
- Bukspyttkjertelens respons, β
- Fordelingsparameter mellom blod og portvenen, ϵ

- Forsinkelse mellom glukosesignal og insulinsekresjon, α

Insulinsensitivitet, V_{mx} er en av parameterne som antas å variere mest fra pasient til pasient. Hvilket tidspunkt på døgnet det er snakk om er også viktig når en ser på parameteren. V_{mx} er 25 % lavere ved kveldsmåltid sammenlignet med frokost og lunsj skriver Dalla Man et al. [8]. Disse forfatterne er også de som står bak store deler av parameterne som Frøyen [12], tar i bruk.
β, bukspyttkjertelens respons til glukose vil også antas å kunne variere fra pasient til pasient. Dalla Man nevner i samme litteratur at β vil være 25% lavere både ved lunsj og kveldsmåltid sammenlignet med frokost.
Ut over dette ble det ikke funnet litteratur på andre parametere, men gjennom undersøkelse av parameterne sin funksjon og etter samtale med veileder, ble det også fokusert på parameterne ϵ og α. Disse parameterne er satt til delvis tilfeldige verdier basert på *subject*-filene. Slik som ϵ, som sier hvor mye av insulinet som går ut i blodet når en påfører insulin intraperitonalt. Dette er en parameter som antas å være rundt 0.5, det vil si 50% antas å gå direkte i blodet. Det samme gjelder forsinkelsen mellom glukosesignal og insulinsekresjon, definert som α. I *subject*-filene er de tre sistnevnte parameterne satt lik fra pasient til pasient, men antageligvis vil ikke dette være tilfelle i praksis. Dette er parametere som mest sannsynlig vil variere.

Tidsforsinkelser eller tidskonstanter vil være med på å påvirke systemet, og noen av disse bør antas usikre. For eksempel k_{sens} som beskriver tidsforsinkelsen til glukose som går inn i sensordelen av den metabolske modellen i Seksjon 2.2. I Frøyen sin masteroppgave, [12], brukes en spesiell type sensor av Glucoset. Med denne sensoren setter de tidsforsinkelsen på 2 minutter, k_{sens}. Glukosesensorer blir som regel litt tregere over tid. Dette øker tidskonstanten som igjen øker forsinkelsen. Derfor vil dette være parametere som kan variere fra pasient til pasient.
Samtidig som glukosesensoren blir påvirket over tid, vil også insulinabsorpsjon variere i modellen. Dette gjelder blant annet k_{a1} og k_{a2} i *Subcutaneous insulin delivery* i metabolsk modell. En ny og fersk subkutan nål vil gi rask effekt i starten. Kroppen vil reagere på nålen som et fremmedlegeme, og vil etterhvert "pakke" nålen inn med vev. Tilslutt vil nålen ha høy forsinkelse i absorpsjon av insulin, og nålen må byttes ut.

3.2.1 Sikre parametere

Mange parametere kan bli sett på som usikre. Det kan være vanskelig å definere disse gjennom forsøk, men det finnes også parametere som kan sees på som sikre. Dette er typisk parametere som kroppsvekt, BW. Denne vil være målbar, og endres lite i løpet av et døgn. Dette gjelder andre parametere som det er

enkelt å måle. Ved slike sikre parametere bør en være oppmerksom på ikke å definere disse med parametrisk usikkerhet.

3.3 Usikker modellering

I usikker modellering ønsker man å ha en måte å definere de usikre parameterne i modellen. Modelleringen baserer seg på at man har ett sett med ideelle parametere som modellen blir kalkulert med. Parameterne blir i neste steg perturbert, og kjørt sammen med modellen igjen. I Figur 6 er det vist hvordan usikker modellering er beskrevet i henhold til *Robust Control Design* [14].

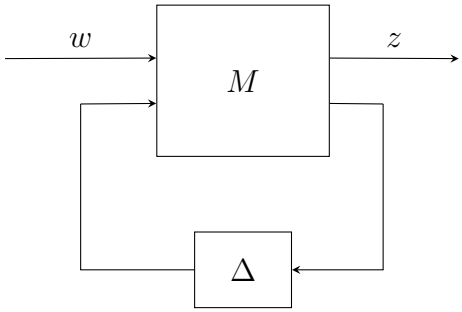


Figure 6: Usikker modellering

Der M er modellen og Δ er perturberingen. For å definere perturbasjonen finnes det flere ulike måter å gjøre dette på. Minimax-metoden, som er definert av blant annet Frediksson et al. [11], er en *robust optimaliseringsmetode* som minimaliserer kostfunksjonen ved et worst-case scenario, for så å sette en nedre og øvre grense. Et intervall som parameteren kan variere innen blir da opprettet. For å utføre et slikt optimaliseringsproblem trenger man å vite hvor mye en parameter kan variere. En *robust optimaliseringsmetode* følger beskrivelsen fra Seksjon 2.5.
I litteraturen er det få parametere i glukose-insulin modellen som er utforsket og definert. Dette gjør at det blir vanskeligere å definere et intervall som parameterne kan ligge innenfor. En måte å komme seg forbi denne problemstillingen er å ta i bruk *sannsynlighetsmodellen* fra Seksjon 2.5. I praksis betyr dette at det blir sett på *subject*-filene og verdien til alle parameterne definert av UVa/Padova. Alle pasientene blir sammenlignet, og regner deretter ut maks- og min-verdi for hver parameter. Da får man en indikasjon på hvor store intervall en kan jobbe med. Dette resultatet vil nok ikke være 100% korrekt, men en tilnærming. Oversikt over intervallene er gitt i Seksjon 7.1.1.

3.4 Sensitivitetsanalyse

Sensitivitetsanalyse baserer seg på å bedømme hvilke parametere i en modell som er mulig å estimere basert på et gitt input scenario og målinger. Dette er viktig for å se om parametere er identifiserbar. Dersom en kan si at en parameter er identifiserbar kan en også bruke dette ytterligere til å utføre flere ulike forsøk med denne parameteren. De parameterne som viser til store utslag på sensitivitetsanalysen, vil være de mest interessante parameterne å følge opp når en skal se på parametrisk usikkerhet. For ethvert dynamisk system med et sett med input og et sett med målinger, er det mulig å utføre en sensitivitetsanalyse. En slik analyse er nyttig å utføre i ulineære modeller. Som et resultat av for eksempel if-setninger, vil slike ulinære modeller ofte ha deler som er inaktiv i deler av tilstandsrommet. Derfor kan det være interessant å utføre en analyse der en sørger for å få aktivisert flest inaktive deler. Det kan være lurt å kjøre analysen flere ganger med ulike input. Med analyse fra flere ulike input-scenario, vil en kunne oppnå flere parametere som bidrar til sensitivitet.
Måten en sensitivitetsanalyse blir utført på er ved å simulere med initielle tilstander og bruke nominelle parameter. Videre justere på en parameter og simuler på nytt. Ved å justere en parameter kan dette medføre endring i output. Dersom det ikke er noen endring i output, altså differansen er lik null, vil ikke data bidra til informasjon. Slik sensitivitet på null vil forekomme dersom modellen sliter med ikke-identifiserbarhet for den perturberte parameteren.
Flere ulike måter finnes for å utføre sensitivitetsanalyse. Odd-Martin Staal legger fram i sin doktoravhandling, [24], en komplett teoretisk framgangsmåte for en slik analyse. Basert på *Singular Value Decomposition, SVD* kan en regne ut en sensitivitetsmatrise, S_y. Denne inneholder alle parameterne systemet har i hver kolonne, og alle tidsmålinger i hver rad.
Sensitivitetsanalyse kan også bli utført via software tools. CasADi tilbyr en algoritme som regner ut sensitivitetsanalyse for ODE- og DAE-systemer. Algoritmen kommer i CasADi sin eksempelmappe, *casadi_package_example.zip*, som kan lastes ned via CasADi sine hjemmesider, [5]. Flere ulike måter for sensitivitetsanalyse kan bli tatt i bruk, men enkleste og mest håndterbare metode baserer seg på *Finite differences approksimasjon*.

3.4.1 Finite differences approksimasjon

I noen tilfeller blir det for komplisert å regne ut den derivative av en funksjon, og andre metoder må bli tatt i bruk. Finite differences er blant de enkleste metodene som regner ut en numerisk approksimasjon til den deriverte. Det er den deriverte som er interessant for sensitivitetsmatrisen, S_y i sensitivitetsanalysen. Dette er en matematisk fremstilling som vist under:

$$f'(x^*) = \frac{f(x^* + dx) - f(x^*)}{dx} \tag{6}$$

Hvor x^* er punktet som en ønsker å finne f sin approksimerte deriverte rundt. $x^* + dx$ blir neste punkt i minimal nærhet til x^*, med dx som skrittlengde.

4 Modell for simulator

Denne delen beskriver hvordan det fullstendige systemet er designet, hvilke "use-case" som finnes, og hvordan en skal ta i bruk de ulike funksjoner og input til simulatoren. Modellen og simulatoren er bygd opp på samme måte som Erlbeck, [9], gjør i sin oppgave. Erlbeck baserte seg på UVa/Padova sin simulator. Funksjoner som har minimal endring fra tidligere versjoner vil ikke bli beskrevet i detalj. Funksjonalitet som er lagt til vil bli nevnt, men beskrevet grundigere i kapittelet om CasADi APT-simulatoren, Kapittel 6.

4.1 Design av simulator

Simulatoren er basert på input fra et datasett, enten det er *.mat*-filer i MATLAB eller ASCII-filer. Basert på disse filene klarer simulatoren å initialisere alle parametere og konstanter, og fastslå simuleringstid og eventuelle måltid. I tillegg er det mulighet for å forhånds bestemme hvilke "use-cases" som skal kjøres. De ulike delene som er med på å initialisere input til simulatoren er beskrevet under.

4.1.1 Input til simulator

Subject-filer

Cobelli et al. [7] har inkludert et parametersett til UVa/Padova-simulatoren som skal generere "virtuelle pasienter". Disse er importert via *.mat*-fil i MATLAB, og kalles *subject*-filer og inneholder 30 pasienter. 10 voksne, 10 ungdom og 10 barn er inkludert, og definert henholdsvis som *adult#001-adult#010*, *adolescent#001-adolescent#010* og *child#001-child#010*. Hver pasient inneholder 45 parametere, som vil være tilstrekkelig for å regne ut resterende parametere samt alle likninger i modellen. *adult#001.mat* blir brukt som standard for simuleringer, men andre pasienter er også mulig å ta i bruk. I Tabell 2 er det vist de 45 parameterne fra *subject*-filene.

***Subject*-fil og dens parametere**			
Age	$alfaG$	b	BW
CF	CL	CR	d
$EGPb$	Gb	Gnb	Ib
$k01g$	$k1$	$k2$	$ka1$
$ka2$	$kabs$	$kcounter$	kd
$kGSRd$	$kGSRs$	ki	$Km0$
$kmax$	$kmin$	$kp2$	$kp3$
ksc	$kXGn$	$m1$	$m5$
$name$	$p2u$	$r1$	$r3$
$SQgluc\_k1$	$SQgluc\_k2$	$SQgluc\_kc1$	$SQgluc\_Vgcn$
$T1DM\_duration$	TDI	Vg	Vi
Vmx			

Table 2: *Subject*-fil generert av UVa/Padova

Scenario-filer

Scenario-filene er også definert via UVa/Padova sitt forskningsmiljø. Dette er ASCII-filer skrevet på *.scn*-format. Informasjonen definert i disse filene er med på å spesifisere simuleringsverdiene til simulatoren. Hvordan *scenario*-filen er implementert er visualisert i Figur 7.

```
1   Test file scenario
2   CL_24hr_50g
3
4   Simulation
5   %Tsimul=24
6   %QTsimul=hour
7
8   %simToD=0
9   closed loop
10
11  %Tclosed=48
12  %QTclosed=hour
13
14  %OB=on
15
16  Meals
17  %Tmeals=[0]
18  %QTmeals=hour
19  %Ameals=[0]
20  %Qmeals=total
21
22  %Qbasal=quest
23
```

Figure 7: *Scenario*-fil

Dette er bare et utdrag av en *scenario*-fil for å ta med de viktigste elementene i en slik fil. Tilleggsinformasjon om eventuell feilhåndtering kan også bli lagt inn her.
For å beskrive denne filen ytterligere, kan en se at dette er et 24-timers scenario siden *Tsimul=24* og *QTsimul=hour*. Videre er *Tmeals=0*, *Ameals=0* og *Qmeals=total*. Dette setter henholdsvis tid for når måltid inntas, amplituden/størrelsen på måltidet og at måltidet er definert som *gram*. Grunnen til at *Tmeals* og *Ameals* er satt til disse verdiene er siden disse blir definert i MATLAB i *load_scenario.m*. Denne delen er beskrevet senere.
'OL_24hr_50g_Error.scn' er *scenario*-filen som blir brukt som standard i simulatoren.

Feilestimat

Studenten ved NTNU, Karl Arthur Unstad, [25], utførte i sin masteroppgave en måte å håndtere feilmargin fra utstyret. En såkalt *pump_error*. Dette fører til at basalt og bolus tilførsel av insulin basert på Erlbeck's og Unstad's simulator er forskjellig. Et slikt feilestimat må brukes videre i oppgaven.

load

load-delen er et MATLAB-script, *load_scenario.m*, skrevet av Erlbeck, [9]. Denne gjør at det blir lettere å definere de ulike initialverdiene for simulatoren, samt omforme det som står i ASCII-verdier til MATLAB-syntaks. En kan da ta i bruk *scenario*-filen *'OL_24hr_50g_Error.scn'*. Det er beskrevet i avsnittet om *scenario*-filer at variablene blir satt til 0. I denne delen av simulatoren blir tidspunkt for når et måltid skal forekomme endret til: *Tmeals* = $[2, 8, 15]$ timer. Mens *Ameals* blir satt til 50, 80 og 80 gram. Hvis ingen måltid er definert ved start er $D = 30$ gram. I tillegg vil *Dmeals* definere lengden på måltidet med "default"-verdi på 15 min.
Nå er alle de viktige parameterne satt for å initialisering, og en kan kjøre simulatoren.

4.1.2 Kjør simulator

Det totale systemet er bygd opp slik at det skal være mulig å sammenligne resultater fra simulator i CasADi med Unstad, [25], sin versjon av simulatoren. Derfor er det definert ulike "use-case" som en kan forhånds bestemme. Dette gjør det mulig å kjøre alle former for simulatoren fra main-script i MATLAB.
Systemet er delt inn i "use-case" **A:** APT-simulator, **B:** CasADi APT-simulator. **B** er igjen delt inn i tre deler, **B.1:** CasADi APT-simulator basert på Runge-Kutta 4.ordens metode, Seksjon 2.6.1, **B.2:** CasADi simulator og optimalisering basert på Implisitt Runge-Kutta-metode, Seksjon 2.6.1. Tilslutt er sensitivitetsanalysen implementert via "use-case" **B.3**.
Ett overordnet flytdiagram over det totale systemet er vist i Figur 8. Dette skal være til hjelp for å visualisere det totale systemet med de ulike "use-case". De ulike metodene er ikke visualisert i detalj her.

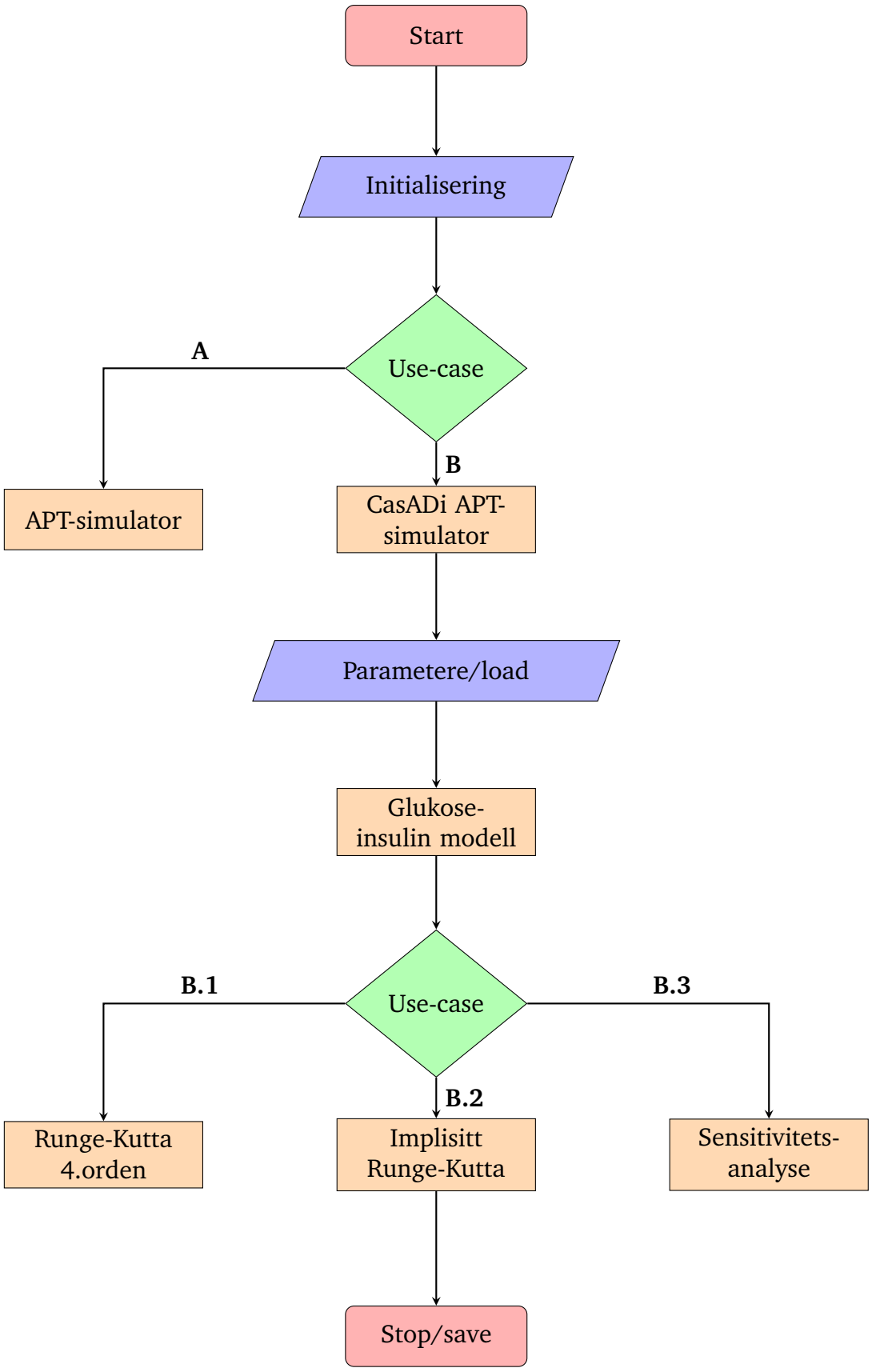


Figure 8: Flytdiagram av totalt system

5 Utførelse av Sensitivitetsanalyse

Dette kapittelet beskriver hvordan en sensitivitetsanalyse er utført og implementert for glukose-insulin modellen.

Sensitivitetsanalyse ved start

For å kunne utføre en sensitivitetsanalyse på vår modell må simulatoren stemme overens med forventningene. Dette ble testet ved å sammenligne med den opprinnelige modellen mot en perturbert modell. Da skal det resultere i en differanse mellom disse. I Figur 9 blir simulatoren først kjørt uten perturbasjon (rød), og så med en perturbasjon for insulinsensitivitet, *Vmx*. Dette er også en av parameterne som antas å være usikker. I original versjon er *Vmx*$= 0.0715$, mens i perturbert versjon er *Vmx*$= 0.3$. Her er det sett på tilstanden *GscSens* som er målinger av glukose subkutant.

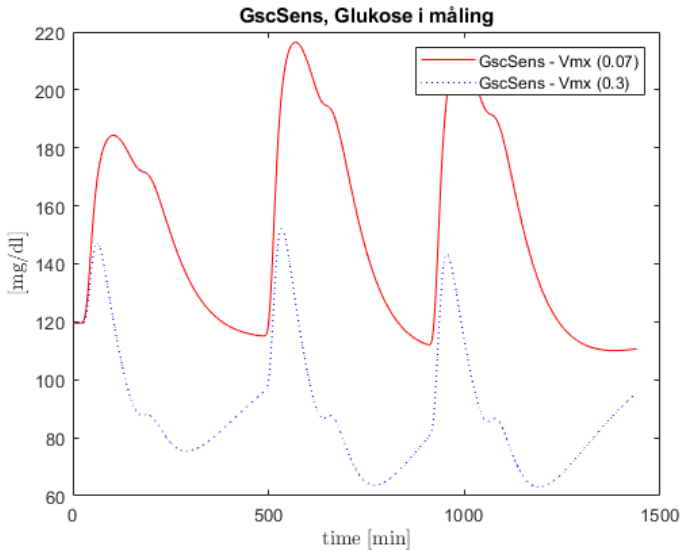


Figure 9: *GscSens* simulert med *Vmx*$= 0.07$ (rød) og *Vmx*$= 0.3$ (blå)

Vmx$= 0.3$ er en veldig høy verdi basert på sitt område definert av maks- og min-grenser, og vil derfor ikke brukes videre.

5.1 Implementasjon av sensitivitetsanalyse

Sensitivitetsanalysen ble implementert ved hjelp av CasADi sin eksempelkode. Koden bruker en parameter som input i systemet, derfor må også denne parameteren være definert med CasADi syntaks. Hvordan denne syntaksen er implementert er beskrevet i Kapittel 6. Tidligere har alle parameterne vært definert

med en numerisk verdi. Dette gjør at en må gjøre noen justeringer når en skal regne ut hele systemet. Sensitivitetsanalysen gir ut sensitiviteten i det bestemte tidssteget basert på *Finite differences*-metoden. Sensitiviteten må derfor regnes ut for hvert tidssteg, samt for hver parameter. En matrise med dimensjoner $1440x59$ viser full oversikt over sensitiviteten. Hver parameter ble justert like mye med hensyn til sitt maksimum- og minimum-område. Måten perturberingen er utført på er vist i Likning 7.

$$Parameter \cdot [1 + h \cdot (maxValueParameter - minValueParameter)] \quad (7)$$

Hvor h er prosentandel en ønsker å perturbere parameteren med. Her er det brukt 1% av parameteren sitt området. Sensitivitetsanalysen er utført basert på *subject*-fil, *adult#average.mat*. Dette er for å ha en gjennomsnittlig pasient, som potensielt kan gi tilnærmede nominelle parameterverdier.
Tilstanden som er av interesse ved sensitivitetsanalyse av glukose-insulin modellen, er $G_{SC,sens}$ eller $G_{IP,sens}$. Disse tilstandene gir ut informasjon om målinger av glukose for henholdsvis subkutant eller intraperitonealt målinger.

For å regne ut området til en parameter blir metoden beskrevet i Seksjon 3.3 brukt, og resultatene er vist i Seksjon 7.1.1. *subject*-filene inkluderer også noen parametere som ikke varierer fra pasient til pasient. For eksempel ϵ som er satt til en fast verdi 0.5. Det er trolig utført en tilfeldig gjetning på denne verdien for denne parameteren, og tilsvarende for andre parametere. Siden lengden på intervallet blir 0 for denne parameteren, så er det utført en gjetning på øvre og nedre grense. Da vil det også være mulig å regne ut sensitiviteten for disse parameterne.

5.2 Singular Value Decomposition

Etter *Finite Difference*-metoden har blitt gjennomført blir videre analyse utført basert på *Singular Value Decomposition*, *SVD*, fra Seksjon 3.4. Dette er basert på MATLAB sin innebygde funksjon. $[U, S, V] = svd(sensitivityMatrix)$. S er da singulærverdiene, *SV*, som er visualisert i Seksjon 7.1, og kan brukes til definere identifiserbarheten av parameteren. V er *Right Singular Value*, *RSV*. SV og RSV er korresponderende og kan bli brukt til å se på mellomrommet i singulærverdiene. Dette er brukt for å si noe om parameteren sin påvirkning på systemet.

6 CasADi APT-simulator

I dette kapittelet blir APT-simulatoren som er implementert i MATLAB overført til CasADi. Det viser hvilke funksjoner som har blitt tatt i bruk, samt hvilke som er nye. Denne delen viser også hvordan Runge-Kutta 4. ordens metode og Implisitt Runge-Kutta er implementert. For å kunne gjøre egne vurderinger og eventuelt egne forbedringer, blir tidligere implementeringer undersøkt. Resultater som er viktig for videre beslutninger er framstilt fortløpende.

6.1 Tester av tidligere simulatorer

For å bli kjent med forskjellige simulatorer ble først Frøyen, [12], sin simulator testet. Denne er i hovedsak implementert i MATLAB Simulink. Frøyen implementerte også en enkel MPC regulator som ble kjørt uten problemer.
Erlbeck, [9], har implementert sin simulator i MATLAB og følger modellikningene til UVa/Padova-simulatoren. Dette resulterte i mer nøyaktig simulering av prosessen. I denne oppgaven skal parametrisk usikkerhet bli utforsket, dermed ble simulatoren til Erlbeck utprøvd ved å justere ulike parametere. Justering av utvalgte parametere ga ingen endring i resultat, dermed ble det konkludert med at det måtte være en feil eller noe som manglet i Erlbeck sin APT-simulator. Ved å se på Unstad, [25], sin oppdaterte versjon fra 2017, ser man at feilen er korrigert, og det er mulig å få ut ønskelig resultat ved justering av parametere. I tillegg legger Unstad til ulike feilestimat, slik at sensor- og pumpefeil kan bli tatt i betraktning. Dette skal bidra til en sikkerhetsmekanisme i et AP-system. Basert på feilen som ble korrigert, og feilestimatet som blir lagt til, blir det valgt å fortsette med denne versjonen av simulatoren.

6.2 Innføring i CasADi

Nyeste versjon av CasADi ble lastet ned, CasADi v3.5.1. Denne ble importert inn i MATLAB-biblioteket, slik at CasADi kan bli brukt direkte opp mot MATLAB. Kunnskap om syntaksen for CasADi måtte tilegnes. En *.zip*-fil med eksempelkoder, *casadi_package_example.zip*, ble lastet ned fra CasADi sine hjemmesider, [5]. Disse ble brukt for å få en innføring i hvordan implementering i CasADi fungerer. En metode som ble testet ut var *Direct Multiple Shooting*. Dette er en metode som løser ODE numerisk basert på Runge-Kutta, som er beskrevet i Seksjon 2.6.1. Andre metoder ble også testet ut, men i hovedsak er det denne algoritmen som er mest relevant for dette prosjektet.
Sammen med eksempelkoden var også en brukerguide for å gi en detaljert og grunnleggende forståelse av syntaks i CasADi. Eksempelkodene vil bli brukt som mal for hvordan man kan sette opp og løse en ODE, men også for videre

optimalisering.

6.3 Implementasjon av APT-simulator i CasADi

For å få en oversikt over hva de ulike funksjonene gjør og hva som er nytt, så er alle funksjoner som er blitt tatt i bruk beskrevet under.

main.m

Initialiserer hvilke *subject*-fil, *scenario*-fil og hvilke "use-case" som skal bli tatt i bruk. Samt initialisering av feilestimering av utstyr implementert av Unstad [25]. *main.m* setter igang systemet og simulatoren.

---

run_apt_simulator.m

Dette scriptet brukes kun til sammenligning, og er uendret fra da Unstad, [25], oppdaterte APT-simulatoren. Denne delen setter opp parametere, kjører *load*-scriptet med ønskede karakteristikk. Forbereder til å kjøre simulatoren med initialverdier for hver tilstand i modellen.

---

casadi_run_apt_simulator.m

Har samme funksjonalitet som *run_apt_simulator.m*, men basert på CasADi-syntaks.

---

casadi_glucose_insulin_model.m

Her blir simulatoren implementert med alle modellikninger slik som i den metabolske modellen vist i Seksjon 2.2. Viktig at likningene følger CasADi-syntaks, der alle if-setninger, max-funksjoner og risk-funksjonen er implementert matematisk.

---

casadi_RungeKutta4.m

Runge-Kutta 4.ordens, RK4, metode blir implementert i dette scriptet for å approksimere en løsning av ODE. En diskret numerisk løsning av ODE blir så sendt inn som nye verdier på tilstandene, og RK4 blir kjørt på ny. Dette blir kjørt for hvert tidssteg over hele simuleringtiden av en for-løkke.

---

casadi_collocation.m

Her blir det definert og implementert en annen versjon av simulatoren basert på Implisitt Runge-Kutta ved hjelp av *Collocation*-metoden fra eksempelkoden. I Seksjon 2.6.1 er det nevnt at dette bidrar til simulering og optimalisering i ett.

6.3.1 Forbedring av implementasjon

For å kunne si hvorfor en ønsker å reimplementere en allerede fungerende simulator, må en gå litt i dybden på de ulike forbedringer som er blitt utført. En mer detaljert framstilling av de viktigste delene som er endret fra tidligere versjoner vil bli lagt frem.

Modell i CasADi

Den nye simulatoren er basert på Unstad, [25], sin nyeste versjon av APT-simulatoren. Denne inkluderer alle modellikningene som tidligere. I tillegg er en feil fra Erlbeck sin versjon rettet opp.
Ved å bruke CasADi kan modellikningene bli implementert som en matematisk framstilling av problemet, uten å legge til ekstra innebygde funksjoner. I likning (8a) vises hvordan likningene er framstilt matematisk på matriseform, og (8c) viser hvordan hver likning er implementert i CasADi.

$$\dot{x} = Ax + Bu \tag{8a}$$

$$y = Cx \tag{8b}$$

$$x1\_dot = x1 + u1 \tag{8c}$$

Hvor x er en vektor med dimensjon $25x1$, det vil si x inneholder 25 tilstander. u er systemets pådrag med dimensjon $5x1$. Både x og u er representert i henholdsvis Likning 9 og 10. A inneholder alle parameterne i modellen, samt de algebraiske likningene som ellers kreves for å sette opp tilstandene for en ODE. Dette blir en matrise med dimensjon $25x25$. Mens B gjenspeiler hvilket pådrag som hører til hvilken tilstand, og har da en dimensjon $5x25$.

$$x = \begin{bmatrix} G_p & G_t & I_p & I_l & Q_{sto1} & Q_{sto2} & Q_{gut} & X^L... \\ X^H & I_1 & H & X & G_{SC} & G_{SC,sens} & I_{ip1} & I_{ip2}... \\ G_{IP} & G_{IP,sens} & SR_H^s & H_{sc1} & H_{sc2} & I_{po} & Y & I_{sc1}... \\ I_{sc2} & & & & & & & \end{bmatrix} \tag{9}$$

$$u = \begin{bmatrix} D \cdot \delta \\ D \\ u_{iv} \\ u_{ip} \\ IIR \end{bmatrix} \tag{10}$$

En ser her at modellen bare inneholder 25 tilstander sammenlignet med 45 fra tidligere simulator. Det betyr at de resterende 20 tilstandene kan bli sett på som algebraiske funksjoner. Dette gjør det mulig å løse problemet som en ODE, dersom en regner ut de algebraiske funksjonene først og kronologisk. CasADi deklarer de algebraiske funksjonene slik at de kan bli brukt i differensialliknin-gene. Dermed unngår man komplikasjoner og lang kjøretid ved å bruke prinsip-per rettet mot DAE. DAE er kort beskrevet i Seksjon 2, og vil ikke bli beskrevet ytterligere. Dette er metoden som tidligere APT-simulator er basert på. Kom-pleksiteten og kjøretiden økes med en slik implementasjon.
Alle tilstandene og input blir deklarert ved hjelp av CasADi-syntaks som vist under:

$$x = SX.sym('x')$$

For å gjøre litt opprydning i tidligere kode blir elementer som baserer seg på UVa/Padova-simulatoren fjernet. Tilgangen til denne simulatoren er forøvrig begrenset, kostbar og ikke lenger nødvendig. I første omgang er det bare Dia-betes Type 1 som er eneste sykdom av interesse, så all kode rettet mot Type 2 blir fjernet.

Som nevnt i Seksjon 2.8.2, så takler ikke optimaliseringsverktøyet CasADi if-setninger, max-funksjoner eller risk-funksjonen. Disse ble reformulert basert på matematisk formulering. For å sjekke at disse ble implementert korrekt, ble de testet mot ulike verdier for σ. Figur 10 og 11 viser henholdsvis switch-formelen og hvordan $SRHd$ oppfører seg basert på denne formelen. $SRHd$ er dynamisk utskilling av glukagon som tilhører seksjon *Pancreatic glucagon secretion* fra den metabolske modellen i Seksjon 2.2. Rød graf viser når $\sigma = 1$, og blå når $\sigma = 100$. Her kan en se hvordan switch-formelen påvirker beregningen av $SRHd$. Jo høyere σ er, desto mer korrekt er beregningene. Så det gjelder å finne en σ som representerer $SRHd$ godt nok, uten at optimaliseringsverktøyet klager på for "strenge" ulineariteter.

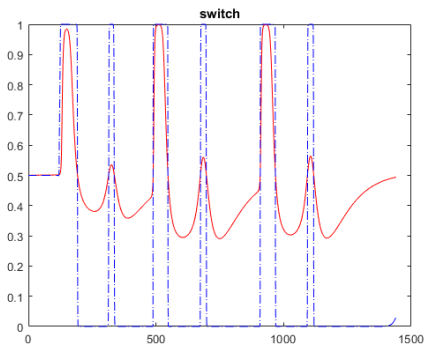


Figure 10: Switch-formel

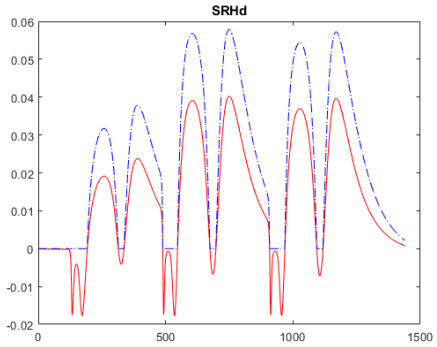


Figure 11: $SRHd$ basert på switch-formelen

Når alle likningene er korrekt implementert og samlet til en ODE kan den numeriske approksimasjonen, Runge-Kutta 4. orden, bli kjørt.

6.3.2 Runge-Kutta 4.ordens metode

Runge-Kutta 4. ordens metoden, RK4, ble implementert i henhold til eksempelkoden og teoriseksjon 2.6.1. ODE-systemet ble definert og løst basert på RK4. En Taylor-approksimasjon med steglengde, $DT = 0.25$. Dette utgjør en tidshorisont på $15sek$. Altså for hvert $15.$ sekund blir det utført en ny approksimasjon. Videre blir det regnet ut en diskret løsning av approksimasjonen for hver tilstand. Denne verdien blir brukt som input til neste tidsskritt i simulatoren. En har da implementert en simulator som er tilsvarende som Unstad, [25].

Sammenlign APT-simulator i MATLAB og RK4

Hvis en tar videre vurderinger om denne metoden og modellen er implementert korrekt må den sammenlignes med tidligere versjon av APT-simulator. Det første å legge merke til er at Unstad bruker i sin versjon en numerisk innebygd funksjon for å løse ODE. Denne kalles *ode15s*, og har en tidshorisont på 1 minutt. I den nye simulatoren basert på CasADi må det brukes en tidshorisont på 15 sekund for å gi samme resultat. Nedenfor viser hvordan de to ulike simulatorene framstiller glukosekonsentrasjonen i plasma, G_p. Steglengden er satt til 15 sekund i Figur 12, og 1 minutt i Figur 13.

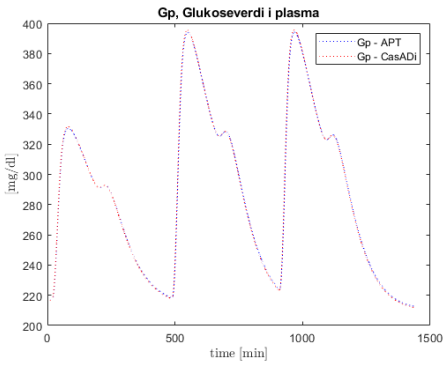


Figure 12: Glukosekonsentrasjon i plasma med steglengde 15 sek.

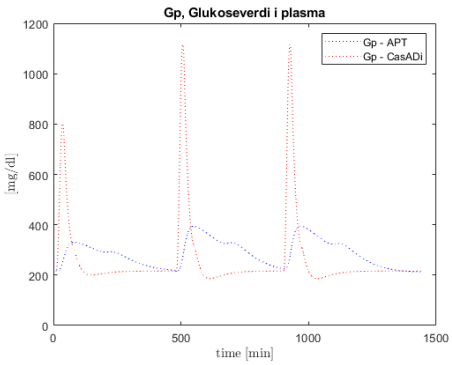


Figure 13: Glukosekonsentrasjon i plasma med steglengde 1 min.

Som vist i Figur 12 så ligger den rød stiplete linjen (CasADi-simulator) langs den blå stiplete linjen (APT-simulator), dette viser en god approksimasjon av modellen, og korrekt simulering. Slik er det ikke ved Figur 13. Da sammenfaller

ikke de to linjene. Selv om begge simulatorene har samme tidshorisont på 1 minutt, så klarer ikke modellen å approksimere riktig. Ved igjen å se på den innebygde funksjonen som er tatt i bruk, *ode15s*, og at Unstad representerer sitt system som en DAE. Basert på *s*-en i funksjonen så defineres systemet som et *stiff* system. Dette tyder på at systemet kanskje har karakteristiske egenskaper som et *stiff* system. Med slike egenskaper kan ikke enkle numeriske metoder som Runge-Kutta 4. ordens metode regne med like stor tidshorisont. Den må derfor regne ut hyppigere, og hvert *15.* sekund ser ut til å være tilstrekkelig.
En kan da se til andre numeriske metoder for å løse problemet. Dette er løst med å ta i bruk Implisitt Runge-Kutta som er implementert i neste seksjon.

6.3.3 Implisitt Runge-Kutta med *Collocation*-metoden

Som nevnt i teoriseksjon 2.6.1 er Implisitt Runge-Kutta, *IRK*, godt egnet for systemer med *stiff* egenskaper. I denne delen av oppgaven blir IRK implementert basert på eksempelkode om *Direct Collocation method*. Denne metoden tar for seg både simulering og optimalisering i ett. Derfor kan Runge-Kutta 4. ordens metode forkastes, og kun brukes til videre sammenligning.
Selve implementeringen av metoden er i praksis "klipp og lim" fra eksempelkoden, men for at systemet skal simulere og optimalisere som forventet, kreves litt forståelse av metoden.

Når polynomet er definert slik som i Seksjon 2.6.1, må en initialisere øvre og nedre grense på alle tilstandene, og samtidig utføre en gjetning på tilstandene innenfor intervallet. Jo bedre gjetting en gir, desto bedre optimalisering oppnås. Alle gjetningene blir lagret i en spesiell rekkefølge i en vektor, $w0$. Denne vektoren, $w0$, representerer vektoren w i $J(w)$ nevnt under Seksjon 2.4.2. I første omgang vil en prøve å sammenligne simulatoren basert på RK4, derfor bør alle input og initielle verdier være tilsvarende. Systemet kjører gjennom alle *collocation*-punktene og alle kontrollintervallene, N. For hvert kontrollintervall defineres hvor mange deler tidshorisonten skal deles inn i. Framgangsmåten som er beskrevet i dette avsnittet samsvarer med teorien og oppsettet bak en MPC-regulator fra Seksjon 2.4.2.
En ønsker at diskretisering av IRK og RK4 er lik, derfor må det være en tidshorisont på 15 sekund. Dette får en ved å basere seg på N (kontrollintervall) og T (lengde på simulering/optimalisering), der N er satt til 120, og T satt til 30. Dette gjør at det blir samme diskretisering $T/N = 0.25$ min.
Tabell 3 skal være til hjelp for å skaffe en oversikt over hvilke grenser eller hvilke verdier som blir satt i *Collocation*-metoden som er satt opp som et Nonlinear Programming-problem, NLP.

Variabel	Grense	Verdi
x	$\pm\infty$	$x[RK4]$
u	$u[RK4]$	$u[RK4]$
N	——	120
T	——	30

Table 3: Oversikt over verdier i NLP-problemet

Alle grenser og gjetninger blir sendt inn for å løse NLP-problemet. CasADi har en egen funksjon for å løse NLP-problemer.

```
1 nlpsol('solver', 'ipopt', model);
2
```

Hvor *'ipopt'* er et software-bibliotek, [26], for å løse NLP-problem i stor skala. Dette verktøyet er designet for å finne lokale løsninger av optimaliseringsproblem basert på en kostfunksjon. Før det blir lagt frem sammenligninger med den tidligere RK4-metoden og simuleringsdelen av IRK, blir valg av kostfunksjon lagt frem.

6.3.4 Valg av kostfunksjon

Det kreves at simulering av *Collocation*-metoden er lik som simulering av Runge-Kutta 4. ordens for å kunne gå videre med å optimalisere systemet. For å bekrefte at metoden er riktig må kostfunksjonen en velger være enkel å løse og følge pådraget til systemet. Det vil si at det er simulatoren blir som i første omgang blir optimalisert. Simulatoren blir styrt av pådrag ved injisert insulin subkutant. Dette tilsvarer inputen IIR. Resten av variablene fra inputvektoren i Seksjon 6.3.1 vil enten være 0 eller bli sett på som forstyrrelser. For eksempel vil måltid bare være en ytre forstyrrelse som en ikke direkte kan kontrollere.
Valg av kostfunksjon baserer seg dermed på IIR og referansen, IIR_{ref}, og blir regnet ut ved å minimalisere avviket fra referansen.

$$J = \frac{1}{2}(IIR - IIR_{ref})^2 \tag{11}$$

IIR er definert som en profil som varierer med tiden. For at kostfunksjonen skal bli løsbar og oppnå tilfredsstillende resultat blir også IIR_{ref} satt til samme profil. Denne referansen er vist i Figur 14, der det forekommer en injeksjon etter 9 min. Den er stykkevis konstant på grunn av diskretiseringen for hvert $15.$ sekund, og er derfor visualisert ved hjelp av trappetrinn.

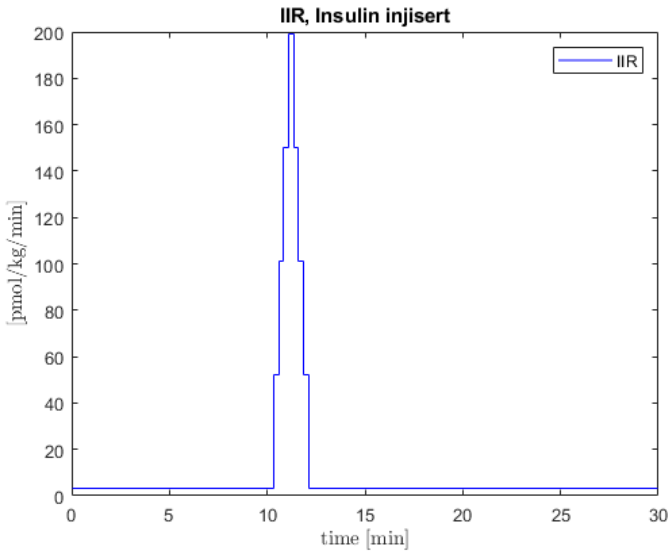


Figure 14: Input til systemet basert på enkel kostfunksjon

Senere vil det bli diskutert hvordan en kan gjøre kostfunksjonen mer komplisert for å optimalisere en regulator som MPC i lukket sløyfe.

6.3.5 Sammenlign IRK med RK4

Dersom kostfunksjonen er implementert korrekt, og tilstandene og input har grensene og verdiene som i Tabell 3, skal resultatet bli som vist i Figur 15.

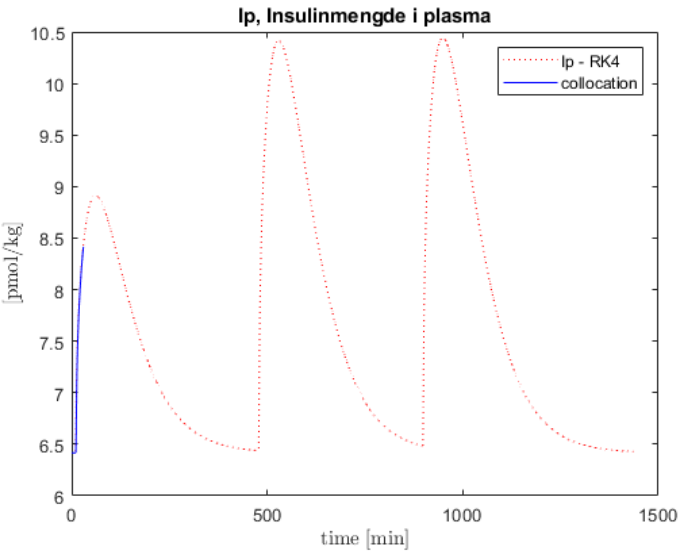


Figure 15: *Collocation*-metoden simulert over *30 min* for Ip

Den blå linjen sammenfaller med den rød stiplete linjen, og derfor følger IRK den opprinnelige simulatoren basert på RK4. Som igjen er sammenlignbar med APT-simulatoren i MATLAB.
Som nevnt tidligere må en mer avansert kostfunksjon til for å optimalisere en regulator i lukket sløyfe. Det ble ikke tid til å implementere dette, men en beskrivelse av videre arbeid vil bli gitt.

7 Andre resultater & observasjoner

Resultater som er viktig for videre avgjørelser har blitt lagt fram underveis i denne rapporten. Andre resultater og observasjoner vil bli nevnt her.

7.1 Sensitivitetsanalyse

En sensitivitetsanalyse har i denne oppgaven blitt utført. Her er utdrag av resultater fra det som ble gjennomført i Seksjon 5. Sensitivitetsmatrisen, S_y, er definert slik at hver kolonne representerer hvert tidssteg, mens hver rad representerer en parameter. Analysen er gjort for tilstanden *GscSens*, som er målingen av glukoseverdier i sensoren. I Figur 16 vises hvordan sensitiviteten er for målingen basert på perturbasjon av parameteren *Vmx*.

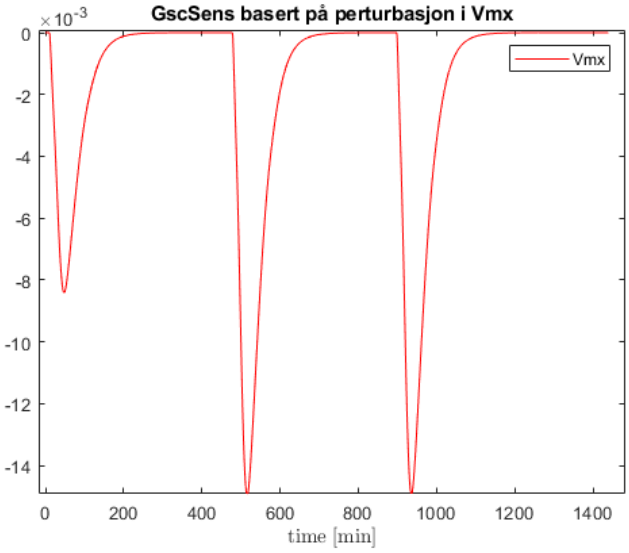


Figure 16: Sensitivitet for *GscSens* basert på perturbasjon i *Vmx*

Perturbasjonen av Vmx er 1% av sitt området, $[0.023 - 0.136]$. Alle parameterne vil gi ulik sensitivitet til *GscSens*, men Figur 16 viser bare hvordan en av parameterne påvirker sensitiviteten til målingene. Hele sensitivitetsmatrisen, S_y, vil vise hvor stor sensitivitet det er for hver parameter. En ser at i samme tidsrom det kommer måltid vil sensitiviteten få ett sprang. Dette gir en indikasjon på at systemet reagerer mer når det er en endring i systemet, som et måltid og injeksjon av insulin.

7.1.1 Maks- & minverdier

For å få en indikasjon på hvor stort område alle parameterne innehar er det definert maksimums- og minimums-verdier. Disse verdiene er generert fra *sub-*

ject-filene. Parameter skrevet til venstre i første kolonne er basert på hvordan MATLAB har implementert parameteren. Mens i høyre del er hvordan parameteren er skrevet i den metabolske modellen. I tillegg har parametere som i utgangspunktet har et null-intervall, fått et nytt intervall. Disse verdiene er visualisert med $*$, dette er basert på omtrentlige maks- og minverdier.

Parameter	**Max**	**Min**
k1/k_1	0.127702	0.05213
k2/k_2	0.217303	0.038924
Vg/V_G	2.158313	1.638485
m1/m_1	0.395022	0.066812
m2/m_2	0.6 $*$	0.2 $*$
m30/m_{30}	0.592532	0.100218
m4/m_4	0.4 $*$	0.1 $*$
Vi/V_I	0.088535	0.029655
kgri/k_{gri}	0.08785	0.016891
kabs/k_{abs}	1.057868	0.012962
f/f	1 $*$	0.5 $*$
BW/BW	98.65616	23.00894
kmin/k_{min}	0.022363	0.008032
kmax/k_{max}	0.08785	0.016891
b	0.866226	0.534785
c	0.196687	0.074888
kp1/k_{p1}	7.886045	2.662038
kp2/k_{p2}	0.010936	0.001756
kp3/k_{p3}	0.026514	0.002921
kp4/k_{p4}	0.1 $*$	0.02 $*$
kcounter/ξ	0.022413	0.003403
ki/k_i	0.023453	0.003846
kXGn/k_H	0.14184	0.058169
Gnb/H_b	82.54177	36.43148
Fcns/F_{cns}	1	0.5 $*$
Vm0/V_{m0}	9.49561	2.405612
Vmx/V_{mx}	0.135785	0.022605
r1/r_1	0.837735	0.780855
r3/r_3	1.526822	1.333509
Km0/K_{m0}	255.8713	184.6906
p2u/p_{2U}	0.072012	0.017592
Ib/I_b	165.6523	71.63808
FORTSETT PÅ NESTE SIDE		

Parameter	Max	Min
ka1/k_{a1}	0.005764	0.001759
ka2/k_{a2}	0.030419	0.00816
kd/k_d	0.019927	0.014035
ksc/k_{sc}	0.204863	0.060159
k01g/n	0.206857	0.095634
rho/ρ	1 $*$	0.5 $*$
kGSRs/σ_2	1.02234	0.128173
Gth/G_{th}	133.4108	110.7803
Gb/G_b	133.4108	110.7803
SRHb/SR_H^b	13.30819	4.49309
kGSRd/δ	0.656802	0.077159
SQgluc_k1/k_{h1}	0.036827	0.004427
SQgluc_kc1/k_{h2}	0.415742	0.039464
SQgluc_k2/k_{h3}	0.075487	0.013151
ke1/k_{e1}	0.001 $*$	0.0001$*$
ke2/k_{e2}	350 $*$	330 $*$
epsilon/ϵ	1 $*$	0.2 $*$
gamma_param/γ	1 $*$	0.2 $*$
alpha_param/α	0.1$*$	0.01 $*$
beta_param/β	0.2$*$	0.05 $*$
Sb/S_b	0.2 $*$	0
K	2.5$*$	1.5 $*$
h	133.4108	110.7803
kip/k_{ip}	0.3$*$	0.1 $*$
kip1/k_{ip1}	1.5 $*$	0.5 $*$
kip2/k_{ip2}	0.1 $*$	0.01 $*$
ksens/k_{sens}	0.7 $*$	0.3 $*$

Table 4: Maks/min-verdier generert av UVa/Padova's *subject*-filer

Singular Value Decomposition

Etter å ha regnet ut en dekomposisjon av singulærverdiene, *SVD*, for sensitivitetsmatrisen, S_y, kan en visualisere dette gjennom singulærverdier, *SV*, og *Right Singular Value*, *RSV*, i Figur 17. Øverste figur viser logaritmisk framstilling av *SV* i synkende rekkefølge. På figuren er det bare 49 verdier, det betyr at mange av de siste verdiene er lik eller ligger nærme 0. I dekomponeringen elimineres *SV* som ligger nærme 0. I nederste figur vises *RSV* som korresponderer til tilsvarende *SV*, og dette gir informasjon om lineære kombinasjoner av parametere og hvorvidt de er identifiserbar. De siste *SV* gir rom for de minst identifiserbare, og de første for de mest identifiserbare parametere.

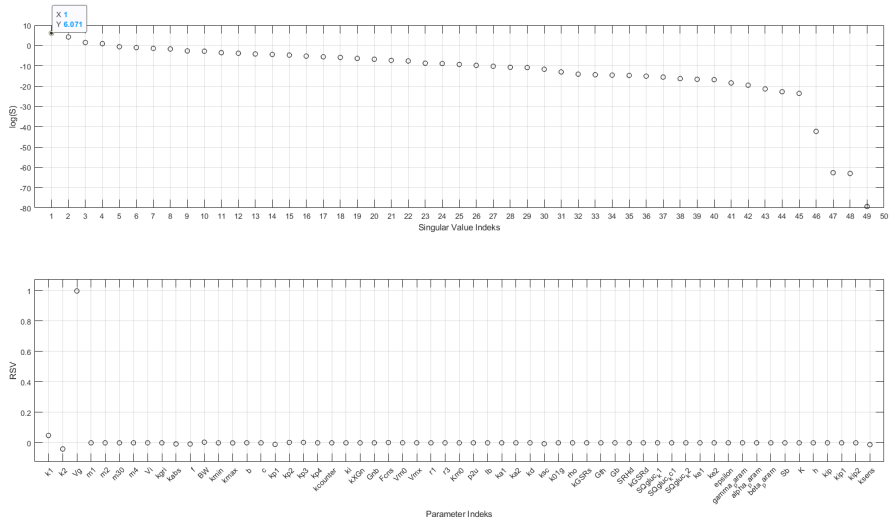


Figure 17: Singulærverdier for sensitivitetsmatrisen, S_y

Mulig det er vanskelig å observere ulike parameternavn samt singulærverdier, men *RSV* korresponderer her med singulærverdi indeks nr. 1. En kan observere at parameter *k1, k2, Vg* og *ksens* er ulik fra 0 i *RSV*. Det betyr at disse bidrar til størst singulærverdi for sensitivitetsmatrisen. Der *Vg* er viktigste parameteren. De parameterne som gir størst utslag er de modellen er mest sensitiv til. Disse kan videre bidra til å finne ut hvilke parametere som gir størst parametrisk usikkerhet. Flere grafiske framstillinger er gitt i Appendix D, disse er framstilt i større skala, og vil være enklere å visualisere. I Tabell 5 er det forsøkt å gi en oversikt over de parameterne som påvirker systemet mest, basert på de fem største singulærverdiene.

Singulærverdi Indeks	**Parameter**
1	*k1, k2, Vg, ksens*
2	*k2, Vg, kabs, f, BW, ksc, ksens*
3	*k2, kabs, f, BW, kp1, kp2, kp3, Fcns, Vmx, ksc, ksens*
4	*k2, kabs, f, b, c, kp1, kp2, kp3, Fcns, ksc, ksens*
5	*k2, kabs, f, BW, kmin, kmax, b, c, kp1, kp2, kp3, Fcns, ksc, ksens*

Table 5: Parametere som gir utslag hos de 5 største *SV*

Hvilke parametere som er av interesse ut fra Tabell 5 blir diskutert under Kapittel 8.

7.2 Sammenligning av simulator

Endel sammenligning har allerede blitt utført samtidig som modellen og simulatoren har blitt framstilt i Seksjon 6. Her blir resterende elementer som ikke nødvendigvis vil ha en innvirkning på framtidige hendelser sammenlignet.
Kun simuleringen av glukosekonsentrasjon i plasma ble vist i Seksjon 6.3.2 i Figur 12. For å bedømme om simulatoren var korrekt, ble også resten av de 25 tilstandene grafisk framstilt og sammenlignet. Visualisering av alle 25 tilstandene blir ikke framvist i rapporten, da alle viser tilfredstillende sammenfallende graf. Unntaket er tilstandene *Ipo, Hsc2, Iip1, Iip2* som viser avvik fra den opprinnelige simulatoren. *Iip1* og *Iip2* kan sees i den metabolske modellen som bidraget av insulin injisert intraperitonalt. Disse to tilstandene påvirker undersystemet for insulin direkte og via *Pancreatic insulin secretion/Insulinutskilling i bukspyttkjertel*. *Ipo* er intraperitonalt insulinleveranse fra *Pancreatic insulin secretion/Insulinutskilling i bukspyttkjertel* til undersystemet for insulin via $S_{IP,Port}$. *Hsc2* bidrar til administrasjon av glukagon i plasma fra *Subcutaneous glucagon delivery/Subkutant glukagonleveranse*-delen i modellen.
Siden det ikke er aktivert noen administrasjon av insulin injisert intraperitonalt skal i utgangspunktet *Ipo, Iip1* og *Iip2* være 0. Det er nevnt i Erlbeck, [9], at glukagonleveranse bare er implementert som en formalitet for å gjøre dokumentasjonen komplett. Derfor skal også *Hsc2* i teorien være 0 gjennom hele simuleringen. I Figur 18 er det visualisert sammenligning mellom de to ulike simulatorene for *Ipo*.

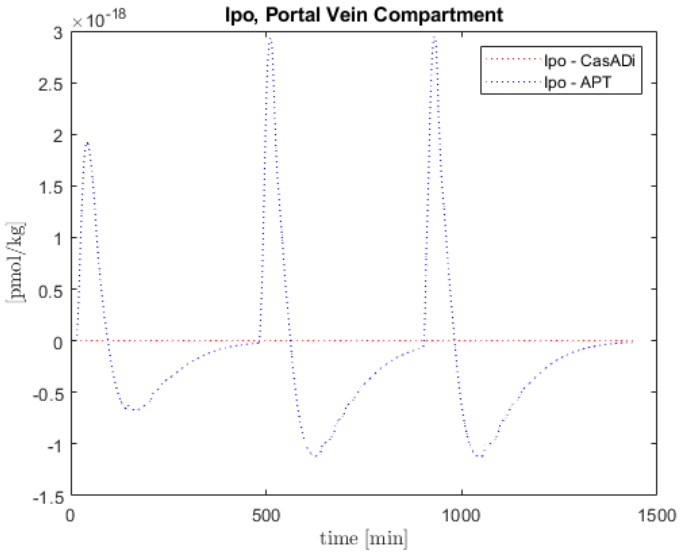


Figure 18: Sammenlign *Ipo* (Portal vein compartment)-verdi i CasADi mot Unstad's simulator, [25]

De resterende tre tilstandene viste lignende avvik. Dersom man undersøker

videre, så er avviket i størrelsesorden $(10^{-17} - 10^{-30})$. Dette utgjør såpass lite, og er dessuten utenfor datamaskinens mulighet for å regne ut. Så dette blir sett på som 0 for RK4-approksimasjonen. Dette vil derfor ikke trenge å gi utslag videre i systemet.

Kjøretid

Når simulatoren kjøres, kan en også sammenligne kjøretiden til simulatoren. Omgivelsene simulatoren ble testet på er MATLAB versjon 2019b sammen med CasADi v3.5.1. Disse verktøyene kjørte på datamaskin tilhørende NTNU og arbeidsplassen med Windows 10 som operativsystem. Ved å sammenligne APT-simulatoren med ny versjon laget i CasADi basert på RK4, ser en at CasADi utfører en betraktelig raskere simulering. I Tabell 6 er kjøretiden for de to ulike simulatorene listet opp. Tiden er basert på gjennomsnitt av 10 simuleringer med tilhørende standardavvik.

Sammenligning	
APT-simulator	$40.5 \pm 0.5sek$
CasADi-simulator RK4	$2.8 \pm 0.3sek$

Table 6: Sammenlign kjøretid til simulator

7.3 Resultat av NLP fra CasADi

Når optimaliseringsverktøyet kjører sin *solver* og løser systemet med tilhørende kostfunksjon blir resultatet vist som i Figur 19.

```
iter    objective    inf_pr   inf_du lg(mu)  ||d||  lg(rg) alpha_du alpha_pr  ls
   0 0.0000000e+000 4.71e+003 0.00e+000  -1.0 0.00e+000    -  0.00e+000 0.00e+000   0
   1 0.0000000e+000 9.81e-001 0.00e+000  -1.0 1.95e+003    -  1.00e+000 1.00e+000h  1
   2 0.0000000e+000 1.78e-005 0.00e+000  -1.7 5.74e+000    -  1.00e+000 1.00e+000h  1
   3 0.0000000e+000 1.17e-010 0.00e+000  -8.6 1.15e-004    -  1.00e+000 1.00e+000h  1
Cannot recompute multipliers for feasibility problem.  Error in eq_mult_calculator

Number of Iterations....: 3

                                   (scaled)                 (unscaled)
Objective...............:  0.0000000000000000e+000   0.0000000000000000e+000
Dual infeasibility......:  0.0000000000000000e+000   0.0000000000000000e+000
Constraint violation....:  1.1709744285326451e-010   1.1709744285326451e-010
Complementarity.........:  0.0000000000000000e+000   0.0000000000000000e+000
Overall NLP error.......:  1.1709744285326451e-010   1.1709744285326451e-010


Number of objective function evaluations             = 4
Number of objective gradient evaluations             = 4
Number of equality constraint evaluations            = 4
Number of inequality constraint evaluations          = 0
Number of equality constraint Jacobian evaluations   = 4
Number of inequality constraint Jacobian evaluations = 0
Number of Lagrangian Hessian evaluations             = 3
Total CPU secs in IPOPT (w/o function evaluations)   =      0.260
Total CPU secs in NLP function evaluations           =      0.087

EXIT: Optimal Solution Found.
      solver  :   t_proc      (avg)    t_wall      (avg)    n_eval
       nlp_f  |   3.00ms (750.00us)   2.99ms (746.50us)         4
       nlp_g  |   4.00ms (  1.00ms)   3.98ms (995.75us)         4
  nlp_grad_f  |   8.00ms (  1.60ms)   7.98ms (  1.60ms)         5
  nlp_hess_l  |  26.00ms (  8.67ms)  25.94ms (  8.65ms)         3
   nlp_jac_g  |  60.00ms ( 12.00ms)  59.85ms ( 11.97ms)         5
       total  | 348.00ms (348.00ms) 348.08ms (348.08ms)         1
```

Figure 19: Oversikt over løsning av NLP fra MATLAB

Her er det viktig å legge merke til at optimal løsning er funnet, samt at iterasjonen øverst viser små verdier. Da er det spesielt viktig å se på *inf_pr*. Denne sier noe om hvor godt *Collocation*-metoden klarer å gjette seg fram til en god løsning. Hvis denne er høy, klarer ikke kostfunksjonen å bli tilfredsstilt. Her starter denne verdien litt høy, men reduseres kraftig etter første iterasjon. Ved at *inf_pr* holder seg lav gjennom iterasjonene, tyder dette på at optimal løsning ble funnet innenfor et *feasible* område.
Resten av tilbakemeldingen fra NLP-*solver* er rettet mot oppsettet av problemet. Hvor mange *equality* og *inequality* begrensninger som er definert. Til slutt er en oversikt over hvor lang tid NLP-*solver* bruker på å kalkulere ulike deler av systemet.

Collocation-metoden mot RK4

Et lite avvik oppstår for utvalgte tilstander når *Collocation*-metoden blir sammenlignet med RK4. Dette svarer til samme fenomen som vist over når MATLAB-simulatoren og RK4 ble sammenlignet. Størrelsesorden for avviket i denne sammenheng blir 10^{-30}, og vil tilsvare 0 i praksis når datamaskinen regner ut. Derfor burde ikke dette skape forvirring om at det er feil i modellen. I Figur 20 er det fremstilt hvordan dette avviket oppfører seg.

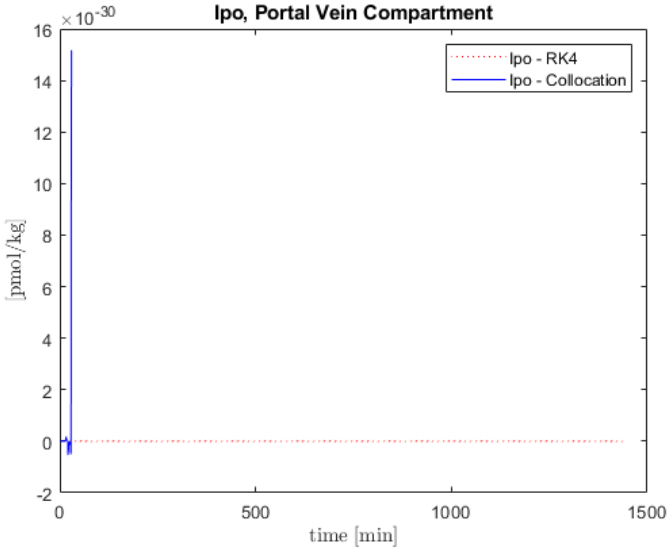


Figure 20: Sammenlign *Ipo* (Portal vein compartment)-verdi med RK4-simulator mot IRK

Dette avviket gjelder tilstandene *Ipo, Hsc2, Iip1* og *Iip2*. De samme tilstandene som ga utslag på sammenligningen av simulatoren implementert i MATLAB og RK4-simulatoren.

8 Diskusjon

I denne oppgaven er det tatt beslutninger som har vært med på å forme retningen på oppgaven. Visse valg finnes det alternative løsninger på, som kanskje kunne gitt et bedre resultat. Her blir valgene som har blitt tatt nevnt sammen med andre potensielle muligheter.

8.1 Parametrisk usikkerhet

I denne oppgaven skulle et sett med usikre parametere defineres. I litteraturen var det få parametere som var beskrevet med parametrisk usikkerhet. Aicha og Mourad bruker i sitt studie, [4], en måte å definere et sett av parametrisk usikkerhet. Her defineres alle parametere til å være usikre, og en kan trekke paralleller til hvordan utførelsen av perturbasjonen er gjort. Et maksimums- og minimumsintervall er definert. Forskjellen er at de bruker Bergman's minimal modell, som kun består av 3 differensiallikninger. Mot vår modell som består av 25 differensiallikninger med til sammen 59 parametere. Det blir betraktelig enklere å finne et oversiktlig system for å definere et slikt parametersett.
Vårt parametersett består av individuelle og felles parametere. Hvordan parametrisk usikkerhet er definert for disse to ulike gruppene av parametere kan være forskjellig. For individuelle parametere er det utført målinger og forsøk for å komme fram til en slik verdi. Det betyr ikke at den parameteren er sikker, for den kan fortsatt være usikker basert på at den kan variere over tid. Siden de er definert til en konstant verdi, gir dette uttrykk for at disse utvalgte individuelle parameterne er enklere å måle. Hvor målbare de er vil kunne variere. Slik som parameteren *BW*, (kroppsvekt), som vil kunne måles hyppig, mens *Vmx* vil kunne måles sjeldnere. De parameterne som er felles vil basere seg på individuelle variasjoner.

Utenom det som er nevnt under *Usikre parametere* i Seksjon 3.2 er det ikke funnet mer utdypende beskrivelse av usikre parametere. Basert på dette er det også vanskelig å si om et intervall er definert riktig. Vi har ikke direkte tilgang til alle parameterne, og for lite forskning er blitt utført for hver enkelt parameter. Derfor vil det være vanskelig å si hvilke parameter som er definert som usikre og dens intervall. Intervallet i denne oppgaven blir definert av maksimums- og minimumsverdier til hver parameter hos en gjennomsnittspasient fra UVa/-Padova's datasett. Dette er kanskje ikke verdier som vil gi 100% korrekt resultat, men fram til det blir funnet gode estimater for parametrisk usikkerhet, vil dette være godt nok. Så lenge intervallet er estimert fornuftig.

Virkning av sensitivitetsanalysen

For å finne ut mer om hvilke parametere systemet er sensitiv til, og eventuelt fastslå parametere som er usikre er det utført en sensitivitetsanalyse.
Basert på Tabell 5 i Seksjon 7.1.1 er det foreslått ulike parametere som er med på å påvirke singulærverdi nr. 1-5. De største singulærverdiene gir utslag for delene av systemet som mest sensitiv. Parameterne har ulik grad av påvirkning, men det er visse parametere som ser ut til å gå igjen. Dette er typisk tidskonstanter som *ksc, ksens, kp1, kp2*, osv. Det har tidligere vært antatt at tidskonstanter vil kunne variere, derfor er det ingen overraskelse at nettopp disse vil bidra til parametrisk usikkerhet. *ksc* og *ksens* er tidskonstanter der det forekommer målinger. Dette kan være avhengig av sensor som blir brukt eller alder på infusjonssettet som en bruker. Dette er nevnt under kapittelet om usikre parametere 3.2. En slik del av systemet vil være et naturlig sted å ha forsinkelser som er vanskelig å estimere.
Parameteren *Vg* vil også påvirke endel, og er med på å si om distribusjonsvolumet av glukosekonsentrasjon i blodet. Det er flere parametere som ikke blir nevnt, men som ut ifra singulærverdi dekomposisjon *SVD* ikke har like stor påvirkning som parameterne nevnt her.

Det som overrasker mest er hvor liten påvirkning insulinsensitiviteten, *Vmx*, har i denne testen og analysen. En kan av den grunn ikke utelukke at det vil være parametrisk usikkerhet for *Vmx*. Basert på Seksjon 3.2 nevnes det at *Vmx* er en parameter som kan variere fra tidspunkt på døgnet, og ikke bare fra pasient til pasient. I denne oppgaven er *Vmx* satt konstant gjennom hele simuleringen på ett døgn. Dette er gjort for å forenkle fremgangsmåten av simulatoren. Ved videre utvikling av simulatoren, bør *Vmx* kunne variere med tid.
Fordelingsparameteren ϵ er også en parameter som er antatt å være usikker, men som ikke gir utslag på denne sensitivitetsanalysen. Dette er en parameter som resulterer i at den er ikke-identifiserbar basert på denne analysen. Grunnen til dette er at ϵ er en fordelingsparameter ved intraperitonalt insulinleveranse. I simulatoren er det kun simulert med subkutan leveranse av insulin. Det kan hende at ϵ bidrar til større sensitivitet dersom dette blir endret, men en slik simulering er ikke utført i denne oppgaven. Det samme gjelder parameteren α, som er forsinkelse mellom glukosesignal og insulinsekresjon. Denne delen av metabolske modellen er bare aktiv for pasienter med diabetes Type 2, og er derfor ikke implementert i denne oppgaven.

Som nevnt tidligere er sensitivitetsanalysen basert på parametriske verdier som er generert fra *subject*-filene fra UVa/Padova-simulatoren. Dette gjør at genererte maksimum- og minimumsverdier kan bestå av et intervall som kan avvike fra et reelt område for parameteren. Parametere som ikke har et definert maks-/min-intervall fra *subject*-filene er det utført gjetninger for disse intervallene.

Dette underbygger at de nye maks- og minverdiene ikke vil være 100% korrekt. De viktigste parameterne vil nok være med å påvirke systemet uansett, og det er disse som har blitt nevnt over.

Det har vært nevnt under Seksjon 3.2.1 at det finnes sikre parametere i modellen. Selv om de er sikre kan de bidra til stor sensitivitet. Blant annet blir parameteren *BW* listet opp i Tabell 5 til å gi stor sensitivitet til systemet. Selv om det gir stor sensitivitet å endre vekten med 1% av sitt området, vil denne parameteren være målbar, og enkel å håndtere i systemet. Derfor vil det ikke gi stor innvirkning siden den blir antatt som en sikker parameter.
Sensitivitetsanalysen i denne oppgaven er mer brukt som en pekepinn på hvordan man kan utføre en slik analyse på glukose-insulin modellen. Dersom en har alle korrekte verdier for hver parameterer i tillegg til sine maks- og minverdier vil en oppnå en mer nøyaktig analyse.

8.2 Valg av software

Optimaliseringsverktøyet CasADi er et verktøy som har blitt brukt i denne oppgaven. Andre verktøy har blitt nevnt som mulige alternativer. Disse vil kanskje gi like gode resultater som CasADi, eller kanskje enda bedre. Stort sett når en velger et verktøy for implementering av optimaliseringsproblem er det viktig å fokusere på kjøretiden til koden. Derfor blir det tatt avstand fra simulatoren implementert i MATLAB av Erlbeck, [9], og Unstad, [25]. Basert på resultatene i Seksjon 7.2, ser man at kjøretiden til en simulering i CasADi er 20 ganger raskere enn ved MATLAB-simulering. Hvis en skulle utført en modellbasert regulering basert på simulatoren i MATLAB vil det kreve lang kjøretid. For hvert tidssteg må det regnes ut en ny optimalisering.
En annen viktig faktor for at CasADi ble valgt som verktøy i min oppgave, var at medveileder, Sebastien Gros, har mye kunnskap innen CasADi. Dette gjør at det blir lettere å sette seg inn i forståelse og syntaks, og det vil være lettere å vite om noe er implementert feil.
Dersom medveileder ikke hadde hatt kunnskap om CasADi, ville YALMIP være et verktøy som burde blitt utforsket grundigere. Siden det er et verktøy laget for optimalisering av kompliserte modeller og robust regulering, vil det antas å være bedre egnet. Om YALMIP kalkulerer hurtigere enn CasADi er det ikke funnet noe litteratur på. Dette ville kanskje fått en større innvirkning når en robust regulator skulle blitt implementert.

8.3 Valg av regulator

Ideelt sett skulle en ferdigimplementert MPC blitt evaluert med gode optimaliseringsegenskaper basert på robust regulering. For å utforske om andre regula-

torer vil gi bedre resultat, må en se på de ulike regulatorene som har vært nevnt tidligere i oppgaven. Selv om robuste egenskaper ikke ble implementert her, vil det fortsatt være en viktig faktor for valg av regulator. PID regulatoren er kanskje den enkleste formen, men vil nok trolig ikke gi tilstrekkelig funksjonalitet mot robuste omgivelser. Både MPC og H_∞ tilbyr gode robuste egenskaper. H_∞ blir sett på som godt egnet til parametrisk usikkerhet, men krever også mer forståelse både av modellen og det matematiske rundt H_∞. Her kan Aicha og Mourad sitt studie, [4], igjen bli trukket inn. Selv om de har en minimal modell sammenlignet med modellen som blir brukt i denne masteroppgaven, klarer de å oppnå robust regulering med usikre parametere for H_∞-regulering. De oppnår å stabilisere seg på et glukosekonsentrasjonsnivå på $81mg/dL$, som også er målet for APT. Dersom det var et ønske om å oppnå gode resultater for robust regulering, burde en slik forenklet modell blitt tatt i bruk.
Hvis en skal sammenligne H_∞ med MPC-metoden så var det lite forkunnskaper og forståelse overfor H_∞-kontroll i begynnelsen av oppgaven. Dessuten var det bedre kjennskap til hvordan en MPC-regulator fungerer teoretisk. Med god hjelp fra veileder med bakgrunn innen optimalisering og MPC, ble denne regulatoren et naturlig valg.

8.4 Grenser på tilstander & pådrag

Veien mot å implementere en fullverdig *Collocation*-metode og en robust regulator kreves det at grenser på tilstander og pådrag blir definert. Målet er å unngå at tilstandene beveger seg utenfor et slikt intervall. Basert på APT er det satt grenser på hva som er akseptabelt for glukosekonsentrasjon i blodet, G. Grensene er satt til $4 - 8mmol/L$ eller $72 - 144mg/dL$, som er enhet i denne oppgaven. Det er også blitt satt en referanse som en ønsker at glukosekonsentrasjonen skal holde seg til. $G_{ref} = 4.5mmol/L$ / $G_{ref} = 81mg/dL$.
I denne oppgaven er insulininjeksjon det eneste pådraget som vil være mulig å styre og sette begrensninger på i regulatoren. Senere vil det kanskje bli mulig å regulere glukagoninjeksjon for å motvirke lavt blodsukker. Målet er ikke at en ønsker å redusere insulin- eller glukagonforbruket basert på økonomiske prinsipper, men heller gi korrekte doseringer slik at pasienten oppnår best mulig kontroll på sykdomsbildet og oppnår best mulig livskvalitet.

8.5 Systemet i helhet

Det som er blitt gjort i denne oppgaven er å omforme Unstad, [25], sin simulator til en raskere simulator basert på Implisitt Runge-Kutta og *Collocation*-metoden. I Figur 15 i Seksjon 6 er det vist hvordan *Collocation*-metoden fungerer som en simulator. Med andre ord er det ikke etablert en optimaliseringsalgoritme. Målet i starten av en slik prosess er at metoden klarer å simulere

allerede kjente verdier. Hvis en da kjører *Collocation*-metoden i lukket sløyfe vil det tilsvare samme resultat som RK4-simulatoren produserer. En ny simulator var ikke målet i seg selv, men et viktig steg på veien for å kunne definere et regulerbart system. Videre vil det være viktig å definere en mer komplisert kostfunksjon som tar hensyn til de tilstandene en ønsker å regulere. Typisk ønsker man å regulere glukosekonsentrasjonen, G, nær en verdi som oppfyller normoglykemi. Det vil si at kostfunksjonen vil ligne litt som i Likning 12:

$$J = \frac{1}{2}(G - G_{ref})^2 + \frac{1}{2}(IIR - IIR_{ref})^2 \tag{12}$$

Videre kan grenser for tilstandene og verdier for referansene bli definert. G er allerede definert i 8.4, men det kreves flere grenser.
For å kunne gjøre optimaliseringsproblemet realiserbart bør slakkvariabler, ϵ, som nevnt i Seksjon 2.4.2, implementeres. Sammen med Q, R og S vil en kunne tune regulatoren slik at den oppfører seg innenfor grensene som er satt.
Disse siste stegene er hva som mangler for å oppnå et helhetlig optimaliseringsproblem i lukket sløyfe. Med disse utvidelsene vil systemet være mulig å bli testet med datasettet fra de 30 *subject*-filene i UVa/Padova-simulatoren, og videre andre datasett. Ved å komme nærmere robust regulering vil en kunne ta modellen fra *in silico* og nærme seg *in vivo*.

Konsekvens med å fortsette med RK4

I Seksjon 6.3.2 blir RK4 implementert etter forventninger, så hva ligger til grunn for å bruke mye tid på å implementere en ny approksimasjon? Tidligere ble det nevnt at systemet viste seg å være et *stiff* system. Dette er en av grunnene til å endre metode, men det at et systemet er *stiff* vil ikke si at det er umulig å utføre en optimalisering på modellen. Problemet er at det krever lang kjøretid med mange utregninger dersom systemet er *stiff*, og RK4 blir brukt som approksimeringsmetode. Denne metoden er ikke egnet for slike systemer. Mye tid kunne blitt spart dersom RK4 hadde blitt tatt i bruk videre, men med råd fra medveileder ble det anbefalt å lage et solid grunnlag for systemets oppbygning. Derfor ble RK4 forkastet, og en IRK ble implementert for å kunne definere et system som er mulig å jobbe videre med i senere tid.

9 Konklusjon

Den enkleste måten å implementere en fullverdig reguleringsmodell for en robust regulator ville vært å ta i bruk Frøyen sin Simulink-modell og simulator. Denne oppgaven inneholdt allerede en metode for å regulere systemet, både med en PID og MPC. Ved å ta utgangspunkt i denne simulatoren, ville det vært oppnåelig å implementere en robust regulator. Problemet med denne simulatoren er at den krever stor datakraft og bruker lang tid for å kjøre glukose-insulin modellen. Robuste egenskaper til en regulator utvider funksjonaliteten betraktelig. Derfor var det viktig å bygge opp en ny modell og simulator for å redusere tidsbruken. Erlbeck klarte i sin masteroppgave å få ned tidsbruken, men fremdeles antar vi at en fullverdig simulering tar for lang tid. I startfasen av vår oppgave fant vi det derfor avgjørende å se på egne verktøy som er egnet for optimalisering. Med CasADi har vi klart å redusere kjøretiden av glukose-insulin simulatoren til 2-3 sekunder som er en betydelig forbedring. Hadde vi ikke oppnådd redusert kjøretid, hadde det vært vanskelig å starte med robust regulering.

Sensitivitetsanalysen som er utført i denne oppgaven bidrar til å gi en indikasjon på hvilke parametere som kan være usikre. Analysen kan bli utført grundigere med mer korrekte estimater, og vil være til stor hjelp når en videre skal definere ett sett med usikre parametere. Når parameterne er definert, kan en bruke dette i robust regulering. Dette kan ha betydning for studenter som senere ønsker å jobbe videre med denne oppgaven.

Målet med denne oppgaven var å finne ut hvilke modellbaserte regulatorer som tar usikre parametere i betraktning, for så å implementere denne regulatoren, og teste den på simulatoren. En Robust MPC-regulator ble antatt å være tilstrekkelig for å oppnå kravene for en modellbasert regulator. Denne regulatoren ble ikke implementert i vår oppgave, men basert på *Collocation*-metode er det lagt til rette for at det skal være enklere å implementere ved senere studier. Dermed kan vi si at det er to vesentlige delmål som er oppnådd i denne oppgaven. Kjøretiden av modellen er redusert, og mye er lagt til rette for videre studier for implementering av robust regulator i en klinisk relevant simulator.

10 Videre arbeid

I denne rapporten er det flere punkter som bør være inkludert under videre arbeid. Enten det er mål som er utenfor målsetting, eller det ikke kom med i denne rapporten. Dette er momenter som andre studenter kan jobbe videre med i framtidige oppgaver.
Først bør det være fokus på å fullføre en modellbasert reguleringsalgoritme relatert til denne oppgaven. Ferdigstille Implisitt Runge-Kutta metoden med å lage en utvidet og mer komplisert kostfunksjon. Dette kan oppnås med å legge til målingen som en ønsker å optimalisere. I utgangspunktet gjelder dette glukosekonsentrasjonen, G, som har vært nevnt tidligere. Kostfunksjonen kan bli utvidet ytterligere ved å legge til slakkvariabler. Ved hjelp av *Q, R* og *S* har man en måte å tune forholdet mellom pådrag, tilstand og effekten av slakkvariabler.
Når kostfunksjonen er definert etter kravspesifikasjonen, bør alle grenser for tilstander og pådrag bli definert. Dette bør være basert på litteratur eller ønskelig estimat fra APT.

I litteraturen er det få parametere i glukose-insulin modellen som er definert som usikre parametere. Det er viktig å klarlegge hvilke parametre som med stor sannsynlighet er usikre. Dette vil være et omfattende studie som går inn på hver enkelt parameter i systemet, og definerer dens usikkerhet. En kan da få satt grensene og intervallet til hver enkelt parameter og lage en Robust MPC. I tillegg bør parametere bli implementert slik at de kan variere med tiden. Hvilke parametere dette gjelder bør være basert på tidligere forskning.

Siste videre forbedringer til systemet, er å introdusere glukagon som pådrag i en Robust MPC. Ved at man kan injisere glukagon kan en ta kontroll over hele prosessen som bukspyttkjertelen har ansvar for. Glukagon blir da et pådrag på lik linje som insulin. Ved en slik injeksjon kan det være enklere og hurtigere å oppnå et referansepunkt for glukosenivå, og for å unngå hypoglykemi eller hyperglykemi.

11 Referanser

[1] All-Party Parliamentary Group for Diabetes: Basic mechanism of blood glucose level control in the pancreas. URL: https://diabetesappg.wordpress.com/2016/05/10/the-technology-jigsaw-for-the-artificial-pancreas/. Accessed: 06-09-2019.

[2] Artificial Pancreas Trondheim, Main Page. URL: https://www.apt-norway.com/. Accessed: 14-11-2019.

[3] Store Medisinske Leksikon: Diabetes. URL: https://sml.snl.no/diabetes. Accessed: 01-10-2019.

[4] H. Aicha and A. Mourad. H-infinity controller design for blood glucose regulation in diabetes patients in the presence of uncertain parameters. In *2015 3rd International Conference on Control, Engineering & Information Technology (CEIT)*, pages 1–6. IEEE, 2015.

[5] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[6] D. U. Campos-Delgado, M. Hernández-Ordoñez, R. Femat, and A. Gordillo-Moscoso. Fuzzy-based controller for glucose regulation in type-1 diabetic patients by subcutaneous route. *IEEE Transactions on Biomedical Engineering*, 53(11):2201–2210, 2006.

[7] C. Cobelli, E. Renard, and B. Kovatchev. Artificial pancreas: past, present, future. *Diabetes*, 60(11):2672–2682, 2011.

[8] C. Dalla Man, R. A. Rizza, and C. Cobelli. Meal simulation model of the glucose-insulin system. *IEEE Transactions on biomedical engineering*, 54(10):1740–1749, 2007.

[9] N. Erlbeck. Design and implementation of a flexible simulation framework for glucose metabolism. Master's thesis, Norges Teknisk-Naturvitenskapelige Universitet, 2016.

[10] B. Foss and T. A. N. Heirung. *Merging Optimization and Control*. PhD thesis, Norwegian University of Science and Technology - Department of Engineering Cybernetics, 2016.

[11] A. Fredriksson, A. Forsgren, and B. Hårdemark. Minimax optimization for handling range and setup uncertainties in proton therapy. *Medical physics*, 38(3):1672–1684, 2011.

[12] H. E. Frøyen. Algorithms for closed-loop glucose control in diabetes and intensive-care patients. Master's thesis, Norges Teknisk-Naturvitenskapelige Universitet, 2014.

[13] G. Gede. The Direct Collocation Method for Optimal Control. URL: https://research.engineering.ucdavis.edu/biosport/wp-content/uploads/sites/24/2014/06/direct-collocation-presentation.pdf. Accessed: 28-11-2019.

[14] D.-W. Gu, P. Petkov, and M. M. Konstantinov. *Robust control design with MATLAB®*. Springer Science & Business Media, 2005.

[15] I. International Diabetes Federation. Diabetes facts. URL: https://www.idf.org/aboutdiabetes/what-is-diabetes/facts-figures.html. Accessed: 20-09-2019.

[16] H. K. Khalil. *Nonlinear Systems Third Edition*. Pearson Education Limited 2015, 2002.

[17] K. H. Kienitz and T. Yoneyama. A robust controller for insulin pumps based on h-infinity theory. *IEEE Transactions on Biomedical Engineering*, 40(11):1133–1137, 1993.

[18] J. Löfberg. Yalmip: A toolbox for modeling and optimization in matlab. In *Proceedings of the CACSD Conference*, volume 3. Taipei, Taiwan, 2004.

[19] D. Q. Mayne, E. C. Kerrigan, E. Van Wyk, and P. Falugi. Tube-based robust nonlinear model predictive control. *International Journal of Robust and Nonlinear Control*, 21(11):1341–1353, 2011.

[20] J. Nocedal and S. J. Wright. *Numerical Optimization, 2nd edition*. Springer Series in Operations Research, 2006.

[21] J. Puschke, A. Zubov, J. Kosek, and A. Mitsos. Multi-model approach based on parametric sensitivities–a heuristic approximation for dynamic optimization of semi-batch processes with parametric uncertainties. *Computers & Chemical Engineering*, 98:161–179, 2017.

[22] L. F. Shampine and C. W. Gear. A user's view of solving stiff ordinary differential equations. *SIAM review*, 21(1):1–17, 1979.

[23] M. Sørensen, F. Arneberg, T. Line, and T. Berg. Cost of diabetes in norway 2011. *Diabetes research and clinical practice*, 122:124–132, 2016.

[24] O. M. Staal, A. L. Fougner, S. Sælid, and Ø. Stavdahl. Glucose-insulin metabolism model reduction and parameter selection using sensitivity analysis. In *2019 American Control Conference (ACC)*, pages 4104–4111. IEEE, 2019.

[25] K. A. F. Undstad. Methods for fault detection in an artificial pancreas. Master's thesis, Norges Teknisk-Naturvitenskapelige Universitet, 2018.

[26] A. Waechter. Ipopt software library. URL: https://github.com/coin-or/Ipopt. Accessed: 16-11-2019.

[27] Wikipedia. Robust Control. URL: https://en.wikipedia.org/wiki/Control_theory#Classical_control_theory. Accessed: 05-10-2019.

[28] R. J. Young, W. J. Hannan, B. M. Frier, J. M. Steel, and L. J. Duncan. Diabetic lipohypertrophy delays insulin absorption. *Diabetes Care*, 7(5):479–480, 1984.

[29] L. Yu and J. Chu. An lmi approach to guaranteed cost control of linear uncertain time-delay systems. *Automatica*, 35(6):1155–1159, 1999.

Appendix A Parametere

Process	Symbol	Meaning	Unit
Glucose subsystem	V_G	Distribution volume of glucose	dl/kg
	k_1	Rate parameter from plasma to tissue	min^{-1}
	k_2	Rate parameter from tissue to plasma	min^{-1}
	G_b	Basal plasma glucose concentration	mg/dl
	G_{tb}	Basal glucose mass at slowly equilibrating (peripheral) tissues	mg/kg
	G_{pb}	Basal plasma glucose mass	mg/kg
Insulin subsystem	V_I	Distribution volume of insulin	l/kg
	m_1	Rate parameter	min^{-1}
	m_2	Rate parameter	min^{-1}
	m_{30}	Rate parameter	min^{-1}
	m_4	Rate parameter	min^{-1}
	m_5	Rate parameter	$min \cdot kg/pmol$
	HE_b	Basal hepatic extraction of insulin	*dimensionless*
	I_b	Basal plasma insulin concentration	$pmol/l$
	I_{lb}	Basal insulin mass in liver	$pmol/kg$
	I_{pb}	Basal insulin mass in plasma	$pmol/kg$
Pancreatic insulin secretion	S_b	Basal pancreatic insulin secretion	$pmol/kg/min$
	K	Pancreatic responsivity to the glucose rate of change	$pmol/kg$ $per\ mg/dl$
	α	Delay between glucose signal and insulin secretion	min^{-1}
	β	Pancreatic responsivity to glucose	$pmol/kg/min$ $per\ mg/dl$
	γ	Portal vein to liver secretion factor/transfer rate constant	min^{-1}
	h	Beta-cell insulin secretion threshold	mg/dl
Subcutaneous insulin delivery	k_d	Rate constant of insulin degradation	min^{-1}
	k_{a1}, k_{a2}	Rate constants of polymeric and monomeric insulin absorption	min^{-1}
	IIR_b	Basal subcutaneous insulin infusion needed to maintain diabetic patient at basal steady state	$pmol/kg/min$
	I_{sc1ss}, I_{sc2ss}	Basal values in the subcutaneous compartments	$pmol/kg$
Intraperitoneal insulin delivery	ϵ	Distribution parameter between the portal vein and the remaining blood circulation	*dimensionless*
	k_{ip1}, k_{ip2}	Rate constants for each of the peritoneal compartments	min^{-1}
Glucagon subsystem	n	Glucagon clearance rate	min^{-1}
	H_b	Basal plasma glucagon concentration	ng/l

FORTSETT PÅ NESTE SIDE

Process	Symbol	Meaning	Unit
Pancreatic glucagon secretion	ρ	Rate parameter accounting for delay between static glucagon secretion and plasma glucose	min^{-1}
	σ	Alpha-cell responsivity to glucose level	$ng/l/min$ $per\ mg/dl \cdot l/pmol$
	σ_2	Alpha-cell responsivity to glucose level	$ng/l/min$ $per\ mg/dl$
	δ	Alpha-cell responsivity to glucose rate of change	$ng/l \cdot dl/mg$
	SR_H^b	Basal glucagon secretion	$ng/l/min$
	SR_H^{sb}	Basal static glucagon secretion	$ng/l/min$
	SR_H^{db}	Basal dynamic glucagon secretion	$ng/l/min$
Subcutaneous glucagon delivery	k_{h1}, k_{h2}, k_{h3}	Rate parameters describing subcutaneous glucagon kinetics	min^{-1}
	H_{sc1b}, H_{sc2b}	Basal glucagon concentration in the two subcutaneous compartments	ng/l
Gastro-intestinal tract	k_{max}, k_{min}	Maximum and minimum emptying rate parameters	min^{-1}
	k_{abs}	Rate of glucose absorption from the gut	min^{-1}
	k_{gri}	Rate of grinding	min^{-1}
	f	Fraction of intestinal absorption which actually appears in plasma	*dimensionless*
	b, c	Rate of appearance parameters	*dimesnionless*
	BW	Body weight	kg
Liver	k_{p1}	Endogenous glucose production at zero glucose and insulin	$mg/kg/min$
	k_{p2}	Liver glucose effectiveness	min^{-1}
	k_{p3}	Parameter governing amplitude of insulin action on the liver	$mg/kg/min$ $per\ pmol/l$
	k_{p4}	Parameter governing amplitude of portal insulin action on the liver	$mg/kg/min$ $per\ pmol/kg$
	ξ	Glucose production rate due to glucagon action	$mg/kg/min$ $per\ ng/l$
	k_i	Rate parameter accounting for delay between insulin signal and insulin action	min^{-1}
	k_H	Rate parameter accounting for delay between glucagon concentration and glucagon action	min^{-1}
Glucose utilization	F_{cns}	Glucose uptake by brain and erythrocytes	$mg/kg/min$
	V_{m0}	Minimum distribution volume used in the calculation of glucose utilization	$mg/kg/min$
	V_{mx}	Parameter governing amplitude of insulin action on glucose utilization (insulin sensitivity)	$mg/kg/min$ $per\ pmol/l$
	K_{m0}	Insulin-dependent utilization	mg/kg
	p_{2U}	Rate constant of insulin action on the peripheral glucose utilization	min^{-1}
	$EGPb$	Basal endogenous glucose production	$mg/kg/min$

FORTSETT PÅ NESTE SIDE

Process	Symbol	Meaning	Unit
	G_{th}	Glucose concentration controlling glucagon secretion (hypoglycemic threshold)	mg/dl
	r_1	Model parameter	*dimensionless*
	r_2	Model parameter	*dimensionless*
Glucose renal excretion	k_{e1}	Glomerular filtration rate	min^{-1}
	k_{e2}	Renal threshold of glucose	mg/kg
Glucose kinetics	k_{ip}	Rate transfer constant from the plasma glucose compartment to the peritoneal glucose compartment	min^{-1}
	k_{sc}	Rate transfer constant from the plasma glucose compartment to the subcutaneous glucose compartment	min^{-1}
	k_{sens}	Rate parameter accounting for glucose leaving the sensor compartment	min^{-1}

Table 7: Liste og beskrivelse av parametere i modellen hentet fra Erlbeck, [9].

Appendix B Ny metabolsk modell

Her er det forøkt å visualisere en enklere metabolsk modell delt inn i tre deler. Glukosesystem, insulinsystem og glukagonsystem. Alle likninger er fjernet, og det vises bare de variablene som beveger seg fra en del til et annet. Disse variablene kan krysskobles både inn og ut av en seksjon. Dette er vist med pil med hvit sirkel. Pil med sort sirkel definerer at noe fra denne seksjonen blir nedbrutt. Mens pil som ikke kommer fra en seksjon definerer input til systemet. Enten i form av insulin eller måltid.

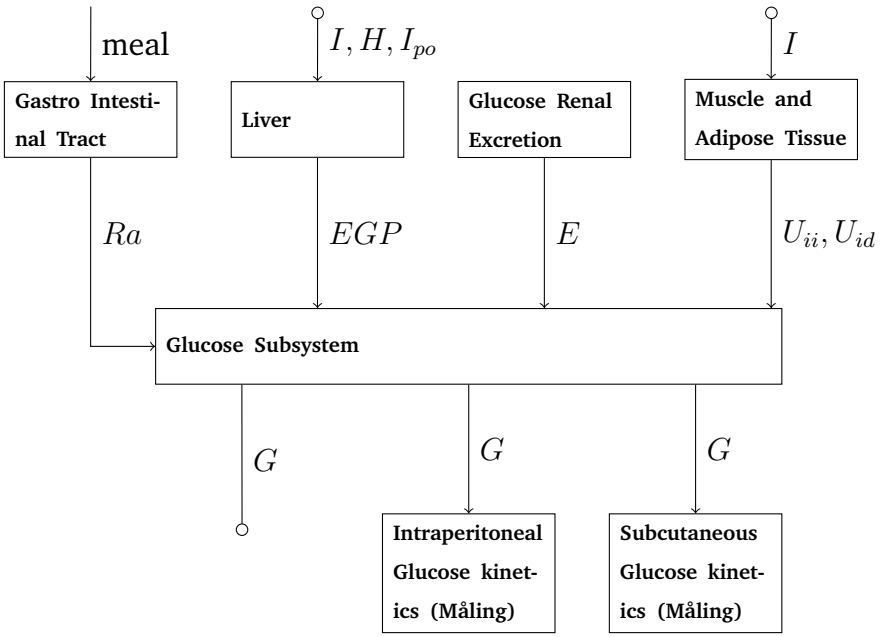


Figure 21: Ny metabolsk modell: Glukosesystem

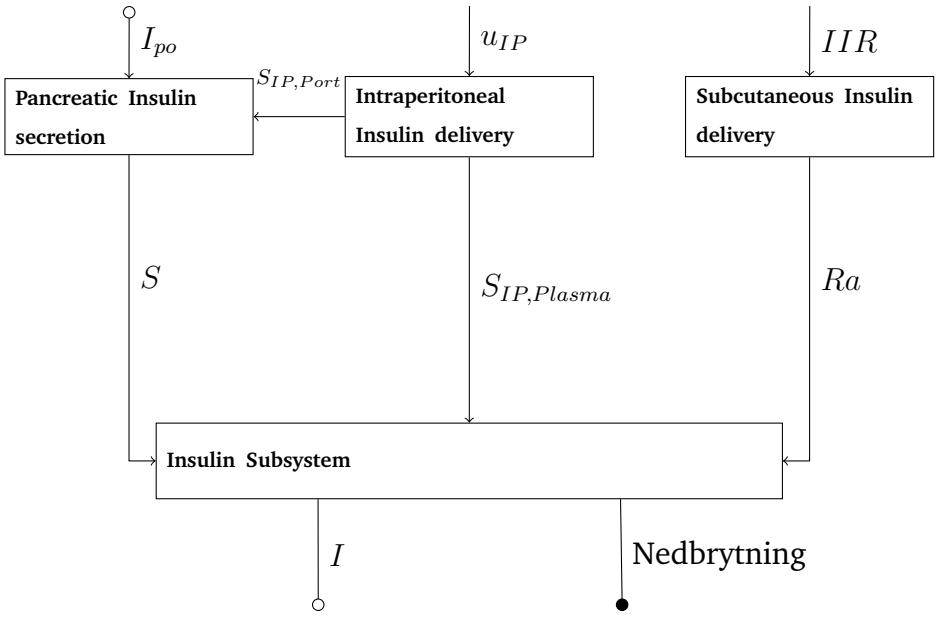


Figure 22: Ny metabolsk modell: Insulinsystem

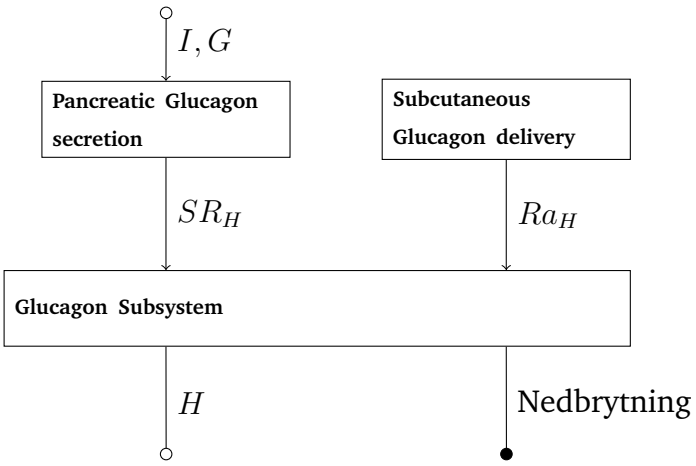


Figure 23: Ny metabolsk modell: Glukagonsystem

Appendix C MATLAB-kode

All MATLAB-kode som er tatt i bruk er vedlagt i egen *.zip*-fil. Dette gjelder all tidligere kode som har vært implementert av Erlbeck, [9], og Unstad, [25], samt nye filer som jeg har implementert.

Appendix D Singular Value Decomposition

Her er flere utdrag fra sensitivitetsanalysen, og dens dekomposisjon av singulærverdier, *SVD*. Figurene viser samme dekomponering som i Seksjon 7.1.1. Figur 28 representerer *SV* og er lik for hele sensitivitetsanalysen. Denne vil da kun være vist en gang. Videre er det visualisert 4 figurer med henholdsvis *RSV* for singulærverdi nr. 2, 3, 4, 5.

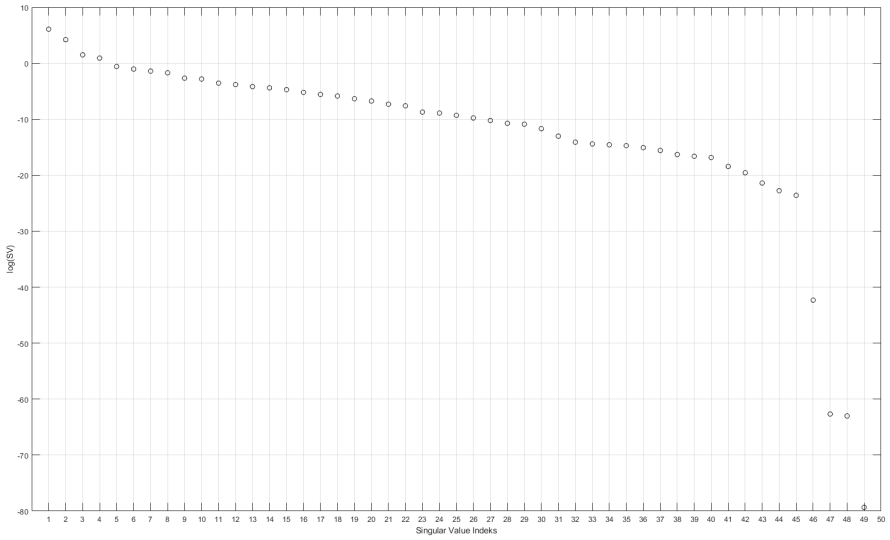


Figure 24: Singulærverdier for sensitivitetsmatrisen, S_y

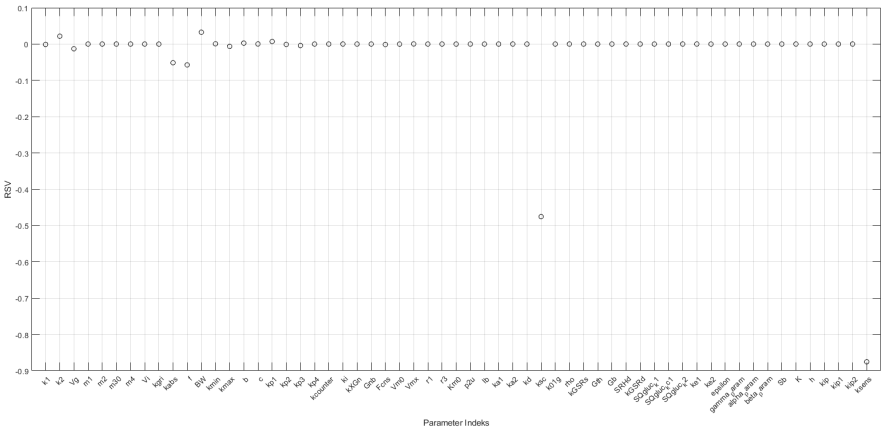


Figure 25: *RSV* for *SV* nr. 2

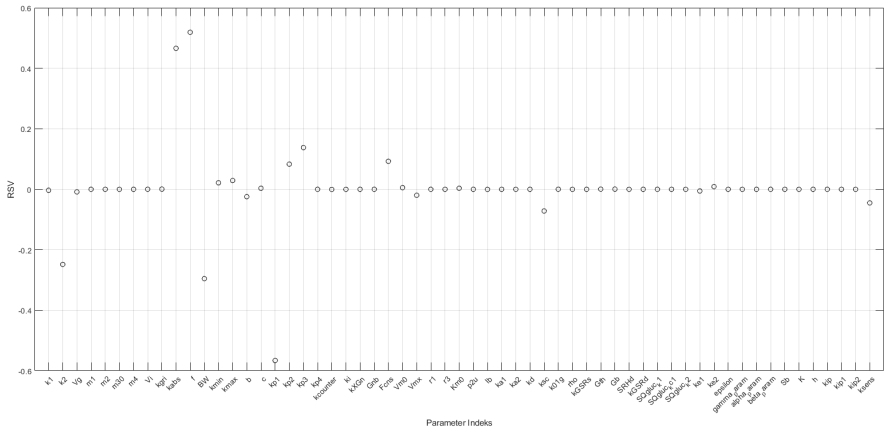


Figure 26: *RSV* for *SV* nr. 3

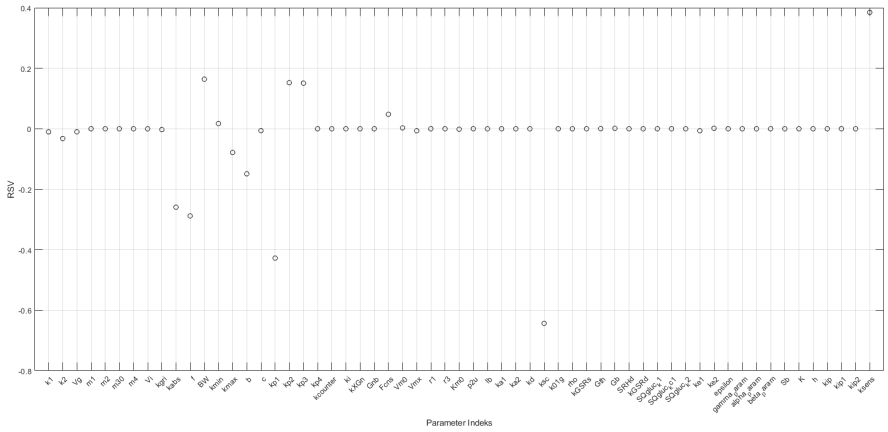


Figure 27: *RSV* for *SV* nr. 4

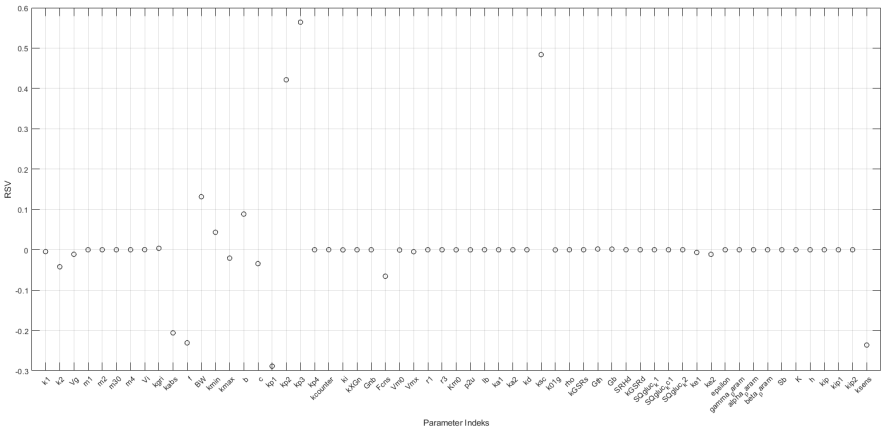
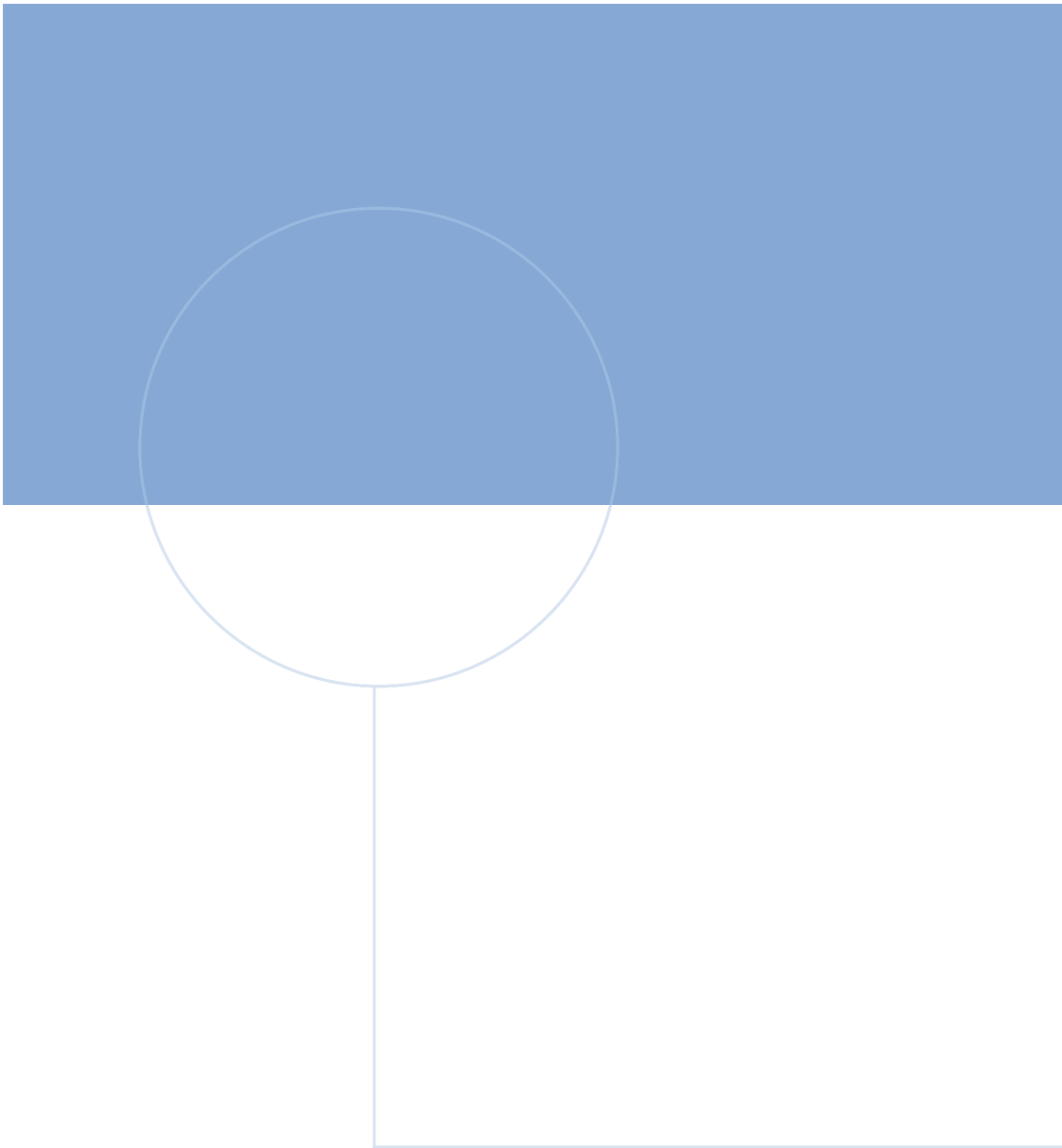


Figure 28: *RSV* for *SV* nr. 5



NTNU
Kunnskap for en bedre verden

apt
ARTIFICIAL PANCREAS TRONDHEIM