# Optimal dykkeadfærd for Storøjet tun

Lene Sommer


Kongens Lyngby 2013
IMM-B.Sc-2013-2

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk IMM-B.Sc-2013-2

# Abstract

The Bigeye tuna, with the Latin name Thunnus obesus, lives in the Indian Ocean, Atlantic Ocean and Pacific Ocean. During daytime the tuna has a distinct diving pattern where it spends most of its time at deep waters hunting prey but makes regular trips to more shallow waters. The objective of this report is to investigate the tuna's vertical behaviour. The goal of the investigation is to see, whether this vertical behaviour is the result of optimal foraging theory?

The method for solving this is to formulate a dynamic optimisation problem for the energy harvest during one day. Dynamic programming is used to solve the optimisation problem and it leads to a Hamilton-Jacobi-Bellman equation. The Hamilton-Jacobi-Bellman equation is then solved numerically. From the numeric solution it is possible to simulate the tuna's vertical behaviour. Furthermore from the numeric solution it is investigated whether there is a threshold in two of the parameters which leads to a change in the tuna's behaviour.

There are two main results from simulating the tuna's diving pattern. First and foremost it can be concluded that optimal foraging theory is the reason for the observed noticeable vertical behaviour. Next it is shown that when varying the two parameters the tuna change its behaviour from staying at the same depth, to diving between shallow waters and deep waters.

# Resume

Storøjet tun, med det latinske navn Thunnus obesus, lever i Atlanterhavet, Stillehavet samt Det Indiske Ocean. Tunen skiller sig ud ved at have et meget særpræget dykkemønster i løbet af dagtimerne. Den ligger primært på dybt vand, hvor byttedyrene lever, men udfører hyppige ture mod lavere dybder. Formålet med denne opgave er at undersøge tunens dykkemønster for at kunne konkludere, hvorvidt det er et resultat af optimal fourageringsteori.

Metoden til at løse denne opgave er at formulere et dynamisk optimeringsproblem over energihøsten i løbet af en dag. Optimeringsproblemet sættes på Hamilton-Jacobi-Bellman form ved at benytte dynamisk programmering og en numerisk løsning til denne findes. Ud fra den numeriske løsning er det muligt at simulere tunens dykkeadfærd. Ydermere undersøges det ud fra den numeriske løsning, hvorvidt der findes en grænse for to af parametrene, således at tunens optimale adfærd ændrer karakter.

To hovedresultater findes ud fra simuleringen af løsningen: Først og fremmest kan det konkluderes, at optimal fourageringsteori er den bagvedliggende årsag til tunens særprægede dykkemønster. Dernæst er det vist, at når de to udvalgte parametre varieres ændres tunens dykkemønster således at tunen går fra at være ved konstant dybde til at dykke mellem dybt og lavt vand.

# Forord

Denne opgave er skrevet på Institut for Akvatiske Ressourcer på Danmark Tekniske Universitet som afslutning på bacheloruddannelsen Matematik og Teknologi under vejledning af seniorforsker Uffe Høgsbro Thygesen fra Sektion for Havøkologi under Institut for Akvatiske Ressourcer.

Opgaven omhandler hvorledes en storøjet tun skal svømme for at høste størst mulig energi. Dette er, såvidt vides, ikke tidligere undersøgt.

En stor tak til Toby Patterson fra Australiens Commonwealth Scientific and Research Organisation (CSIRO) Hobart, Tasmanien, Australien. for at være behjælpelig med data samt min vejleder for et spændende projekt og gode diskussioner. Til sidst en tak til Henrik Sommer og Jacob Schack Vestergaard for at læse min rapport igennem og for at lytte og snakke med om storøjede tunfisk igennem hele projektet.

Lyngby, 1. februar 2013

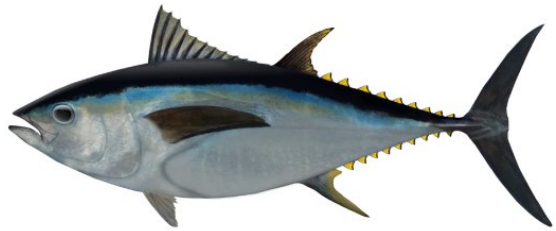
Lene Sommer

# Indhold

Kapitel 1

# Introduktion

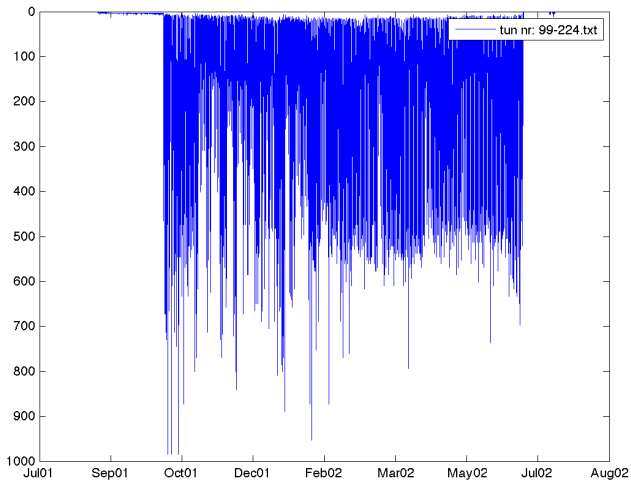
Storøjet tun, med det latinske navn Thunnus obesus, lever i Atlanterhavet, Stillehavet samt Det Indiske Ocean. Individer af arten kan blive helt op til 2.36 m lange og veje 210 kg. Føden, der indtages, spænder vidt fra fisk til blæksprutter og krebs. Aldersmæssigt kan individer blive op til 12 år gamle [17, 22].



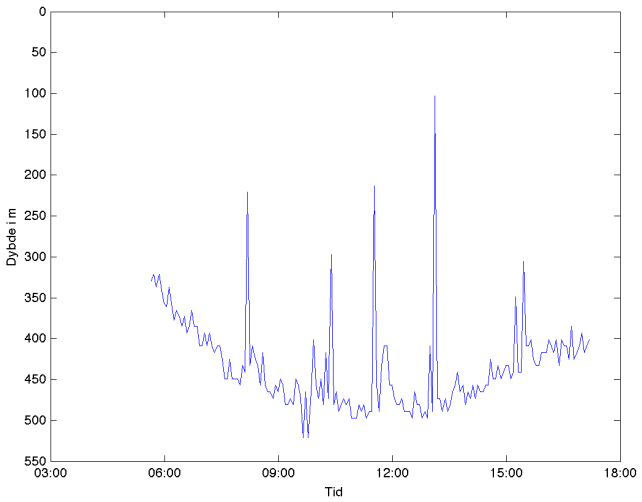
**Figur 1.1:** En Storøjet tun [7].

Den storøjede tun skiller sig ud ved at have et meget særpræget dykkemønster i løbet af et døgn. På Figur 1.2(a) er vist dybdedata opsamlet fra en Storøjet tun over cirka ni måneder.

Figur 1.2(b) viser, at den storøjede tun om dagen primært er på dybt vand hvor den jager føde, men udfører hyppige ture mod lavere dybder. Om natten ligger tunen i overfladen [3, s. 1]. Forklaringen på denne adfærd er, at der på de




(a) Dykkemønstret for tunen over mange måneder.




(b) Adfærd i løbet af en dag.

**Figur 1.2:** Figur 1.2(a) viser dybdemålinger fra en Storøjet tun med GPS nummeret: 99 – 224. Data er fra Toby Patterson fra Australiens Commonwealth Scientific and Research Organisation (CSIRO) Hobart, Tasmanien, Australien. [3]. På Figur 1.2(b) ses tunens særprægede dykkemønster i løbet af dagtimerne.

store dybder er føde, men for koldt til, at tunen kan opholde sig der hele tiden. Dermed er tunen nødsaget til at søge op mod overfladen engang imellem for at blive varmet op. Her er dog ingen føde, så når den er blevet varm nok, er det nødvendigt igen at søge mod dybet og føden [3, s. 9].

I denne opgave vil det blive undersøgt, hvad der er tunens optimale dykkestrategi i løbet af en dag, med henblik på at den skal høste mest mulig energi. Tunen høster energi når den spiser byttedyr. Det er her interessant, at se hvorvidt den optimale strategi resulterer i det samme karakteristiske dykkemønster, som data viser. Hvis det er tilfældet, vil paradigmet at dyrenes adfærd styres af, at opnå størst mulig energihøst blive bekræftet. Paradigmet kaldes også optimal fourageringsteori [19] og bygger på principper, som stammer helt tilbage fra Darwins udgivelse af "Arternes oprindelse"for mere end 150 år siden. Formålet med opgaven er således at kvantificere optimal fouragerinsteori. Forskellige parametres indvirkning på dykkemønstret vil ligeledes blive undersøgt, her iblandt hvorvidt størrelsen af tunen påvirker dykkemønstret.

Viden om tunens dykkemønster og muligheden for at opstille en model for det er yderligere interessant for eksempelvis havbiologer, som har mulighed for at undersøge, hvorledes et ændret miljø kan påvirke tunens adfærd. Det kan således undersøges i hvilken grad global opvarmning vil påvirke tunen dykkemønster og dermed, bl.a. dens fangst af blæksprutter, fisk og krebs.

Det er ikke tidligere undersøgt, om tunens dykkemønster er et resultat af optimal fourageringsteori. Der er dog lavet en del statistiske undersøgelser ud fra data omkring hvordan tunen bevæger sig samt opstillet modeller, der beskriver dykkemønstret, eksempelvis [3] og [15]. Selve optimeringsproblemet er kendt fra f.eks. pattedyr der lever i vandet, eksempelvis hvaler, og de relaterede iltbegrænsninger [13].

## 1.1 Problemformulering

Helt konkret ønskes det i opgaven at finde svar på følgende spørgsmål:

- Kan tunens dykkemønster beskrives ved optimal fourageringsteori?
- Findes der en grænse for en af parameterne således, at når parameteren varieres, ændres tunens adfærd fra konstant dybde til variabel dybde?

## 1.2 Overblik over opgaven

Et overblik over opgaven angives her i punktform:

- Kapitel 1: En Storøjet tun og dens dykkemønster er kort beskrevet sammen med problemformuleringen og tilhørende motivation.
- Kapitel 2: Gennemgang af tunens miljø samt en model for hvis tunen holder konstant dybde.
- Kapitel 3: Modellen for tunens adfærd ændres til at tage højde for at tunen kan variere sin dybde på to forskellige måder. Det er en indledende øvelse inden selve analysen gennemgåes i Kapitel 4.
- Kapitel 4: Tunens adfærd behandles som et dynamisk optimeringsproblem, og en numerisk løsning hertil findes.
- Kapitel 5: Det undersøges hvad der sker når to forskellige parametre i modellen varieres.
- Kapitel 6: Beslutninger og overvejelser gjort gennem opgaven diskuteres inden en konklusion i Kapitel 7.
- Appendiks A: De numeriske størrelser for parametrene er angivet.
- Appendiks B: Baggrundsviden til maksimeringsmetoden og startbetingelser til løsningsmetoden brugt i Kapitel 3
- Appendiks C: Flere plots af tunens dykkemønster fra data.
- Appendiks D: Al MATLAB koden til projektet.

Kapitel 2

# Konstant dybde

Som beskrevet i introduktionen er formålet med opgaven at finde tunens maksimale energihøst i løbet af en dag. Det følgende afsnit beskriver, hvilken dybde tunen ville opholde sig ved, hvis den havde konstant dybde og skulle maksimere sin energihøst. Det er interessant at undersøge med henblik på senere at kunne sammenligne med det optimale udtryk for en tun, som udnytter at kunne variere dybden.

## 2.1 Tunens miljø

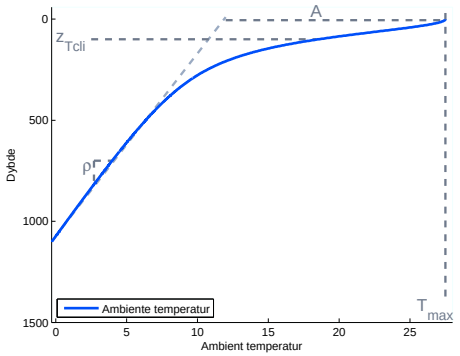
Når tunen holder konstant dybde, er den maksimale energihøst på den dybde, hvor der er føde nok, men samtidigt også varmt nok vand. Vandets temperatur og fødetætheden defineres først. I Afsnit 2.2 defineres energihøsten.

Et udtryk for vandets temperatur i grader celcius er givet ved Formel (2.1). Vandets temperatur kaldes den ambiente temperatur for at skelne mellem tunens indre temperatur og det vand, som omgiver tunen. Det er valgt at modellere den ambiente temperatur $T_a(z)$ som:

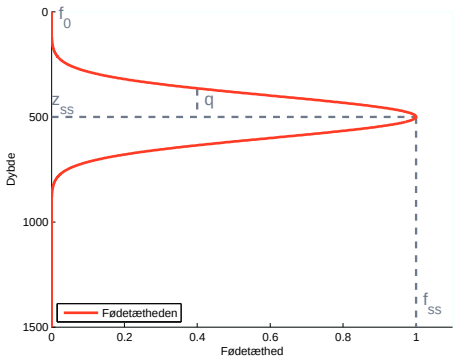
$$T_a(z) = T_{max} + (\rho \cdot z - A) \cdot \frac{z^\lambda}{(z^\lambda_{T_{cli}}) + z^\lambda} \tag{2.1}$$

hvor $z$ er dybden i meter, $T_{max}$ er overfladetemperaturen, $A$ er temperaturfaldet over termoklinen, $\rho$ er temperaturfaldet med dybden efter termoklinen, $\lambda$ er hvor brat overgangen er fra termoklinen til det lineære temperatur fald og $z_{Tcli}$ er dybden hvor termoklinen slutter. Figur 2.1(a) viser et plot af den ambiente temperatur som funktion af dybden, hvor parametrene er indtegnet. For illustrationens skyld er akserne bytte om. For værdier for parametrene se Appendiks A.

Det er en generel antagelse, at den ambiente temperatur er uafhængig af tiden på døgnet [3, s. 10] samt at den ambiente temperatur påvirker tunen [3, s. 8].




(a) Den ambiente temperatur.




(b) Fødetætheden.

**Figur 2.1:** Plot af den ambiente temperatur og fødetætheden som funktion af dybden. De forskellige parametre der indgår i Ligning (2.1) og Ligning (2.2) er indtegnet. For illustrationens skyld er akserne byttet om.

Det er valgt at definere fødetætheden som:

$$f(z) = f_0 + f_{ss} \cdot e^{-\frac{1}{2}\left(\frac{(z-z_{ss})^2}{q^2}\right)} \tag{2.2}$$

Fødetætheden er, ligesom den ambiente temperatur, en funktion af dybden $z$, $f_0$ er fødetætheden ved overfladense, $f_{ss}$ er den maksimale fødetæthed, $z_{ss}$ er dybden hvor der er maksimal fødetæthed og $q$ er bredden af det halve fødefelt. På Figur 2.1(b) er parametrene indtegnet. For værdier for parametrene se Appendiks A. Fødetætheden har form som en Gauss fordeling og stiger i takt med at dybden stiger indtil $z_{ss}$, hvorefter fødetætheden igen falder.

## 2.2 Tunens fødesøgning

Tunens svømmehastighed $U$ målt i $\frac{m}{s}$ er en funktion af tunens krops temperaturen $T$. Det er en rimelig antagelse, da varme påvirker musklernes ydeevne [4]. Varme muskler øger evnen til at svømme hurtigt, og omvendt er svømmehastigheden lav, når musklerne er kolde.

Den præcise afhængighed mellem svømmehastighed og tunens temperatur er svær at bestemme, men baseret på data fra [14], vælges det at opstilles den simple model i Ligning (2.3), hvor svømmehastighed varierer lineært i forhold til temperatur. Hvis tunen ligger ved konstant dybde, kan det yderligere antages, at tunen er i termisk ligevægt med vandet, det vil sige $T = T_a$:

$$U(T) = 0.053T + 0.481 \Rightarrow \tag{2.3}$$

$$U(T_a) = 0.053T_a + 0.481 \tag{2.4}$$

Størrelsen på hældningen og skæringspunktet er beregnet ud fra lineær regression for to punkter, se Figur A.1. Punkterne er givet ved én tuns maksimale og minimale temperatur og svømmehastighed fra data [14, s. 5 og s. 6]. Det antages, at tunen svømmer ved maksimal hastighed, når den svømmer.

Ved at gå ud fra Figur 2.2, som illustrerer tunens synsfelt under jagt i en forsimplet 2D udgave, kan udtrykket for energihøsten i Ligning (2.11) udledes. $A$ er arealet af tværsnittet for tunens synsfelt. $B$ er bredden på tunens synsfelt. $L$ er længden af synsfeltet givet ved tunens hastighed multipliceret med tiden. $S_p$ angiver antallet af byttedyr og findes som arealet multipliceret med den

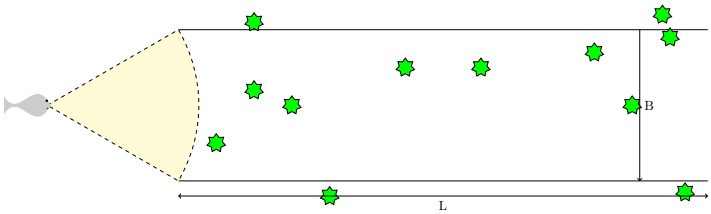
pågældende fødetæthed.

$$A = L \cdot B \tag{2.5}$$
$$L = U(T) \cdot t \tag{2.6}$$
$$A = U(T) \cdot t \cdot B \tag{2.7}$$
$$S_p = f(z) \cdot U(T) \cdot t \cdot B \tag{2.8}$$
$$E = \frac{S_p}{t} = f(z) \cdot U(T) \cdot B \tag{2.9}$$
$$\tag{2.10}$$

Hermed fås energihøsten udtrykt ved svømmehastigheden og fødetætheden for en tunen som holder konstant dybde:

$$E(z) = f(z) \cdot U(T(z)) \Rightarrow \tag{2.11}$$
$$E(z) = f(z) \cdot U(T_a(z)) \tag{2.12}$$

Bredden på tunens synsfelt $B$ er udeladt ud fra den antagelse, at for én enkel tun er bredden konstant. Tunen vil uanset bredden af synsfeltet forsøge at fange flest mulige byttedyr, det vil sige søge mod områder med stor fødetæthed.




**Figur 2.2:** Illustration af tunens synsfelt under jagt i en forsimplet 2D udgave. B er bredden på tunens synsfelt og de grønne stjerner er byttedyr (blæksprutter).

Fødetætheden er givet ved antal blæksprutter pr. $m^3$, der er i tværsnittet af tunens synsfelt [1, s. 28], se Figur 2.2. Produktet af densiteten af blæksprutter og tilført energi per blæksprutte giver enheden for fødetætheden:

$$\frac{\text{blæksprutte}}{m^3} \cdot \frac{energi}{\text{blæksprutte}} \cdot m^2 = \frac{energi}{m} \tag{2.13}$$

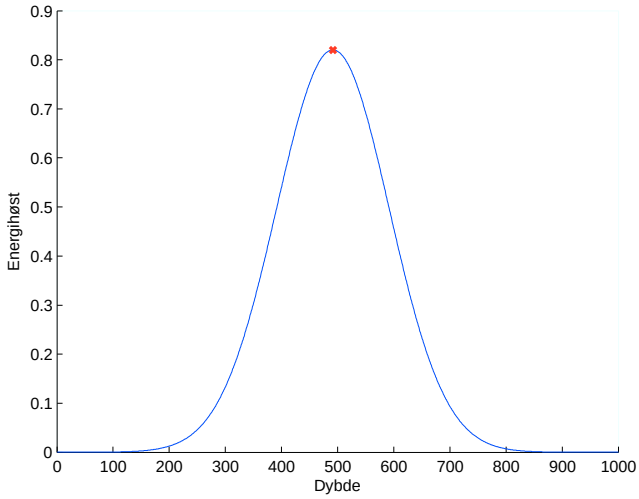
Den maksimale fødetæthed kan sættes til 1, dvs. $f_0 = 0$ og $f_{ss} = 1$, og indgå i Ligning (2.11) som dimensionløs, fordi der ikke er inkluderet nogle udgifter i Ligning (2.11). Udgifter er energitab for tunen eksempelvis i form af varmetab.

## 2.3 Maksimum

Når tunen holder konstant dybde, kan en maksimal energihøst findes ved at udregne energihøsten for forskellige dybder og finde den største værdi. For de givne parametre i Appendiks A bestemmes den maksimale energihøst og den dertilhørende dybde til:

$$E = 0.82, \quad z = 491.5\ m \tag{2.14}$$

Når den maksimale energihøst er fundet er det oplagt at undersøge, om det




**Figur 2.3:** Energihøsten som funktion af dybden. Det røde kryds er en makering af den maksimale dybde. Det kan konkluderes at der kun er ét maksimum.

fundne maksimum, er det globale maksimum. Ud fra Figur 2.3 kan det konkluderes, at der kun er et maksimum. Konkavitet for udtrykket kunne også være undersøgt. Hvis det gælder, at udtrykket er strikt konkavt, vil der altid være et maksimum, og det maksimum vil være det globale maksimum [10, Afsnit 12.2].

## 2.4 Opsummering

Dette kapitel viser, hvorledes tunens energihøst kan beskrives, hvis den opholder sig ved konstant dybde, se Ligning (2.12). Yderligere er det beskrevet, at

et fundent maksimum vil være et globalt maksimum, når den definerede temperaturfunktion og fødetæthed gælder. I det følgende kapitel vil et udtryk for tunens energihøst, hvor tunen varierer sin dybde, blive beskrevet.

Kapitel 3

# Variabel dybde

I det følgende opstilles et udtryk for den gennemsnitlige energihøst med variabel dybde. Den variable dybde gør det muligt for tunen at jage nede i dybden og blive varmet op i overfladen [3, s.1].

## 3.1 Model

For at modellere den variable dybde opstilles tre koblede differentialligninger; en for tunens temperaturen $T$, en for dybden $D$ og en for energihøsten $E$. Alle med løsninger som funktion af tiden $t$:

$$\frac{dT}{dt}(t) = k(T(t) - T_a(D(t))) \cdot (T_a(D(t)) - T(t)) \tag{3.1}$$

$$\frac{dD}{dt}(t) = U(T(t)) \cdot \sin(\Phi(t)) \tag{3.2}$$

$$\frac{dE}{dt}(t) = U(T(t)) \cdot f(D(t)) \tag{3.3}$$

Tunens dybde modelleres ikke længere som konstant, og der er derfor ikke mere ligevægt mellem tunens temperatur $T$ og den ambiente temperatur $T_a$. Svømmehastigheden er således afhængig af tunens temperatur $T$.

$k$ er ikke en konstant, men et udtryk for varmeledningen:

$$k(T(t) - T_a(D(t))) = k_{low} + (k_{hign} - k_{low}) \frac{1}{1 + e^{\frac{T(t) - T_a(D(t)) + T_{bb}}{\triangle T}}} \tag{3.4}$$

Tunen kan justere, hvor hurtigt den afgiver og optager varme fra det omgivne vand, alt afhængigt af om den svømmer i koldt eller varmt vand, angivet ved $k_{low}$ og $k_{high}$ [14, s. 4-5]. $T_{bb}$ er et udtryk for at tunens temperatur er lavere end omgivelserne, inden der sker et skift fra $k_{low}$ til $k_{high}$. Med inspiration fra [14, s. 2] vælges det specifikt at sætte udtrykket for varmeledningen på formen i Ligning (3.4). For de numeriske værdier for parametrene se Appendiks A.1.

Ligningerne (3.1)-(3.3) er valgt ud fra følgende antagelser: I Ligning (3.1) er det antaget, at tunens temperatur kan beskrives ved kun én kropstemperatur. I Ligning (3.2) antages det, at tunen svømmer maksimal hastighed, når den svømmer og at der i øvirgt ikke er nogle omkostninger i forbindelse med det, se Ligning (3.3).

$\Phi$ repræsenterer vinklen på tunens svømmeretning. To specifikke former af $\Phi$ undersøges i det følgende inden den optimale $\Phi$ findes i Kapitel 4. $\Phi$ antager én form med sinussvingninger og én form udtrykt ved såkaldte logiske svingninger, se Afsnit 3.3 og Afsnit 3.4.

De tre koblede differentialligninger løses numerisk med `ode45`, som er standard løsningsmetoden i MATLAB. `ode45` kan anvendes på gængse differentialligninger, når blot de er ikke-stive [8, s.270], hvilket forventes at være tilfældet her. A posteriori ses det at være korrekt, og en løsning til de tre differentialligninger findes. `ode45` er baseret på eksplicit Runge-Kutta metoden [8, s. 270], hvor 4 og 5 står for henholdsvis fjerde og femte ordens Runge-Kutta [16].

## 3.2 Gennemsnitlig energihøst

Det er nødvendigt at kunne beregne en gennemsnitlig energihøst $E_{genm}$ ud fra løsningen af de tre differentialligninger, se Ligning (3.5). En vektor **e** indeholder den akkumulerede energihøst og hvert element $e_t$ er den akkumulerede energihøst efter tiden $t$.

$$E_{genm} = \frac{e_{t+\triangle t} - e_t}{\triangle t} \tag{3.5}$$

Her skal tidsforskellen $\triangle t$ være et helt antal perioder $n$:

$$\triangle t = n \cdot \frac{2\pi}{\omega} \tag{3.6}$$

For at finde det højeste antal perioder, udregnes $n$ til den største værdi af vektoren $\mathbf{t}$, som indeholder alle tiderne hvortil $\mathbf{e}$ er beregnet. $t_1$ er startpunktet.

$$t_1 + n \cdot \frac{2\pi}{\omega} < \max(\mathbf{t}) \tag{3.7}$$

$$n < \frac{\max(\mathbf{t}) - t_1}{2\pi} \cdot \omega \Rightarrow \tag{3.8}$$

$$n = \lfloor (\frac{\max(\mathbf{t}) - t_1}{2\pi} \cdot \omega) \rfloor \tag{3.9}$$

## 3.3 Sinussvingninger

For sinussvingningerne er $\Phi$ givet ved:

$$\Phi(t) = \epsilon \cdot \sin(\omega \cdot t) \tag{3.10}$$

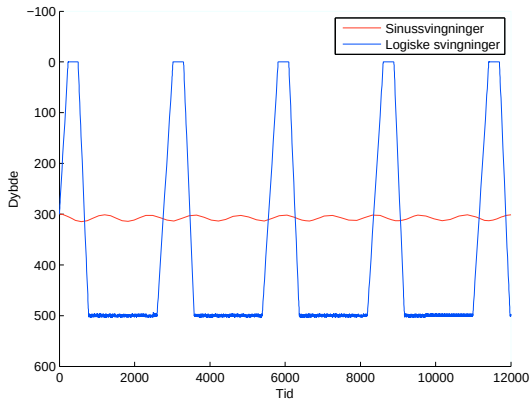
hvor $\omega$ er vinkelfrekvensen og $\epsilon$ amplituden på svingningerne, se evt. Appendiks A.2. Initialbetingelserne sættes til at være den optimale konstante dybde fundet i Kapitel 2.

Nu kan energihøsten bestemmes ud fra Ligning (3.5) og sammen med en simulering af løsning vises den på Figur 3.1. På Figur 3.1(a) ses de karakteristiske sinussvingninger, mens Figur 3.1(b) viser hvordan tunens temperatur støt falder indtil den finder et fast leje, fordi tunen ligger og svømmer omkring den samme dybde.

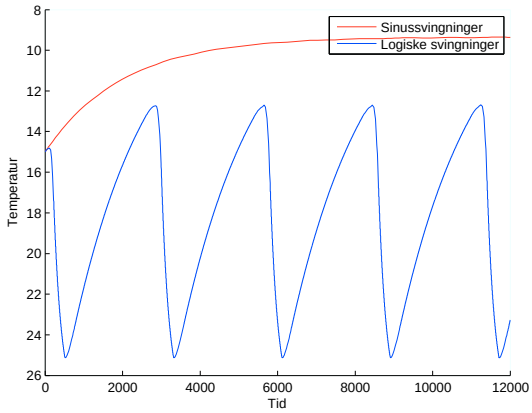
Når en gennemsnitlig energihøst kendes, se Figur 3.1(c), er det muligt at maksimere energihøsten ved at optimere på parametrene i udtrykkene for $\Phi$. Når $\Phi$ antager formen i Formel 3.10, kan den gennemsnitlige energihøst maksimeres ved at variere vinkel frekvenksen $\omega$ og størrelsen af dybdeforandringen $\epsilon$. Ved at sammenligne Figur 3.1(a) med observationerne fra data i Figur 1.2(b), kan det konkluderes, at sinussvingningerne ikke er konsistente med data. Maksimering af sinussvingningerne udforskes derfor ikke videre. Istedet undersøges det, hvad der sker, når $\Phi$ er givet ved de logiske svingninger.

## 3.4 Logiske svingninger

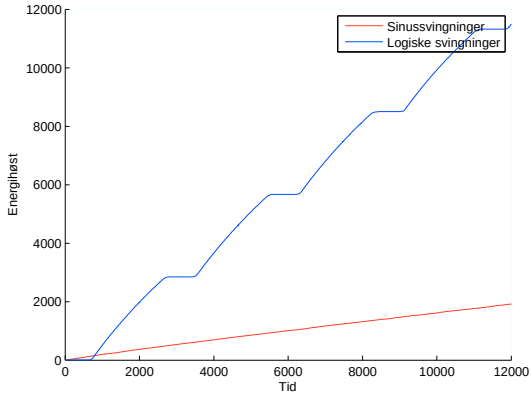
De logiske svingninger kan modelleres ved at lave en grænse i dybden, som opdeler tunens svømmeområde i to dele, se Figur 3.2. Alt afhængigt af hvilket område tunen er i, skal den svømme op eller ned. Udtrykket for $\Phi$ er er givet




(a) Dybde.




(b) Temperatur.



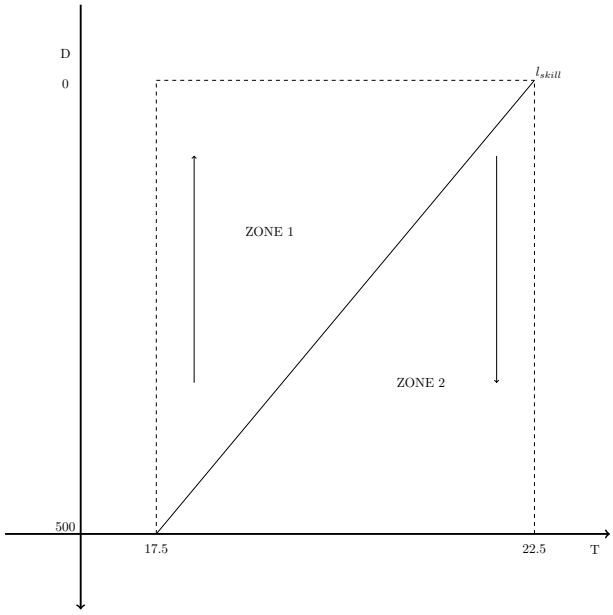

(c) Akkumuleret energi.

**Figur 3.1:** En simulering af løsningen til de koblede differentialligninger: Ligning (3.1) - (3.3). Φ er modelleret som sinussvingninger eller logiske svingninger.

i Tabel 3.1 ud fra tunens dybde $D(t)$ til tiden $t$. Som et eksempel kan nævnes, at hvis tunen er kold og i dybden skal den svømme op. Når den gennemsnitlige

| $\Phi$ | område | reaktion |
|---|---|---|
| $\Phi(t) = -\tan^{-1}(\mathrm{D(t)})$ | ZONE1 | Tunen svømmer mod dybden 0. |
| $\Phi(t) = \tan^{-1}(z_{ss}$ - D(t)) | ZONE2 | Tunen svømmer mod dybden $z_{ss}$. |

**Tabel 3.1:** Udtryk for $\Phi$ ved logiske svingninger. ZONE1 og ZONE2 er de to dele som tunens svømmeområde er delt op i. Alt afhængigt af hvilket område tunen er i skal den svømme op eller ned.




**Figur 3.2:** $l_{skill}$ er grænsen mellem hvornår tunen skal svømme op og ned når dens bevægelse modelleres med logiske svingninger. Opdelingen mellem områderne er givet ved følgende, hvor $T$ er tunenes temperatur til tiden t:
$l_{skill}(T(t)) > D(t) \Rightarrow$ ZONE1, $l_{skill}(T(t)) \leq D(t) \Rightarrow$ ZONE2.

energihøst skal bestemmes, anvendes igen Ligning (3.5), men modsat sinussvingningerne kendes værdien for $\omega$ ikke på forhånd. Værdien for $\omega$ må bestemmes ud fra simulering af løsningen. Ud fra simuleringen på Figur 3.1(a) findes perioden $T_p$ og dermed $\omega$:

$$T_p = \frac{2 \cdot \pi}{\omega} \Leftrightarrow \omega = \frac{2 \cdot \pi}{T_p} \tag{3.11}$$

Den gennemsnitlige energihøst kan nu maksimeres. Maksimeringen sker ved at optimere på de to parametre for den linje der adskiller ZONE1 og ZONE2 og som afgør om tunen svømmer op eller ned. Parametrene er: tunens temperatur ved overfladen $T_{high}$ og tunens temperatur i den dybde hvor der er mest føde $T_{low}$. Til at maksimere dette bruges `fminsearch`, se Appendiks B.2 for yderligere information om `fminsearch` og Appendiks B.1 for initialbetingelser. Figur 3.1(c) viser den akkumulerede energihøst med de optimerede parametre.

Ved gennemgangen af sinussvingningerne blev det bemærket, at tunens temperatur faldt støt, men ved de logiske svingninger gælder det omvendt, at temperaturen varierer med dybden og tunens adfærd. Ved at sammenligne de logiske svingninger på Figur 3.1(a) og Figur 3.1(b) ses det, at når tunens minimums temperaturen ligger omkring 12°C.

Fra Kapitel 1 erindres det igen, hvorledes tunens observerede dykkemønster ser ud, se Figur 1.2(b). På Figur 3.1(a) kan det ses, at opførselen for de logiske svingninger er sammenlignelig med det observerede dykkemønster.

## 3.5 Opsummering

Et koblet differentialligningssystem, som kan beskrive tunens varierende dybde, er i det forrige blevet opstillet, en løsning er simuleret og energihøsten er optimeret. Det kan konkluderes, at når $\Phi$ antager logiske svingninger, er tunens adfærd sammenlignelig med det observerede dykkemønster fra data. Strategien i dette kapitel har været først at finde et dykkemønster, der ligner det observerede, og derefter at optimere energihøsten. I det følgende kapitel vil den omvendte strategi blive benyttet: Her vil tunens optimale energihøst findes, og det skal derefter undersøges hvorvidt den optimale energihøst resulterer i det observerede dykkemønster fra data.

Kapitel 4

# Dynamisk udtryk for tunens adfærd

Det ønskes nu at se på tunens adfærd som et dynamisk udtryk. Med dynamisk henvises ikke til tunens adfærd, men det at den dynamiske optimering maksimerer over et funktionsrum. Det er modsat det forrige kapitel, hvor der var tale om et statisk problem, men stadigvæk en dynamisk adfærd. Med statisk menes at optimeringen maksimerer over en del af $\mathbb{R}^n$ [21, s. 14].

Tunens adfærd kan modelleres ved at antage, at den har nogle mulige handlinger. Adfærden i løbet af en dag er modelleret som en sekvens af særskilte handlinger til tiden $t$ [11, s. 12-13]. Det er en deterministisk model, således at til en given tid vil tunen udføre én handling [11, s. 26].

Tilstandsvariable beskriver tunens tilstand til en given tid og i denne opgave er temperaturen $T$ og dybden $D$ valgt som tilstandsvariable. Tilstandsvariablene skal være faktorer, der har signifikant indflydelse på tunens mulighed for at høste energi. Som eksempel ville tunens vægt ikke være aktuel at inkludere, da vægtændringen over et døgn er minimal. Havde det derimod været et ønske at se på energihøsten over et år, så kunne vægten være interessant at inkludere [11, s.10-11].

Funktionen $V(D, T, t)$ beskriver tunens "fitness"i dybden D med temperaturen T og til tiden t. Med fitness menes tunens mulige energihøst inden dagen er

omme. Fitness begrebet kommer fra Darwins "survival of the fittest". Inden for biologien refererer fitness til antallet af unger som det er muligt at få. Det kan dog antages, at antallet af unger hænger sammen med, hvor stor en energireserve tunen har, og dermed er energihøsten en proxy for fitness [1, s. 27].

Hver tilstandsvariabel er underlagt en af de følgende differentialligninger, der også er givet i Kapitel 3:

$$\frac{dT}{dt}(t) = k(T(t) - T_a(D(t))) \cdot (T_a(D(t) - T(t))) \tag{4.1}$$

$$\frac{dD}{dt}(t) = U(t) \cdot \sin(\Phi(t)) \tag{4.2}$$

$$\frac{dE}{dt}(t) = f(D(t)) \cdot U(T(t)) \tag{4.3}$$

For at finde tunens størst mulige energihøst i løbet af en dag skal følgende maksimeringsproblem løses:

$$\max_{\phi} \{E(T_s) - E(0)\} \tag{4.4}$$

Dette skal løses over et tidsspænd på én dag, hvor $T_s$ er sluttidspunktet for dagen. Det gælder at beslutningsvariablen er $\Phi \in \mathbb{R},\ \Phi \in [0, T_s]$. Hvor $\Phi$ er en stykkevis kontinuert funktion af tiden, således at (4.1) - (4.3) har en entydig løsning. Optimeringsproblemet i Ligning (4.4) har begrænsninger givet ved initialbetingelserne samt de tre systemligninger. Samlet kan det opskrives som:

$$\max_{\phi} \{E(T_s) - E(0)\} \tag{4.5}$$

$$st. : T(0),\ D(0),\ E(0) \tag{4.6}$$

$$\frac{dT}{dt}(t) = k(T(t) - T_a(D(t)))(T_a(D(t) - T(t))) \tag{4.7}$$

$$\frac{dD}{dt}(t) = U(t) \cdot \sin(\Phi(t)) \tag{4.8}$$

$$\frac{dE}{dt}(t) = f(D(t)) \cdot U(T(t)) \tag{4.9}$$

Nu haves optimeringsproblemet og i det følgende skal en løsningsmetode beskrives.

## 4.1 Dynamisk programmering

For at løse det dynamiske udtryk for tunens adfærd anvendes dynamisk programmering. Først er det dog nødvendigt at se på The Principle of Optimality.

Princippet siger, at en optimal løsning kan findes ved at løse problemet stykkevis. Først kan en optimal løsning findes for det stykke, som indeholder slutpunktet og herefter udvides til de to sidste stykker. Sådan fortsættes indtil en optimal løsning for hele problemet er fundet [2, s. 18-19].

En god sammenligning er en køretur fra København til Århus. Antag at den hurtigste rute går over Odense. Det betyder, at den del af ruten, som går fra Odense til Århus, også ville være den hurtigste rute for en tur, som startede i Odense og endte i Århus [2, s. 18].

Dynamisk programmering bygger på The Principle of Optimality. For at dynamisk programmering kan anvendes, er det dog vigtigt at slå en af de vigtige egenskaber ved dynamiske systemer fast, nemlig kausalitet. For dynamiske systemer gælder kausalitet, dvs. at en beslutning ikke påvirker den forrige tilstand, kun den aktuelle og de fremtidige tilstande [18, s. 55].

En skitse til et bevis for vores system vises i det følgende. Metoden er beskrevet i [2, s. 101-102]. Tunens fitness til en given tid $t$, dybde $D$ og temperatur $T$ kan ud fra Formel (4.4) opskrives som det maksimale integral over energihøsten:

$$V(D(t),T(t),t) = \max_{\phi} \left\{ \int_t^{T_s} f(D(s)) \cdot U(T(s)) ds \right\} \tag{4.10}$$

hvor $T_s$ er når dagen er slut. For Ligning (4.10) gælder begrænsningerne givet i (4.6) - (4.9). Det antages at $f(D(s)) \cdot U(T(s))$ er integrabelt [2, s. 102].

Nu skal det vises, at udtrykket for tunens fitness opfylder en Hamilton-Jacobi-Bellman ligning. Årsagen til at det er belejligt er at, hvis én løsning findes til en Hamilton-Jacobi-Bellman ligning, så er det også den optimale løsning[2, s.103]. Først opdeles integralet med et punkt $p$:

$$V(D(t),T(t),t) = \max_{\phi} \left\{ \int_t^{p} f(D(s)) \cdot U(T(s)) ds + V(D(p),T(p),p) \right\} \tag{4.11}$$

hvor det sidste led er tunens fitness fra punktet $p$ til $T_s$. Taylor udviklingen af integralet er angivet i Ligning (4.12), hvor højereordensledene er udeladt.

$$\begin{aligned} V(D(t),T(t),t) = \max_{\phi} \{ & f(D(s)) \cdot U(T(s)) \cdot (p-t) + \\ & \frac{\partial V}{\partial D}\frac{dD}{dt} \cdot (p-t) + \\ & \frac{\partial V}{\partial T}\frac{dT}{dt} \cdot (p-t) + \\ & \frac{\partial V}{\partial t} \cdot (p-t) + \\ & V\left(D(t),T(t),t\right) \} \end{aligned} \tag{4.12}$$

Det ses, at $V\left(D(t), T(t), t\right)$ optræder på begge sider af lighedstegnet, samt at $(p-t)$ indgår i alle ledene. Hvis det forkortes ud findes:

$$0 = \max_{\phi} \left\{ f(D(s)) \cdot U(T(s)) + \frac{\partial V}{\partial D}\frac{\partial D}{\partial t} + \frac{\partial V}{\partial T}\frac{\partial T}{\partial t} + \frac{\partial V}{\partial t} \right\} \tag{4.13}$$

Ligning (4.13) er en Hamilton-Jacobi-Bellman ligning, hvilket var formålet at vise. Det er dog muligt at rearrangere ligningen lidt.

For at rearrangere Ligning (4.13), må der ses på hvad tunen aktivt selv kan gøre. Det tunen rent faktisk kan ændre er, hvorvidt den skal svømme op, ned eller ligge stille. Temperaturændringen er afhængig af dybdeændringen, og tunen kan spise hele tiden.

$$\begin{aligned} 0 &= f(D(s)) \cdot U(T(s)) + \frac{\partial V}{\partial T}\frac{\partial T}{\partial t} + \frac{\partial V}{\partial t} + \max_{\phi} \left\{ \frac{\partial V}{\partial D}\frac{\partial D}{\partial t} \right\} \\ 0 &= f(D(s)) \cdot U(T(s)) + \frac{\partial V}{\partial T} k(T(s) - T_a(D(s)))(T_a(D(s)) - T(s)) + \\ &\qquad \frac{\partial V}{\partial t} + \max_{\phi} \left\{ \frac{\partial V}{\partial D} \cdot U(T(s)) \cdot \sin(\Phi(s)) \right\} \end{aligned} \tag{4.14}$$

Ligning (4.14) er en førsteordens homogen ikke-lineær partial differentialligning med den tilknyttet terminal betingelse i Ligning (4.16).

## 4.2 Løsningens form

Det er interessant at løse Ligning (4.14) som et uendeligt horisont problem i stedet for et endeligt horisont problem. Løsningen hertil kaldes den asymptotiske løsning og er når $t \to \infty$. Tunen udfører mange dyk i løbet af en dag, og derfor kan det retfærdigøres at se på problemet som værende et uendeligt horisont problem. Når problemet er et uendeligt horisont problem, så gælder stationaritet og dermed i det her tilfælde, at energihøsten fra én tid til én anden er konstant [2, s. 362]. Når det er interessant at finde den asymptotiske løsning er det fordi, den optimale dykkestrategi for tunen, da er uafhængig af tiden [2, s. 362].

A priori antages det, at løsningen til Ligning (4.14) er på formen:

$$V(D, T, t) = V_{\infty}(D, T) + \gamma t \tag{4.15}$$

hvor $V(D, T, t)$ er en proxy til værdifunktionen, kaldet tunens fitness (også kendt som optimal cost-to-go funktion). $\gamma$ er den optimale fødehøstrate, og beskriver hvilken energihøst over tid en tun, der svømmer optimalt, vil få (den er konstant

i henhold til, at det er et uendeligt horizont problem). $V_\infty(D,T)$ er den optimale svømmestrategi for tunen (uaghængigt af tiden). A posteriori viser det sig, at være korrekt at antage denne løsnings form, se Afsnit 4.3.

Terminal betingelsen for Ligning (4.14) er bestemt ud fra følgende antagelse: Ved solnedgang har tunen ingen mulig fremtidig energihøst (når der kun ses på et døgn) og derfor er terminal betingelsen givet ved:

$$V(D,T,T_s) = 0 \tag{4.16}$$

En analytisk løsning af Hamilton-Jacobi-Bellman ligningen er ikke mulig, så i det følgende vil en numerisk løsning gennemgås [2, s. 25].

## 4.3 Numerisk løsning

Når Hamilton-Jacobi-Bellman ligningen løses, findes tunens optimale adfærd. En overordnet gennemgang af metoden beskrives i det følgende. Selve MATLAB koden er vedlagt i Appendiks D og er en udvidelse af et kodeafsnit fra Uffe Høgsbro Thygesen.

Først ses på en diskretisering af problemet. En dybde vektor, $\mathbf{d}$, laves ved at sample ækvidistant fra 0 til $z_{ss}$ adderet med 5 standard afvigelser $q$, hvor $N_d = 50$.

$$\mathbf{d} = [0, \ldots, z_{ss} + 5q] = \left[d_1, \cdots, d_{N_d}\right] \tag{4.17}$$

Temperaturvektoren $\mathbf{T}$ er ligeledes ækvidistant samplet mellem nul og $T_{\max}$.

$$\mathbf{T} = \left[0, \ldots, T_{\max}\right] = \left[T_1, \ldots, T_{N_T}\right] \tag{4.18}$$

hvor $N_T = 45$. $\mathbf{K}$ er en matrix med elementer givet ved Ligning (4.19), som beskriver varmeledningsraten for tunen.

$$K_{ij} = k_{ij}(T_i^T - T_a(d_j)) \cdot (T_a(d_j) - T_i^T), \quad i \in [1, N_T],\ j \in [1, N_d] \tag{4.19}$$

hvor $k_{ij}$ er varmeledningskonstanten fra Ligning (3.4). $T^a$ repræsenterer den ambiente temperatur, mens $T^T$ repræsenterer tunens temperatur.

En matrix $\mathbf{V}$ defineres som tunens fitness til en given tid, hvor $V_{ij}$ er tunens fitness til den $i$'te dybde og $j$'te temperatur. Tunens optimale adfærd samt den optimale energihøstrate kan findes ved at gå baglæns startende fra tunens fitness ved solnedgang. Terminalbetingelsen ved solnedgang er angivet i Ligning

(4.16). Der gås tilbage 1 sekund ad gangen, $S = 1$. Ved at anvende "operator splitting"kan opdatering af tunens fitness ske i to dele. Operator splitting metoden kan anvendes ved numerisk løsning af ikke-lineære partielle differentialligninger ved at dele den partielle differentialligning op i en lineær del og i en ikke-lineær del [24] [5, s. 106]. Først behandles den lineære del og $\mathbf{V}$ opdateres søjlevis, dvs. ud fra temperaturen. $\bar{V}_{ij}$ er en midlertidig værdi af $V_{ij}$:

$$\bar{V}_{ij}^{t} = V_{ij}^{t-1} + K_{ij} \cdot S \cdot \frac{V_{i,j+1} - V_{i,j}}{\triangle T} \quad \text{for } K_{ij} > 0 \tag{4.20}$$

$$\bar{V}_{ij}^{t} = V_{ij}^{t-1} - K_{ij} \cdot S \cdot \frac{V_{i,j-1} - V_{i,j}}{\triangle T} \quad \text{for } K_{ij} < 0 \tag{4.21}$$

Da temperatur vektoren indeholder ækvidistant fordelte temperaturer er temperaturforskellen mellem to dybder, $\triangle T$, konstant og det samme gælder dybdeforskellen $\triangle D$. Kanterne i $\mathbf{V}$ håndteres specielt for at holde tunen inden for området.

Herefter behandles den ikke-lineære del og opdatering af $\mathbf{V}$ sker rækkevis dvs. ved at se på tunens fitness forøgelse ved at variere dybden:

$$V_{ij}^{t} = \bar{V}_{ij}^{t-1} + E_{ij} \cdot S \qquad \text{for } V_{ij} > V_{(i-1)j} \wedge V_{ij} > V_{i+1,j} \tag{4.22}$$

$$V_{ij}^{t} = \bar{V}_{ij}^{t-1} + E_{ij} \cdot S + S \cdot \frac{V_{(i-1)j} - V_{i,j}}{\triangle D} \cdot U_{ij} \qquad \text{for } V_{ij} < V_{(i-1)j} \tag{4.23}$$

$$V_{ij}^{t} = \bar{V}_{ij}^{t-1} + E_{ij} \cdot S + S \cdot \frac{V_{(i+1)j} - V_{i,j}}{\triangle D} \cdot U_{ij} \quad \text{for } V_{ij} < V_{(i+1)j} \tag{4.24}$$

Herefter startes forfra med at opdatere med temperaturen. $E_{ij}$ er energihøsten givet i Ligning (2.11). De to former for opdatering af $\mathbf{V}$ fortsætter indtil forbedringen er meget lille, og systemet dermed har nået en tilstand meget tæt på ligevægt.

### 4.3.1 Stopkriterie

For at afgøre hvor mange iterationer af $\mathbf{V}$, der skal til før end den asymptotiske løsning i Ligning (4.15) nåes, udregnes efter hver iteration $t$ følgende:

$$\triangle \mathbf{V} = \mathbf{V}_t - \mathbf{V}_{t-1} \tag{4.25}$$

Når asymptoten er nået, gælder det som sagt, at energihøsten fra én tid til én anden er konstant; det vil sige at:

$$\frac{\partial^2 V}{\partial t \partial D} = \frac{\partial^2 V}{\partial t \partial T} = 0 \tag{4.26}$$

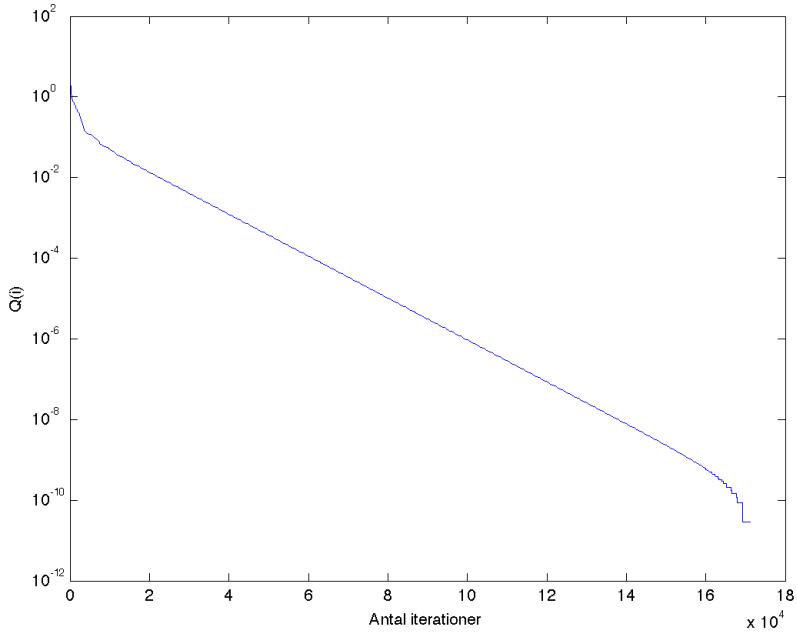
For at kontrollere om uafhængigheden af dybden og temperaturen er nået, som antaget i Ligning (4.15), udregnes Ligning (4.27) efter hver iteration.

$$Q_t = \max(\triangle \mathbf{V}) - \min(\triangle \mathbf{V}) \tag{4.27}$$

hvor max og min udtager henholdsvis det numerisk største og numerisk mindste element i $\mathbf{V}$. Når $Q$ er tæt nok på nul sluttes iterationerne. På Figur 4.1 ses et plot over udviklingen af $Q$ i forhold til iterationerne. Ud fra plottet vælges stopkriteriet til at være når:

$$Q \leq 1 \cdot 10^{-8} \tag{4.28}$$

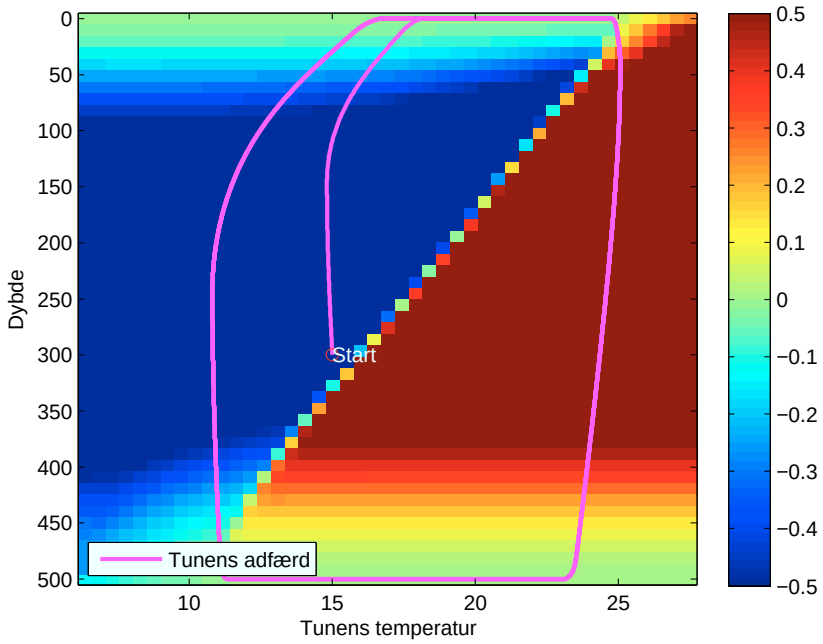
På det tidspunkt er den forbedrede energihøst så lille at det kan antages at det ikke længere giver et reelt positivt bidrag for tunen.




**Figur 4.1:** Bestemmelse af stopkriterie for antal iterationer af $\mathbf{V}$, der skal til før end den asymptotiske løsning er nået. Den rette linje viser, at der er stabil linearisering i nærheden af steady state løsningen til Hamilton-Jacobi-Bellman ligningen. Det zigzak der sker omkring $1 \cdot 10^{-10}$ skyldes formentlig maskinnøjagtigheden. Det vælges at sætte stopkriteriet til at være når $Q \leq 1 \cdot 10^{-8}$.

### 4.3.2 Simulering af tunens bevægelse

For at kunne sammenligne den optimale adfærd med det observerede dykkemønster fra data, er det nødvendigt, at kunne simulere tunens bevægelse i løbet af et døgn. Ud fra løsningen givet ved Ligning (4.15) er den optimale strategi for adfærden givet ved $V_\infty(D,T)$, og denne er uafhængig af tiden. Det betyder, at tunen til en given temperatur og dybde har én optimal adfærd, uanset hvornår på døgnet det er. På Figur 4.2 er vist den mulige fitness ændring ved at svømme ned givet ved Ligning (4.30). Således er det bedst at svømme ned, når udtrykket er positivt og omvendt ved negativ, mens nul medfører, at tunen skal bibeholde dybden. Generelt kan det konkluderes, at hvis tunen er nede i dybdet og varm, skal den blive nede, mens hvis den er varm og oppe mod overfladen skal den svømme ned. Det omvendte gælder hvis tunen er kold.




**Figur 4.2:** Den optimale strategi for tunen, vist ved $\triangle V_\infty$ givet i Ligning (4.30). Det er bedst at svømme ned, når udtrykket er positivt og omvendt ved negativ, mens nul medfører, at tunen skal bibeholde dybden. Den lyserøde streg viser tunens adfærd ud fra den optimale strategi.

Ud fra den optimale strategi kan en simulering af dybden $D^t_{sim}$ og temperatur $T^t_{sim}$ til et hvilket som helst tidspunkt $t$ findes, og på den måde findes en samlet

simulering af løsningen. Simuleringsmetoden adskiller sig en lille smule fra selve den numeriske løsningsmetode. I Ligningerne (4.22)-(4.24) er angivet en enten eller løsning ved at sammenligne rækkerne i $\mathbf{V}$. Enten så skal tunen holde samme dybde, svømme op eller svømme ned. For at ændre denne diskontinuitet mellem de tre forskellige muligheder, indføres en funktion $\varphi$:

$$\varphi = -1 + \frac{2}{1 + e^{-\triangle V_\infty \cdot 10}} \tag{4.29}$$

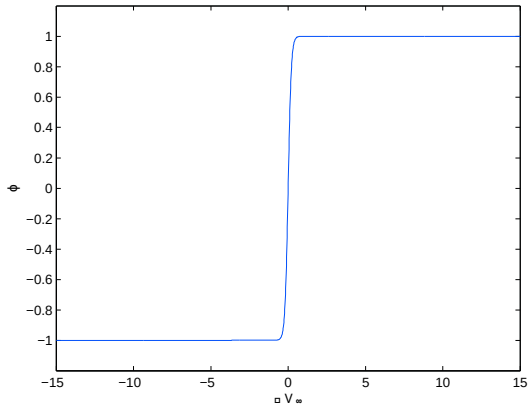
hvor $\triangle V_\infty$ er fitness forskellen givet ved følgende:

$$\triangle V_\infty = \frac{V_{\infty_{(i+1)j}} - V_{\infty_{(i-1)j}}}{2\triangle D} \tag{4.30}$$

På Figur 4.3 er funktionen illustreret. Startende fra venstre på figuren betyder det, at tunen dykker ned, så længe funktionen har samme $\varphi$ værdi. Herefter er der en "blød"overgang til, når tunen holder samme dybde, når $\triangle V_\infty = 0$ og derefter en overgang igen til når tunen svømmer op. Ved at arrangere Ligning (4.29) og (4.30) på algoritmisk form kan tunens dykkeadfærd simuleres:

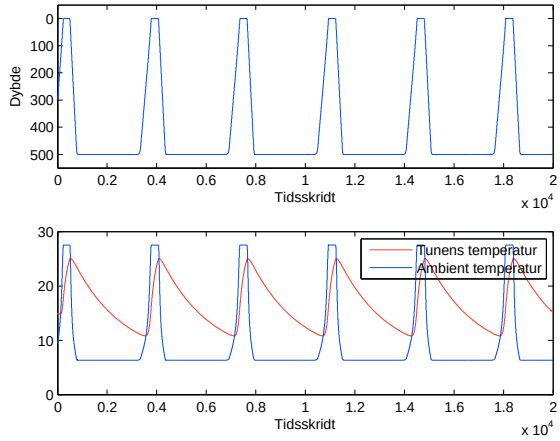
$D^1_{sim} = 0$
**for** $t = 1, \ldots, 86400$ **do**
  $D^{t+1}_{sim} = D^t_{sim} + S \cdot U^t \cdot \varphi^{\triangle V^t_\infty}$
**end**

hvor 86400 er antallet af sekunder på et døgn og tunen i simuleringen er sat til at starte i overfladen.




**Figur 4.3:** Plot af Ligning (4.29) som sørger for at overgangen mellem de tre mulige adfærer ikke er en binær beslutning.

På Figur 4.2 er tunens adfærd illustreret med den lyserøde farve. Adfærden er givet ud fra et startpunkt og viser, hvorledes den optimale strategi medfører cykliske bevægelser. På Figur 4.4 er det vist, hvordan denne adfærd forløber over tid. Tunens adfærd er vist sammen med temperaturen. Her er det spændene at se, at tunens laveste temperatur er omkring 11°C. Det er ikke urealistisk at antage, at tunen bliver så kold, se [3, s. 8]. Det er yderligere interessant at lave en simulering af tunens adfærd for at kunne sammenligne den beregnede optimale adfærd med den observerede adfærd fra data, se Figur 1.2(b). Det kan her umiddelbart konkluderes, at der er sammenhæng, mellem det observerede dykkemønster og det simulerede, og dermed kan påstanden om, at tunens adfærd kan forklares ud fra optimal fourageringsteori underbygges.




**Figur 4.4:** Tunens dykkemønster, temperatur og tilhørende ambiente temperatur. Dykkemønstret og temperaturen er simuleret ud fra den optimale strategi vist på Figur 4.2.

## 4.4 Opsummering

I dette kapitel er det vist, at et dynamisk optimeringsproblem kan opstilles for tunens energihøst. Optimeringsproblemet kan sættes på Hamilton-Jacobi-Bellman form ved at benytte dynamisk programmering, og dermed kan en optimal energihøst og optimal strategi findes som den numeriske løsning. Det dertilhørende simulerede dykkemønster er sammenligneligt med det observerede data og dermed underbygger det antagelsen om, at tunens dykkemønster kan forklares ud fra optimal fourageringsteori. I det følgende kapitel vil det blive gennemgået, hvad der sker hvis en parameterværdi i modellen varieres.

Kapitel 5

# Variation af parametre

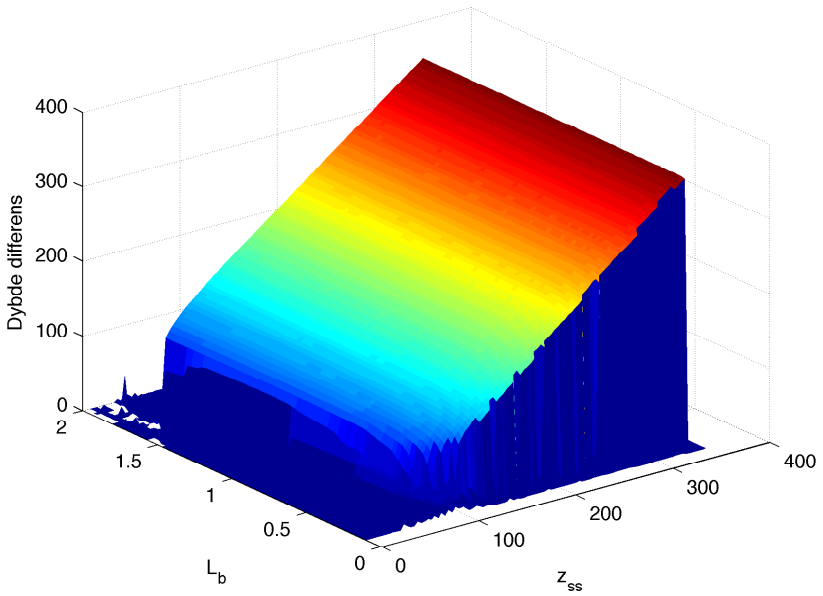
Efter nu at være i stand til at finde en numerisk løsning til den optimale dykkeadfærd, er det muligt at undersøge, om der findes en grænse for to af parametrene, således at tunens optimale adfærd går fra at være ved konstant dybde til at være ved variabel dybde. Dette skal undersøges for længden af tunen $L_b$ og for dybden med den maksimale fødetæthed $z_{ss}$.

$L_b$ og $z_{ss}$ bør varieres samtidigt, som det er gjort i Figur 5.1, men for at få et mere tydeligt billede, ændres de to paramtere i det følgende hver for sig. Mønstret i Figur 5.2(a) og Figur 5.2(c) kan således genfindes i Figur 5.1.

Tunens størrelse ændres ved at ændre på dens "fork length"$L_b$, som sættes til at variere mellem 0.1 m og 2.0 m. Fork length er en måde at måle tunens længde på og måles fra snuden til halen. Målepunktet på halen er midten af gaflen [6], deraf navnet. Svømmehastigheden antages i denne opgave at være påvirket af længden af tunen, og derfor omskrives svømmehastighedsligningen, Ligning (2.3) til udtrykket givet i Ligning (5.1). Det gamle udtryk for svømmehastigheden er på den måde for en tun med længden 0.74 m [14, s. 6] .

$$U(T) = (0.05 \cdot T + 0.48) \cdot \frac{L_b}{0.74} \tag{5.1}$$

På Figur 5.2(c) er vist hvorledes tunens adfærd ændres. Indtil $L_b$ når en vis størrelse, kan det ikke betale sig for tunen at svømme op og ned, og den vil ligge




**Figur 5.1:** Tunens adfærdsændring når $L_b$ og $z_{ss}$ varieres. Dybde differensen er forskellen mellem den maksimale og minimale dybde. På Figur 5.2(a) og Figur 5.2(c) varieres $z_{ss}$ og $L_b$ hver for sig.

ved konstant dybde. Når $L_b$ bliver større kan det dog betale sig for tunen at svømme mellem det varme vand uden føden og det kolde vand med føden. Den størrelse af $L_b$, hvor tunen går fra at ligge ved konstant dybde til at svømme op og ned, defineres som $s_b$.

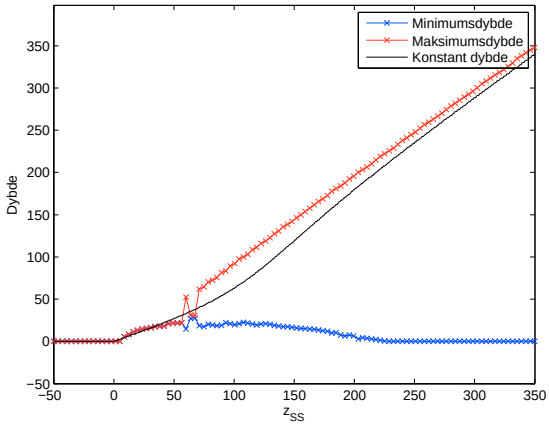
Dybden, hvor der er den maksimale fødetæthed $z_{ss}$, skal ligeledes varieres, for at se hvordan det påvirker tunens adfærd. I Ligning (2.2) er fødefeltet defineret. På Figur 5.2(a) er tunens maksimale og minimale dybde plottet for hver $z_{ss}$ værdi sammen med den dybde, tunen ville have, hvis den havde konstant dybde. $z_{ss}$ antager også negative værdier, og det skal i den forbindelse bemærkes, at tunen ikke kan opholde sig ved negative dybder, men at funktionsudtrykket for fødetætheden er veldefineret for $z_{SS} < 0$. Det er her interessant at se, at når fødetætheden ligger på lav nok dybde, så kan det ikke betale sig for tunen at svømme op og ned. I stedet vil den finde en konstant dybde at ligge ved. Dybden $z_b$ defineres som den dybde, hvor tunen går fra at holde konstant dybde til at variere dybden.

For begge parametre kan ændringen i svømmeadfærden forklares med begrebet bifurkationer. Bifurkationer kan optræde ved dynamiske systemer og er når løsningen ændrer sin struktur som følge af at en parameter ændres [9, s. 245]. Bifurkationspunkterne er $z_b$ og $s_b$ og er givet ved:

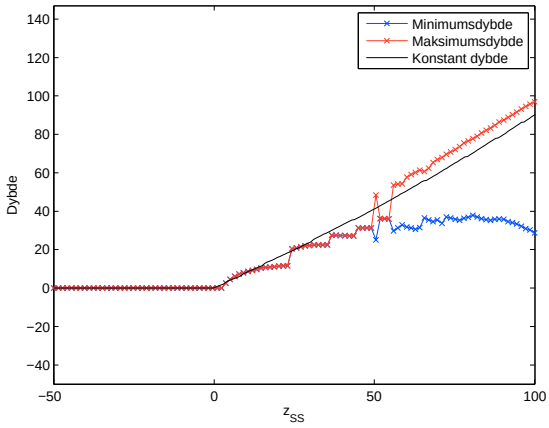
$$z_b = 60\ m,\ s_b = 0.22\ m \tag{5.2}$$

For at se hvordan tunens adfærd er før og efter $z_b$ og $s_b$, er der på Figur 5.3 og Figur 5.4 plottet tunens simulerede adfærd. Der er to forhold som skal bemærkes på de fire plots. Det første forhold er især tydeligt på Figur 5.3(a). På begge figurer er den givne længde og dybde mindre end henholdsvis $z_{ss}$ og $s_b$. Her burde tunens adfærd gøre, at dybden tunen tilslut fastholde er den samme dybde som blev fundet ud fra modellen med konstant dybde i Kapitel 2. Det skal understreges at denne afvigelse ikke har noget med bifurkationer at gøre, men derimod skyldes at selve $V_\infty$ kurven er flad, se Figur 5.5. Da kurven er flad, er det svært at ramme det præcise maksimum, en yderlige beskrivelse findes i Kapitel 6.

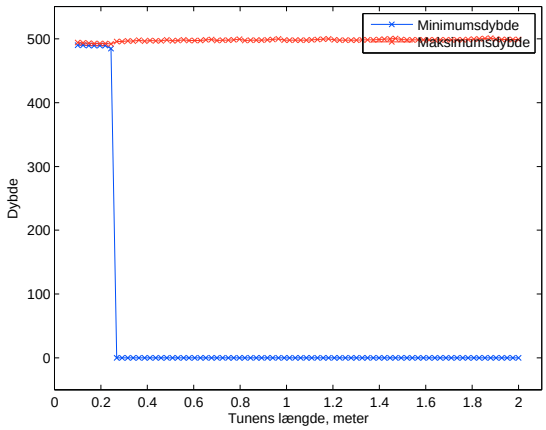
Det andet forhold der skal bemærkes ses på Figur 5.3(b) og 5.4(b). Her er adfærden plottet for flere værdier af $L_b$ og $z_{ss}$, som alle er større end henholdsvis $s_b$ og $z_b$. Uden yderligere bevis, påståes det, at det mønster der ses skyldes superkritiske og subkritiske bifurkationer. Det er navne for to forskellige Hopf bifukationer. En superkritisk bifurkation resulterer generelt set i en bevægelse med lille amplitude omkring den oprindelige løsning, hvorimod en subkritisk medfører en bevægelse med større amplitude [20, s. 249 og s. 252]. En hurtig, men ikke så grundig måde, at skelne de to bifurkationstyper fra hinanden kan




(a) Tunens adfærdsændring når $z_{ss}$ varieres.

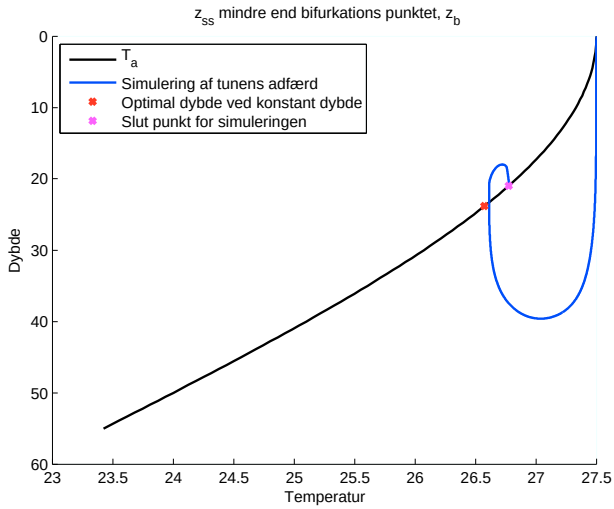



(b) Zoom på området omkring $z_b$.

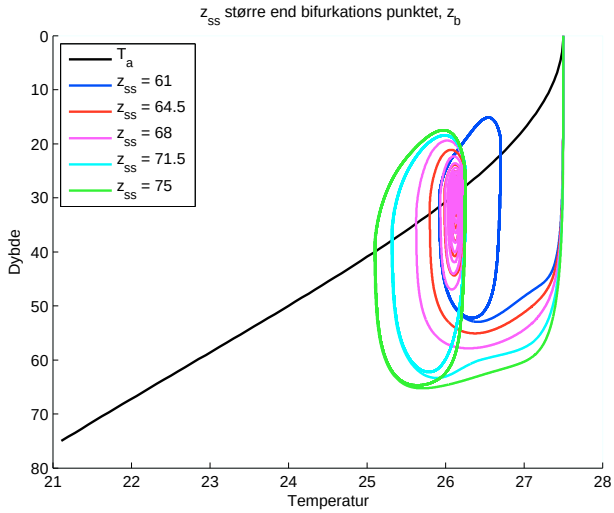



(c) Tunens adfærdsændring når $L_b$ varieres.

**Figur 5.2:** Tunens adfærd når $z_{ss}$ og $L_b$ varieres vist sammen med den konstante dybde. Bifurkationspunkterne er: $z_b = 60$ m, $s_b = 0.22$ m.

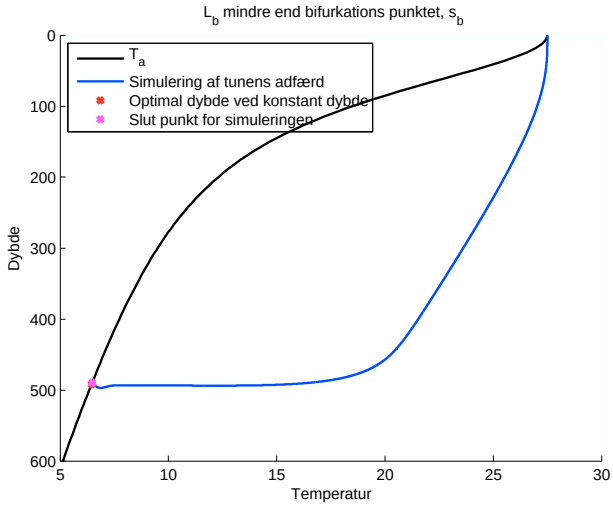



(a) Tunens adfærd når $z_{ss}$ er mindre end $z_b$.

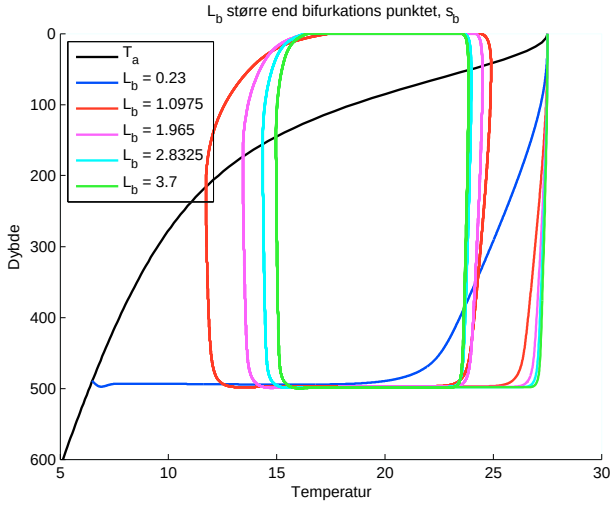



(b) Tunens adfærd når $z_{ss}$ er større end $z_b$.

**Figur 5.3:** Plot af tunens adfærd på hver sin side af $z_b$. Når tunen holder henholdsvis konstant dybde og variabel dybde. På Figur 5.3(b) kan den superkritiske bifurkation ses.




(a) Tunens adfærd når $L_b$ er mindre end $s_b$.

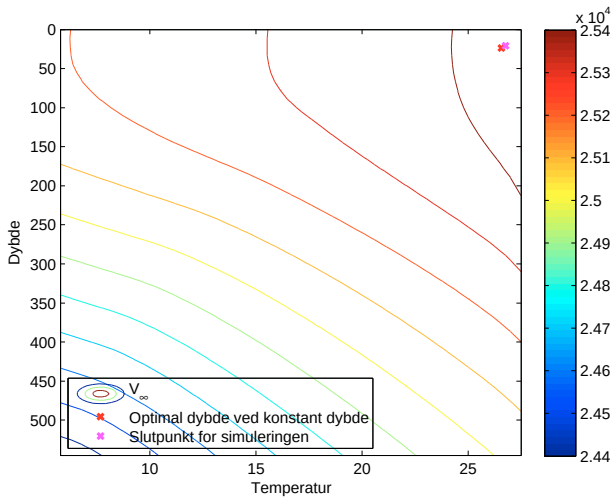



(b) Tunens adfærd når $L_b$ er større end $s_b$.

**Figur 5.4:** Plot af tunens adfærd på hver sin side af $L_b$. Når tunen holder henholdsvis konstant dybde og variabel dybde. På Figur 5.4(b) kan den subkritiske bifurkation ses.

således være, at se hvordan bevægelsen er efter bifurkationspunktet [20, s. 253]. Det kan ud fra Figur 5.4(b) og Figur 5.3(b) konkluderes, at der, når $L_b$ ændres, er tale om en subkritisk bifurkation. Omvendt er der ved ændring af $z_{ss}$ tale om en superkritiske bifurkation.

Det er postuleret, at der er tale om Hopf bifurkationer, men hvis det skal verificeres, at det er sandt, skal egenværdierne for Jacobianten til systemet undersøges. Ved Hopf bifurkationer er egenværdierne, $\lambda$, kompleks konjugerede. Når løsningen er stabil er $\mathrm{Re}\{\lambda\} < 0$ og ved en destabilisering af løsningen får en eller begge egenværdierne positiv real del. En variation af en parameter medfører $\mathrm{Re}\{\lambda\} > 0$ [20, s. 249] og i selve bifurkationspunktet er $\lambda = 0$ [9, s. 246]. En detaljeret gennemgang af stabiliteten for Ligning (4.14) er for omfattende for denne rapport og gennemgåes ikke yderligere.




**Figur 5.5:** Plot af $V_\infty$ for tilfældet givet i Figur 5.3(a). Det ses at $V_\infty$ er meget flad omkring maksimummet.

## 5.1 Eksempler på anvendelse

I Kapitel 1 blev det nævnt, at det optimale dykkemønster kunne interessere eksempelvis havbiologer. I det følgende er der givet to eksempler på anvendelse af den optimale løsning.

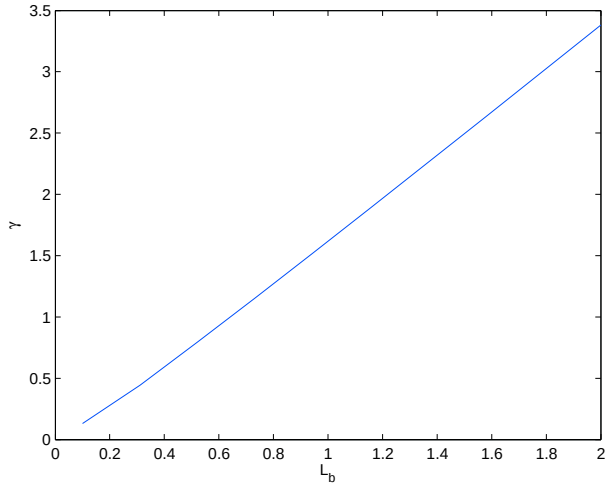
I dette kapitel er det nævnt, hvorledes tunens adfærd ændres, når $z_{ss}$ og $L_b$

varieres. Det er her interessant at undersøge, hvad det, ifølge modellen, betyder for tunens optimale fødehøstrate $\gamma$. På Figur 5.6(a) og 5.6(b) er vist, hvorledes tunens fødehøstrate bliver mindre og mindre jo dybere føden befinder sig. Når føden er tæt på overfladen, kan tunen være i det varme vand samtidig med at den kan jage, men når føden rykker længere ned, må den pendle frem og tilbage. Således er det muligt ud fra den optimale adfærd at sige noget om, hvorledes en ændret fødetæthed vil påvirke tunen. På Figur 5.6(a) er det iøvrigt tydeligt, at der i modellen ikke er knyttet nogen energi udgifter til, at tunen bliver større. Energi udgifter kunne f.eks. være tunens varmetab. I Kapitel 6 vil det blive diskuteret yderligere.

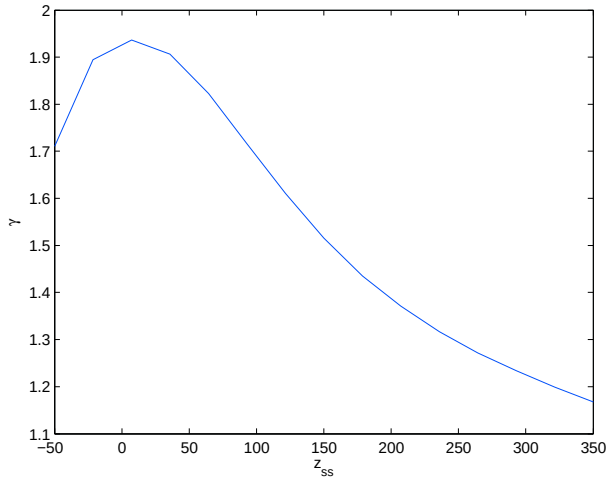
Et andet interessant forhold at se på er, hvordan fordeling af hvor meget tunen svømmer, fouragerer og får varmen i løbet af en dag ændres, som resultat af at $z_{ss}$ og $L_b$ varieres. På Figur 5.7 er det vist, hvordan tunen i følge modellen fordeler sin tid i løbet af en dag, alt afhængigt af størrelsen af $L_b$ og $z_{ss}$. Tidsbudgettet er vist efter bifurkationspunkterne når tunen ikke længere ligger ved en konstant dybde. Når $z_{ss}$ varieres er der store ændringer lige efter bifurkations punktet, men derefter bibeholdes et vist fohold mellem de tre adfærde. Når størrelsen af $L_b$ stiger mindskes tiden tunen ligger i overfladen, samtidigt med at tiden, hvor den spiser og svømmer, vokser. Dette fortsætter indtil den når en given længde, hvorefter det modsatte sker. Hvis det antages at forskellige længde tun er ensbetydende med tun i forskellige aldre, så er det muligt ud fra den optimale adfærd, at sige noget om hvordan tunens adfærd ændres i takt med, at den ældes.

## 5.2 Opsummering

Det er vist, at når parametrene varieres, opstår der en grænse, et bifurkations punkt, hvor tunen går fra at holde konstant dybde til at variere sin dybde. Følgende parametre er varieret: tunens længde $L_b$ og dybden med den største fødetæthed $z_{ss}$. Grænsen for parametrene er givet ved værdierne $z_b$ og $s_b$ i Ligning (5.2).

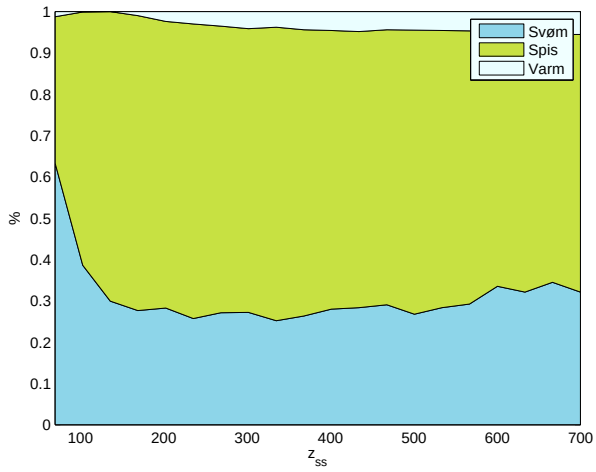



(a) Fødehøstraten når $L_b$ varieres.

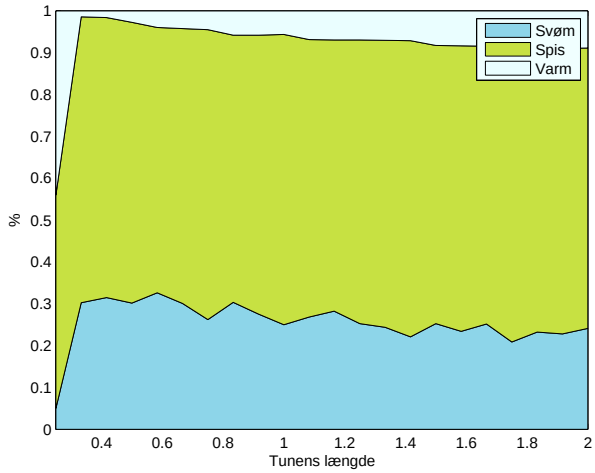



(b) Fødehøstraten når $z_{ss}$ varieres.

**Figur 5.6:** Tunens fødehøstrate $\gamma$ når $L_b$ og $z_{ss}$ varieres.




(a) Tidsbudget for tunen for en dag ved forskellige værdier af $z_{ss}$.




(b) Tidsbudget for tunen for en dag når $L_b$ varieres.

**Figur 5.7:** Ud fra den optimale adfærd er tidsbudgettet for tunen i løbet af en dag undersøgt. Tidsbudgettet er undersøgt for begge parametre større end bifurkationspunktet.

Kapitel 6

# Diskussion

Figur 1.2(b) i Kapitel 1 viser tunens observerede dykkemønster. Figuren viser dykkemønstret for én dag ud af mere end 10 måneder. Den specifikke dag er valgt, fordi den viser adfærden på en typisk dag, lignende mønstre er rapporteret i [**?**, Malte] En typisk dag er defineret ved dykketure til dybder på $300 - 500$ m med regelmæssige hurtige opstigninger til dybder mindre end 200 m [3, s. 8].

Ved en kvantitativ sammenligning mellem data og tunens simulerede dykkeadfærd, kan det undersøges i hvilken grad frekvensen af tunens ture mod overflade er den samme. Når simulering af tunens adfærd i Figur 4.4 sammenlignes med Figur 1.2(b), ses det hurtigt, at frekvensen for opstigningerne er langt større i simuleringen end ved data. Figur 1.2(b) er dog som nævnt kun én dags dykkemønster ud af mange. Spørgsmålet er i virkeligheden, hvorvidt det giver mening at lave en kvantitativ sammenligning mellem data og simuleringen, når modellen ikke er fittet mere præcist til data. Resultatet af en sådan sammenligning ville formentlig blot resultere i et ikke statistisk signifikant fit. Det ville være en opgave i sig selv at lave en grundig statistiske analyse af data og envidere fitte modellen yderligere til data. Som beskrevet i [1, s. 74] er mangel på et godt fit mellem data og modellen ikke årsag nok til at afvise modellen.

Det vigtige at konkludere er, hvorvidt modellen er i stand til, at beskrive hvad der driver tunens adfærd [1, s. 74]. I dette tilfælde er en kvalitativ sammenligning således mere på sin plads. Ved igen at sammenligne data og simuleringen kan

det med rimelighed konkluderes, at optimal fourageringsteori ligger til grund for tunens specielle dykkemønster observeret i data. Forslag til forbedringer af modellen er gennemgået i Afsnit 6.2.

Det er valgt at opstille optimeringsproblemet som en Hamilton-Jacobi-Bellman ligning, men det havde også været muligt at bruge Pontryagins maksimum princip. Fordelen ved at bruge Hamilton-Jacobi-Bellman er, at hvis der findes en løsning, så er det den optimale løsning. Det er dog ikke sikkert, at der findes en løsning, da det ikke vides på forhånd, hvorvidt V(D,T,t), i Ligning (4.14), er differentiabel [2, s. 102]. Ved Pontryagins maksimum princip gælder det derimod, at der findes en løsning, men at denne ikke nødvendigvis er den maksimale og at bevis for optimalitet derfor må findes [2, s. 111]. I denne situation havde en fordel ved Pontryagins maksimum princip været, at problemet var reduceret til løsning af fire sædvanlige differentialligninger [1, s. 157], i stedet for en ikke-lineær partial differentialligning. Hvis tre tilstandsvariable var valgt, eksempelvis en ekstra temperatur for tunen eller lysintensiteten, havde brugen af Hamilton-Jacobi-Bellman været mulig men utrolig krævende. Beregningstiderne var vokset enormt, og Pontryagins maksimum princip havde været at foretrække. Således er der både fordele og ulemper ved begge løsningsmetoder.

## 6.1 Sensitivitet og robusted for modellen

Indenfor nogle områder er modellen ganske velfunderet, mens den andre steder er mere spekulativ, f.eks. er parametrene valgt ud fra en subjektiv vurdering. Derfor følger her to eksempler på små ændringer af parameterværdier. Små ændringer af en parameter og den resulterende effekt på løsningen, kan beskrives ved parametrens elasticitet. Parametrens elasticitet er det procentuelle forhold mellem ændringer i to variable og udtrykker sensitiviteten for modellen [1, s. 80].

- Ændringen af varmeledningskonstanterne $k_{low}$ og $k_{high}$ påvirker, hvor hurtigt tunen afgiver varmen og holder på varmen. Dermed påvirker det også, hvor lang tid tunen er oppe og nede og i sidste ende den optimale adfærd og den optimale fødehøstrate $\gamma$. Når $k_{low}$ ændres til at være 5 % mindre, så er ændringen i $\gamma$ på 1.45 %, og samlet svarer det til en elasticit i $\gamma$ relativ til $k_{low}$ på 0.29.
- Bredden af fødefeltet $q$ er et udtryk for, hvor koncentreret føden er. Sammen med $z_{ss}$ påvirker parameteren også, hvor langt ned tunen skal for at finde føde. En ændring i bredden af fødefeltet vil påvirke tunens optimale ædfærd og dermed også den optimale fødehøstrate $\gamma$. Når $q$ ændres til

at være 5 %, større så er ændringen i $\gamma$ på 0.17 %, og det svarer til en elasticitet på 0.03.

Robustheden for modellen er modsat sensitiviteten et udtryk for, hvad store ændringer i modellen vil medføre for resultatet. I det følgende er givet nogle eksempler på mulige ændringer.

- I Kapitel 5 er det vist, hvor stor en påvirkning af tunens adfærd det er, når dybden for fødetætheden ændres. Hvis fødetæthedsformen ændres fra den nuværende Gaussiske fordeling, vil det således ændre tunens optimale adfærd og dermed også energihøsten.

- Tunens svømmehastighed er afhængig af temperaturen, men det er realistisk at antage, at tunen på et tidspunkt under nedkøling vil svømme så langsomt, at det ikke er muligt for den at fange føde. Det vil være interessant at inkludere en grænse for, hvornår tunen svømmer for langsomt og dermed er nødsaget til at svømme op til lavere dybder. Det ville antageligt korte tiden i dybden af.

- Varmeledningsligningen tager højde for, at tunen har to forskellige varmeledningskonstanter. Alt afhængigt af om tunen er under opvarmning eller nedkøling, antager konstanten forskellige værdier. Der er dog mange forhold varmeledningsligningen ikke tager højde for. Det, at tunen udvikler varme når den svømmer og mister varme, når den spiser, kunne være medtaget. Temperaturfordelingen mellem det indre og ydre af selve tunen kunne også være medtaget og ydermere, kunne tunens størrelse også medregnes med hensyn til afgivelse og optagelse af varme, se også Afsnit 6.2.

- Som nævnt i afsnittet om Pontryagins maksimum princip, kunne det være interessant at inkludere én tredje tilstandsvariabel i form af lysintensiteten, som også er givet i data. Tunen ligger om natten nær overfladen og årsagen til det er angiveligt, at det bliver for mørkt om natten til, at den kan se noget i dybden [15]. Når lysintensiteten er inkluderet i modellen, kunne det ydermere være spændende at undersøge, hvordan det forholder sig med tunens byttedyr. Søger de også mod lyset og medfører det, at tunen også har mulighed for at jage om natten. Det ville dog kræve en tilføjelse til det eksisterende data.

## 6.2 Udgifter for tunen

I Afsnit 4.3 blev det understreget at kurven for $\mathbf{V}$ er meget flad omkring sit maksimum. Det er uheldigt, da det som vist gør det svært at finde det præcise maksimum. Det er oplagt at se på muligheden for at gøre dette maksimum mere tydeligt. En mulighed er at ændre modellen givet i Ligning (4.14), således at tunen ikke jager så effektivt, når den svømmer op eller ned:

$$\begin{aligned} 0 =& \frac{\partial V}{\partial T} k(T - T_a(D))(T_a(D) - T) + \\ & \frac{\partial V}{\partial t} + \max_{\phi} \left\{ \frac{\partial V}{\partial D} \cdot U(T) \cdot \sin(\Phi) + f(D) \cdot U(T) \cdot \cos(\Phi) \right\} \end{aligned} \tag{6.1}$$

En yderligere mulighed er at ændre tunens energi til også at inkludere energi udgifter. Det gælder f.eks. svømmeeffekten samt tunens varmetab, hvor svømmeeffekten er det energitab, tunen har, ved at opretholde den vertikale svømmehastighed. Hvis tunens varmetab bliver inkluderet i modellen vil det samlede udtryk kunne se ud som Ligning (6.3), hvor $c$ er tunens varmekapacitet.

$$0 = \frac{\partial V}{\partial T} k(T - T_a(D))(T_a(D) - T) - c \cdot \frac{k}{L_b}(T - T_a(D)) + \tag{6.2}$$

$$\frac{\partial V}{\partial t} + \max_{\phi} \left\{ \frac{\partial V}{\partial D} \cdot U(T) \cdot \sin(\Phi) + f(D) \cdot U(T) \cdot \cos(\Phi) \right\} \tag{6.3}$$

I Afsnit 5.1 blev det vist, at energihøst stiger i takt med at tunen bliver større. Det kunne være interessant at se på, om der er nogle mulige energi udgifter, som er forbundet med at tunen bliver større. Eksempelvis kunne det antages, at en større tun er længere tid end en mindre tun om at blive varmet op og nedkølet som foreslået i [15, s. 7].

Kapitel 7

# Konklusion

I denne opgave er årsagen til en storøjet tuns særprægede dykkemønster blevet undersøgt. Et dynamisk optimeringsproblem er opstillet for tunenes energihøst, og optimeringsproblemet er sat på Hamilton-Jacobi-Bellman form ved at benytte dynamisk programmering. Problemet er løst numerisk, og den asymptotiske løsning er fundet. Den optimale energihøst og den optimale strategi er givet ud fra den asymptotiske løsning og gør det muligt at simulere tunens dykkeadfærd.

I opgaven er det undersøgt, om der findes en eller flere parameterværdier, som når de ændres, får tunen til at ændre adfærd. Med andre ord er det undersøgt, om der findes et bifurkationspunkt således, at løsningen ændrer sin struktur som følge af, at parametrene ændres. Tunens længde og dybden for fødetætheden varieres. For begge parametre gælder det, at ved ét specifikt punkt går tunen fra at have konstant dybde til at svømme op og ned. Således kan det konkluderes, at bifurkationer finder sted.

To nye specifikke modeller er foreslået samt mulige udvidelser og forbedringer af den eksisterende model. Den ene nye model tager højde for, at tunen ikke spise så effektivt, når den svømmer op og ned. Det skulle resultere i at løsningens maksimum bliver mere specifik, hvilket er et problem for den eksisterende model. Den anden nye model er en udvidelse af den førnævnte, men hvor tunens varmetab er inkluderet som en energi udgift i modellen.

Den simulerede dykkeadfærd er sammenlignet med data, og det er konkluderet at omend simuleringen og oberservationerne fra data ikke er ens, så er modellen i stand til at beskrive, hvad der driver tunens adfærd. Det kan således konkluderes, at tunens dykkemønster kan forklares ud fra optimal fourageringsteori.

Bilag A

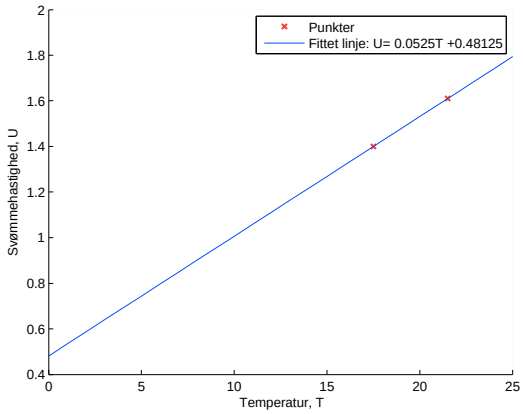
# Værdier for parametre og variable

| Parametre og variable | Enheder | Forklaring | Værdi |
|---|---|---|---|
| $z$ | $m$ | Vanddybden | |
| $T_{max}$ | $^\circ$ C | Overfladetemperaturen (når $z = 0$) | $27,5$ |
| $z_{Tcli}$ | $m$ C | Dybden hvor termoklinen slutter | 100 |
| $\lambda$ | | Hvor brat overgangen er når temperatur udviklingen skifter fra termoklinen til det linære temperatur fald i dybden | 2 |
| $A$ | $^\circ$ C | Temperaturfaldet over thermoklinen | 17 |
| $\rho$ | $\frac{^\circ C}{m}$ | Temperaturfaldet med dybden efter termoklinen | $\frac{-1}{100}$ |

**Tabel A.1:** Parametre og variable brugt til den ambiente temperaturudviklingen

| Parametre og variable | Enheder | Forklaring | Værdi |
|---|---|---|---|
| $f$ | | Fødetætheden | |
| $f_0$ | | Fødetætheden ved overfladen | 0 |
| $f_{ss}$ | | Den maksimale fødetæthed $z_{ss}$ | 1 |
| $z_{ss}$ | $m$ | Dybden hvor der er maksimal fødetæthed | 500 |
| $q$ | $m$ | bredden af det halve fødefelt | 100 |

**Tabel A.2:** Parametre og variable brugt til fødetætheden




**Figur A.1:** Svømmehastighed afhængigt af temperaturen. De to røde punkter er taget fra [14, s. 5 og s. 6]

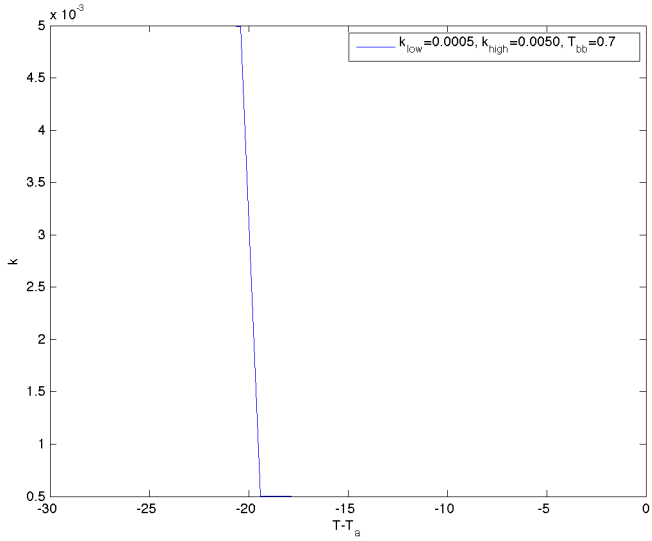
## A.1 Varmeledningen k

Forklaring på parameterværdierne i Ligning (3.4).

$k_{low}$ er varmeledningskonstanten, når tunen er nede i dybden. Værdien er lille som et udtryk for at tunen holder på varmen og ikke afgiver varmen til vandet så let. Omvendt er $k_{high}$ større, når tunen er oppe og gerne vil optage varmen fra det varme overfladevand [15]. $\triangle$T beskriver, hvor brat overgangen er fra $k_{low}$ til $k_{high}$. Parameteren $T_{bb}$ forskyder overgangen fra $k_{low}$ til $k_{high}$, det betyder, at tunens temperatur er lavere end omgivelserne inden tunen begynder at optage varme. I Tabel A.3 er værdierne for konstanterne angivet [15, s. 4]. På Figur A.2 er angivet et plot for varmeledningskonstanten. Ligning (3.4) for $k$ anvendes både når tunens adfærd modelleres med sinus svingninger og logiske svingninger.

| $\mathbf{k_{low}}$ | $\mathbf{k_{hign}}$ | $\triangle\mathbf{T}$ | $\mathbf{T_{bb}}$ |
|---|---|---|---|
| 0.0005 $min^{-1}$ | 0.005 $min^{-1}$ | 0.1 °C | 0.2 |

**Tabel A.3:** Værdier for konstanter til varmeledningen k, givet i Ligning (3.4)




**Figur A.2:** Varmeledningen, k. Værdierne for $k_{low}$ og $k_{high}$ er angivet på plottet. Plottet viser hvorledes tunen er koldere end det omgivne vand før den skifter strategi og begynder at optage varme.

## A.2 Omega

Af udtrykket for $\Phi$ ses det at $\omega \cdot t$ skal være dimensionløst. For at kunne opnå dette, skal der først findes et udtryk for perioden, $T_p$. Det vides fra [15][s. 125], at en periode for tunen er på ca. 20 minutter, dvs. $T_p = 1200s$. Sammenhængen mellem vinkelfrekvensen, $\omega$ og perioden, $T_p$ er givet ved:

$$\omega = \frac{2 \cdot \pi}{T_p} = 0.005 \tag{A.1}$$

Bilag B

# Variabel dybde

## B.1 Startbetingelser

De koblede differentialligninger for modellen med logiske svingninger løses i MATLAB ved hjælp af `ode45`. Startbetingelserne til tiden $t_0$ for dybden $D$ og temperaturen $T$ ses i Formel B.1 [15, s. 125]. Start energihøsten er sat til nul, da det kan antages, at tunen har meget lille energihøst i overfladen.

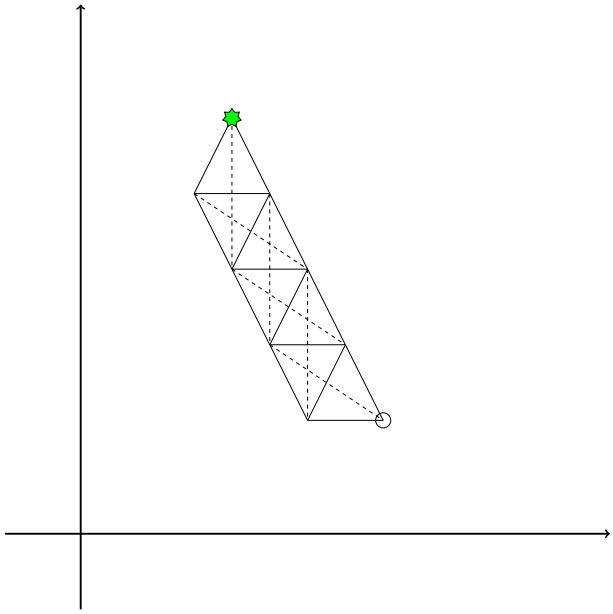
$$D(t_0) = 300m, T(t_0) = 15°C, E(t_0) = 0 \tag{B.1}$$

## B.2 `fminsearch`

`fminsearch` i MATLAB bruges til at finde maksimum for de logiske svingninger i Afsnit 3.4. Da `fminsearch` bruges til at finde minimum, findes den negative udgave af funktionen i stedet og det tilhørende minimum. Der optimeres på de to parametre for den linje, der adskiller ZONE1 og ZONE2 og som afgører om tunen svømmer op eller ned. Parametrene er: tunens temperatur ved overfladen $T_{high}$ og tunens temperatur i dybden, hvor der er mest føde $T_{low}$.

`fminsearch` anvender Simplex metoden også kaldet Nelder-Mead. Metoden evaluerer ud fra funktionsværdien og der bruges ikke den afledte af funktionen. Med metoden er det ikke helt sikkert, at det er det globale maksimum der findes. Metoden kan anvendes på funktionsudtryk med flere variable. Inden metoden forklares er det godt at specificere, at en 2-simplex er en trekant, en 3-simplex er en tetraeder og en $n$-dimesional simplex er formet af $n+1$ punkter. En regulær 2-simplex er således en ligesidet trekant [23].

For at forklare hvad der ligger bag `fminsearch` metoden, ses først på en basis simplex metode i planet. På Figur B.1 er metoden illustreret. En regulær 2-simplex dannes ud fra et angivnet startpunkt med en given sidelængde. Det hjørne i trekanten med den største funktionsværdi, spejles i linjen, udspændt af de andre to punker, og dermed haves en ny simplex. Metoden gentages, indtil det punkt, der lige er blevet spejlet, vil spejles tilbage igen til samme sted. Metoden der ligger til grund for `fminsearch` er en udvidet simplex metode. Den udvidede metode bruger, at det er muligt at lave siderne i trekanten større eller mindre, og dermed kan et mere præcist minimum findes [12].



**Figur B.1:** Illustration af basis simplex metoden. Den grønne stjerne er startpunktet og længden på siderne i trekanten er sat til 1. Slutpunktet er markeret med en cirkel. En udvidet simplex metode ligger til grund for `fminsearch`, der bruges til finde den maksimale energihøst, når tunens adfærd modelleres med logiske svingninger. Illustationen er med inspiration fra [12, s. 199]

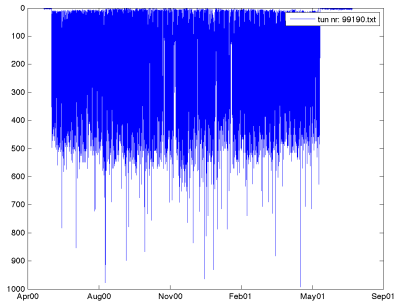
Bilag C

# Data fra CSIRO

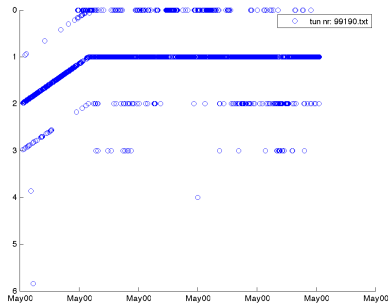
Jeg har fået stillet data til rådighed af Toby Patterson for tre forskellige tun [3]. Data indeholder en dybdemåling hvert fjerde minut.

På Figur C.1 er plottet dybdemålingerne for tunen med nummeret $99 - 190$. Det er interessant at undersøge tunens adfærd i løbet af et døgn, og det vælges at plotte for ét døgn cirka 200 dage inde i målingerne. Årsagen til det er, at i starten af dybdemålingerne afviger dykkemønstret fra det generelle mønster, hvilket tydeligt ses på Figur C.1(b). Dykkemønstret i slutningen af data afviger også fra det generelle, og det kan forklares med, at tunen bliver fanget, og dens svømmemønster dermed ophører. Der kan være flere årsager til at tunens svømmemønster afviger i starten; én oplagt mulighed kan være at senderen er tændt, men at tunen slet ikke har fået den på endnu.

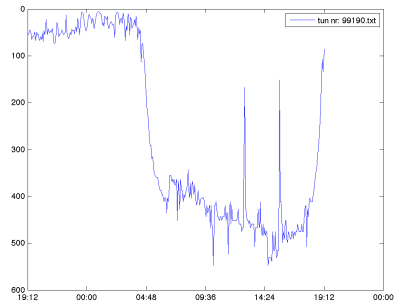
Figur C.2 og Figur C.3 viser dybdemålinger for to tun med numrene: $99 - 224$ og $99 - 243$.




(a) Al data

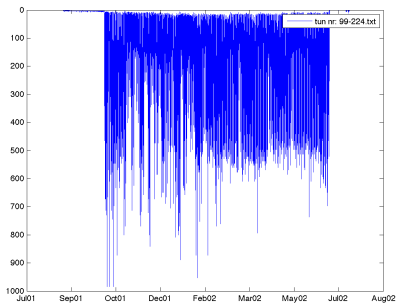



(b) For de første døgn

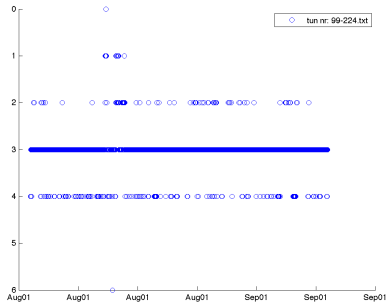



(c) Fra 200 dage og ét døgn frem

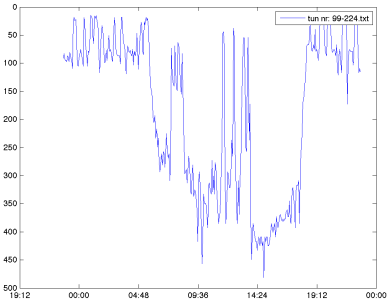
**Figur C.1:** Dybdemålinger for tunen med nummeret 99−190 og med en længde på 89 cm.




(a) Al data

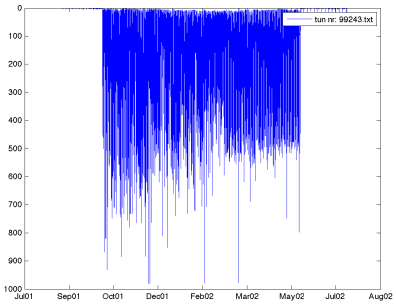



(b) For de første døgn

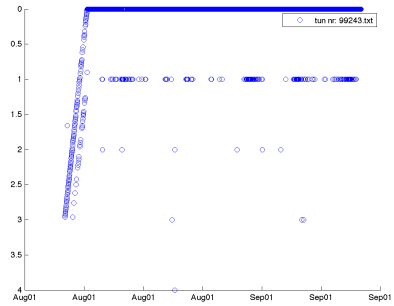



(c) Fra 200 dage og ét døgn frem

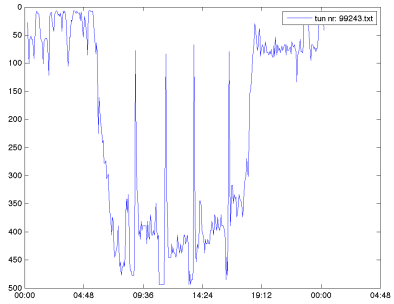
**Figur C.2:** Dybdemålinger for tunen med nummeret 99−224 og med en længde på 81 cm.




(a) Al data




(b) For de første døgn




(c) Fra 200 dage og ét døgn frem

**Figur C.3:** Dybdemålinger for tunen med nummeret 99−243 og med en længde op 83 cm.

Bilag D

# Matlab kode

```
1  %%Script til at opstille og løse det dynamiske udtryk for
       tunens adfærd.
2
3  %clc, clear, %close all
4
5  %% Konstanter
6
7  %Temperatur udviklingen i dybden
8  konstanter.Tmax = 27.5;  %overflade temperatur i grader
9  konstanter.p = -1/100; % temperaturfald efter thermoklin org
       . test=0.
10 konstanter.A = 17; % Temperaturfald over thermoklin org. 12
11 konstanter.Lambda = 2; % overgangen fra det blandede vand
       til kraftigere fald i temp
12 konstanter.zTcli = 100; %Thermoklinen
13
14 %Svømmehastighed ud fra temperatur i konstantDybde
15 konstanter.Vmin=(2.0*70)/100;
16 konstanter.Vmax = (2.3*70)/100;
17
18 %Værdier for fødetætheden
19 konstanter.f0=0; %fødetætheden ved overfladen
20 konstanter.fSS=1; %
21 konstanter.zSS=400; %dybden hvor der er mest føde, er dybF
       -100.
```

```
22 konstanter.q=50; %længden af overgangszonen org. 100
23
24 %Værdier til kTemp i varDyb
25 konstanter.klow=0.03/60; %pr minut hos Malte så del med 60
26 %konstanter.klow=0.03/60*10;
27 konstanter.khigh = 0.3/60; %pr minut hos Malte så del med 60
28 %konstanter.khigh = 0.3/60*10;
29 konstanter.Tbb = 0.2; % org. 0.7. forskydning så tunen
       bliver afkølet lidt, inden der skiftes til khigh
30
31
32 %Periode
33 konstanter.Tp = (20*60);  %Én periode er 20 minutter
34
35 %Omega og epsilon
36 konstanter.omega = (2*pi)/konstanter.Tp; %Vinkel frekvens
37
38 %Start betingelser
39 konstanter.D0 = 300; %ud fra Olivier s. 125.
40 konstanter.T0 = 15; %ud fra Olivier s. 125.
41 konstanter.E0 = 0; % energien den har til at starte med er
       nul.
42
43 konstanter.TidI = konstanter.Tp*10; % skal køres i et helt
       antal perioder
44
45 %Konstanter til Phi=svingPhi i varDybPhi
46 %Den rette linje ud fra de to punkter
47 konstanter.Tlow = 17.5;
48 konstanter.Thigh = 22.5;
49 konstanter.Dlinje = 500;
50
51 konstanter.U0=1.5;
52
53 %% Model funktioner, grids mm:
54
55 %Grids
56 nDyb=50; %Begrund valg
57 nTemp=45; %Bergund valg
58
59 %Skift: (indeksering)
60 Op=[1,1:(nDyb-1)]; %Rækkevektor med dobbelt start værdi
61 Ned=[2:nDyb,nDyb]; % Rækkevektor med dobbelt slut værdi
62
63 varme = [2:nTemp,nTemp]; %Rækkevektor med dobbelt slut
       nummer
```

```
64 kulde = [1,1:(nTemp-1)]; %Rækkevektor med dobbelt start
       nummer
65
66 %Tidsskridt
67 tidS=1;
68
69 %H=350
70 H=konstanter.zSS+5*konstanter.q;
71 dVec=linspace(0,H,nDyb); %50 punkter fordelt lige over de
       500 meters dybde
72 MadVec=fodetaet(dVec, konstanter); %Fødetætheden i hvert
       dybdepunkt
73 %MadVec2 = 1./(1+exp(-(dVec/H-0.85)*10));
74 TempVec=ambientTemp(dVec, konstanter); % vandtemp. i hvert
       dybdepunkt
75 %TempVec = 28 - 18*1./(1+exp(-(dVec/H-1/2)*10));
76
77
78 %Plot af Temperaur og fødetæthed afhængigt af dybden
79 figure(1)
80 plot(MadVec,dVec,'r')
81 hold on
82 plot(TempVec, dVec)
83 hold off
84 axis ij
85 ylabel('Dybde')
86 legend('Fødetæthed','Temperatur')
87
88 TempGrid=linspace(TempVec(end),TempVec(1),nTemp);
89 %Temperaturen stiger igennem vektoren. Der er 45 punkter
       mellem endepunkterne.
90
91 DeltaDyb = dVec(2)-dVec(1); %Forskellen mellem 2 dybde
       punkter.
92 DeltaTemp= TempGrid(2)-TempGrid(1); %Forskellen mellem 2
       temp punkter
93
94 %% State space
95 [DD,TT]=meshgrid(dVec,TempGrid);
96 %DD har størrelsen 45*50. Søjle 1 indeholder 45 rækker med
       værdien dVec(1),
97 %søjle 2 indeholder 45 rækker med værdien dVec(2)
98 %TT har størrelsen 45*50. Række 1 indeholder 50 kolonner med
        værdien
99 %TempGrid(1). Række 2 indeholder 50 kolonner med TempGrid(2)
        værdier
100
```

```
101 %% Temperatur- og fødefunktioner i state space punkterne
102 Tamb=meshgrid(TempVec,TempGrid);
103 %Størrelse 45*50. Hver kolonne k indeholder 45 kopier af
        TempVec(k).
104 FodeS=meshgrid(MadVec,TempGrid);
105 %Størrelse 45*50. Hver kolonne k indeholder 45 kopier af
        MadVec(k).
106
107 %Energihøst i statespace punkterne
108 Usvom=svomHastighed(TT, konstanter);
109 %Størrelse: 45*50. Svømmehastigheden til den ambiente temp.
        Hver række
110 %indeholder 50 kolonner som er ens.
111 EnergiH=FodeS.*Usvom;
112 %Størrelse: 45*50. Elementvis multiplikation af
        svømmehastighed og
113 %fødetæthed.
114 %EnergiH=FodeS.*(TT-5);
115
116
117 %Varmekoefficienten evalueret i statespace
118 k1 = kTemp(TT,DD, konstanter);
119 k= k1.*(Tamb-TT);
120 %KappaHeat = 0.02;
121 %KappaCool = 0.005;
122 %k = konstanter.khigh*max(0,Tamb-TT) + konstanter.klow*min
        (0,Tamb-TT);
123 %max: sætter nul alle de steder som er negative.
124 %min: sætter nul alle de steder som er positive.
125 %Alle de steder hvor vandet er varmere end "fisken" ganges
        med kappaheat
126 %Alle de steder hvor vandet er koldere end "fisken" ganges
        med kappacool
127 %Der er et skarp overgang mellem de 2
128
129 figure(2) %Plot af varmekoefficienten
130 subplot(2,1,1)
131 imagesc(dVec,TempGrid,k) %farven er 3. dim.
132 colorbar
133 title('Varmekoefficient')
134 xlabel('Dybde')
135 ylabel('Temperatur')
136
137 subplot(2,1,2) %Plot af Energihøsten
138 imagesc(dVec,TempGrid, EnergiH)
139 colorbar
140 title('Energihøst')
```

```
141 xlabel('Dybde')
142 ylabel('Temperatur')
143
144 %% Info til plot af når Tamb=TT
145 tt = linspace(TempGrid(1),TempGrid(end),1000);
146 dd = linspace(dVec(1),dVec(end),1000);
147 at = ambientTemp(dd,konstanter);
148
149 tdiff = (repmat(tt,1000,1)-repmat(at',1,1000)).^2;
150 [~,idx] = min(tdiff);
151 ttdyb = dd(idx);
152
153 %% Optimal adfærd ud fra statespace ved baglæns iterationer
154
155 %figure(3)
156
157 %Slut tilstand. På dette tidspunkt er det ikke længere
        muligt at have en
158 %fremtidig høste, det er ved solnedgang.
159 V = zeros(nTemp, nDyb);
160
161 %for i=1:1000
162 maxIt=1e6;
163 Q=ones(1,maxIt)*Inf; %Det skal være en matrix, med inf
164 i=1;
165
166 tic %tids tæller
167 while Q(i) >1e-14 && i<maxIt
168
169     Vgam=V;
170
171     %Værdien, V, opdateres ud fra kulde eller varme.
172     %Upwind discretization anvendes - forklar.
173     V = V + max(0,k)*tidS.*(V(varme,:)-V)/DeltaTemp - min(0,
            k)*tidS.*(V(kulde,:)-V)/DeltaTemp;
174     %k*(Ta-T) * forskellen i fitness fra et step til det
            næste. Første led
175     %i formlens max udtryk.
176
177
178     %Vælg nu den optimale adfærd. Dyk, svøm op og jag
179     %For hvert time step bevæges en grid cell når der
            svømmes eller dykkes
180
181     %V=max(max(V+EnergiH*tidS,V(:,Op)),V(:,Ned)); %
            oprindelig
```

```
182     %V=V+tidS*max(max(EnergiH,(V(:,Op)-V)/DeltaDyb.*Usvom),(
            V(:,Ned)-V)/Del
183     %taDyb.*Usvom); Oprindelig
184     V=V+EnergiH.*tidS + tidS*max(max(0,(V(:,Op)-V)/DeltaDyb
            .*Usvom),(V(:,Ned)-V)/DeltaDyb.*Usvom);
185
186     % Den optimale adfærd findes ved at se hvilket udtryk
            der er størst.
187     % V(:,Ned) alle søjler som Ned siger.
188     % I det nye udtryk kan den spise samtidigt med at den
            svømmer
189
190 %
191 %     if(mod(i,1000)==0) %Hvor mange iterationer der skal
        laves før den plotter
192 %
193 %         subplot(1,2,1)
194 %         imagesc(TempGrid,dVec,V'); %v'=ombytning af rækker
         og søjler
195 %         title('Energihøst')
196 %         xlabel('Tunens temperatur')
197 %         ylabel('Dybde')
198 %         %colorbar
199 %
200 %         spis=max(V(:,Op),V(:,Ned)) < V+EnergiH./DeltaDyb*
        tidS;
201 %         %spis=max(V(:,Op),V(:,Ned)) < V+EnergiH*tidS;
        Oprindelig
202 %         %Giver 1 eller 0, alt afhængigt af om det er sandt
         eller falskt.
203 %         %Hvis det ikke giver bedre "fitness" at ændre
        dybde så spis.
204 %
205 %         SvomOp= V(:,Op) > V(:,Ned);
206 %         %Forskellen mellem nu og næste step er negativ.
        Den taber
207 %         %"fitness", svøm derfor op. Hvis sandt, er der 1
        taller ellers
208 %         % nul og dermed svøm ned.
209 %
210 %         subplot(1,2,2)
211 %         imagesc(TempGrid,dVec,(spis+(1-spis).*(SvomOp+2))
        ');
212 %         % Hvis spis er værdien=1, hvis svøm op er værdien
        =3, hvis svøm ned
213 %         % er værdien =2.
214 %         hold on;
```

```
215 %          plot(tt,ttdyb,'-w');
216 %          hold off;
217 %          title('Optimal adfærd');
218 %          xlabel('Tunens temperatur')
219 %          ylabel('Dybde')
220 %          %farve=colorbar;
221 %          %set(farve,'visible','off')
222 %          pause(1) %Tid til plot inden der plottes igen
223 %     end
224      i=i+1;
225      %Q(i)=norm(V-Vgam)/norm(V);
226      deltaV=V-Vgam;
227      Q(i)=max(deltaV(:)) - min(deltaV(:));
228
229 end
230 toc %Tidstæller
231 gennemV=mean(deltaV(:))/tidS;
232 %% Plot af den optimale adfærd
233
234          figure(3)
235
236          subplot(1,2,1)
237          imagesc(TempGrid,dVec,V'); %v'=ombytning af rækker
                 og søjler
238          title('Energihøst')
239          xlabel('Tunens temperatur')
240          ylabel('Dybde')
241          %colorbar
242
243          spis=max(V(:,Op),V(:,Ned)) < V+EnergiH./DeltaDyb*
                 tidS;
244          %spis=max(V(:,Op),V(:,Ned)) < V+EnergiH*tidS;
                 Oprindelig
245          %Giver 1 eller 0, alt afhængigt af om det er sandt
                 eller falskt.
246          %Hvis det ikke giver bedre "fitness" at ændre dybde
                 så spis.
247
248          SvomOp= V(:,Op) > V(:,Ned);
249          %Forskellen mellem nu og næste step er negativ. Den
                 taber
250          %"fitness", svøm derfor op. Hvis sandt, er der 1
                 taller ellers
251          % nul og dermed svøm ned.
252
253          subplot(1,2,2)
254          imagesc(TempGrid,dVec,(spis+(1-spis).*(SvomOp+2))');
```

```
255         % Hvis spis er værdien=1, hvis svøm op er værdien
                =3, hvis svøm ned
256         % er værdien =2.
257         hold on;
258         plot(tt,ttdyb,'-w');
259         hold off;
260         title('Optimal adfærd');
261         xlabel('Tunens temperatur')
262         ylabel('Dybde')
263         %farve=colorbar;
264         %set(farve,'visible','off')
265         %gemFigur('2111OptimalAd')
266
267 %Plot af konvergerings variablen
268 %Plot de værdier der er forskellige fra inf
269 figure
270 Q_ikke_inf = Q(Q~=inf);
271 semilogy(1:i,[1,Q_ikke_inf]) %Logaritmisk akse
272 xlabel('Antal iterationer')
273 ylabel('Logaritmisk akse')
274 legend('norm(V-Vgam)/norm(V)')
275 %gemFigur('2111KonValue')
276
277
278 %% Simulering af tunens bevægelse
279
280 D0 = konstanter.D0;
281 T0 = konstanter.T0;
282
283
284 AntalSim = 20000; %Antal tidsskridt
285
286 Dsim = zeros(AntalSim,1); %Vektorer
287 Tsim = zeros(AntalSim,1);
288
289 Dsim(1) = D0; %Først punkt sættes til startværdier
290 Tsim(1) = T0;
291
292 [DVEC,TEMPGRID] = meshgrid(dVec,TempGrid);
293
294 %Overall
295 dVdZ=(V(:,Ned)-V(:,Op))./(DeltaDyb*2);
296
297 %Overgangen mellem at spise og svømme udglattes
298 mu = -1+(2./(1+exp(-dVdZ.*10)));
299
300 %figure
```

```
301 %plot(dVdZ,mu)
302
303 for i=1:(AntalSim-1)
304
305 k1 = kTemp(Tsim(i),Dsim(i), konstanter);
306 dTdtsim= k1.*(ambientTemp(Dsim(i),konstanter)-Tsim(i));
307 Tsim(i+1) = Tsim(i) + dTdtsim * tidS;
308
309 hastighedSim = svomHastighed(Tsim(i), konstanter);
310
311
312 dVdZsim=interp2(DVEC, TEMPGRID, dVdZ, Dsim(i), Tsim(i), '
    cubic',0);
313
314 Dsim(i+1) = Dsim(i)+tidS*hastighedSim*(-1+(2./(1+exp(-
    dVdZsim.*10))));
315 Dsim(i+1) = max(0,Dsim(i+1)); %sikre at tunen bliver inden
    for området.
316 Dsim(i+1) = min(H,Dsim(i+1));
317 end
318
319 % [DVEC,TEMPGRID] = meshgrid(dVec,TempGrid);
320 % for i=1:(AntalSim-1)
321 %
322 %   %dTdtsim = interp2(dVec,TempGrid,k,Dsim(i)*(1-1e-6),Tsim
    (i));
323 %   k1 = kTemp(Tsim(i),Dsim(i), konstanter);
324 %   dTdtsim= k1.*(ambientTemp(Dsim(i),konstanter)-Tsim(i));
325 %   Tsim(i+1) = Tsim(i) + dTdtsim * tidS;
326 %
327 %   SpisSim = interp2(DVEC,TEMPGRID,spis,Dsim(i),Tsim(i),'
    nearest',0);
328 %   SvomOpSim = interp2(DVEC,TEMPGRID,SvomOp,Dsim(i),Tsim(i)
    ,'nearest',0);
329 %
330 %
331 %   hastighedSim = svomHastighed(Tsim(i), konstanter);
332 %
333 %   if ~SpisSim,
334 %     if SvomOpSim,
335 %       Dsim(i+1) = max(0,Dsim(i) - DeltaDyb*hastighedSim);
336 %     else
337 %       Dsim(i+1) = Dsim(i) + DeltaDyb*hastighedSim;
338 %     end
339 %   else
340 %     Dsim(i+1) = Dsim(i);
341 %   end
```

```
%
% end

figure

subplot(2,1,1)
plot(-Dsim)
axis([0 AntalSim -500 50])
xlabel('Tidsskridt')
ylabel('Dybde')

subplot(2,1,2)
plot(Tsim,'r')
axis([0 AntalSim 0 30])
hold on
plot(interp1(dVec,TempVec,Dsim*0.999))
hold off
xlabel('Tidsskridt')
legend('Tunens temperatur','Ambiente temperatur')
%
```

```
%-------------------------------------------------------------------------

%%Script til at lave plot over hvordan fordelingen er mellem
     at svømme,
%spise og varme.
%Der plottes for forskellige størrelser tun og for
    forskellige værdier af
%z_ss.
%-------------------------------------------------------------------------


%%Konstanter

%Temperatur udviklingen i dybden
konstanter.Tmax = 27.5;  %overflade temperatur i grader
konstanter.p = -1/100; % temperaturfaldet efter thermoklinen
konstanter.A = 17; % Temperaturfaldet over thermoklinen
konstanter.Lambda = 2; % overgangen fra thermoklinen til
    temperatur faldet i dybdem
konstanter.zTcli = 100; % slut dybden på thermoklinen

%Svømmehastighed ud fra temperatur i konstantDybde
konstanter.Vmin=(2.0*70)/100;
konstanter.Vmax = (2.3*70)/100;

%Værdier for fødetætheden
```

```
konstanter.f0=0; %fødetætheden ved overfladen
konstanter.fSS=1; %
konstanter.zSS=500; %dybden hvor der er mest føde
konstanter.q=50; %længden af overgangszonen org. 100

%Værdier til kTemp i varDyb
konstanter.klow=0.03/60; %pr minut hos Malte så del med 60
konstanter.khigh = 0.3/60; %pr minut hos Malte så del med 60
konstanter.Tbb = 0.2; % org. 0.7. forskydning så tunen
    bliver afkølet lidt, inden der skiftes til khigh


%Periode
konstanter.Tp = (20*60);  %Én periode er 20 minutter

%Omega og epsilon
konstanter.omega = (2*pi)/konstanter.Tp; %Vinkel frekvens

%Start betingelser
konstanter.D0 = 300; %ud fra Olivier s. 125.
konstanter.T0 = 15; %ud fra Olivier s. 125.
konstanter.E0 = 0; % energien den har til at starte med er
    nul.

konstanter.TidI = konstanter.Tp*10; % skal køres i et helt
    antal perioder

%Konstanter til Phi=svingPhi i varDybPhi
%Den rette linje ud fra de to punkter
konstanter.Tlow = 17.5;
konstanter.Thigh = 22.5;
konstanter.Dlinje = 500;

konstanter.U0=1.5;

%Tunens størrelse

%%Tunens udgifter
%Svømme effekt
konstanter.beta=2;
konstanter.alpha=0;
%Varmetab
konstanter.c=0; %varmekapacitet

%%
    ----------------------------------------------------------------
```

```
64  %%Tids fordeling for 1 dag for 1 tun ved en given z_ss.
65  %%-----------------------------------------------------------------------
66
67  vD=20; %Antal størrelser
68  zSizes = linspace(70,700,vD); %Interval som zSS skal være
        imellem
69
70  svom = zeros(vD,1);
71  spis = zeros(vD,1);
72  varm = zeros(vD,1);
73
74  nDyb=50;
75  nTemp=45;
76  Op=[1,1:(nDyb-1)];
77  Ned=[2:nDyb,nDyb];
78
79  H=konstanter.zSS+5*konstanter.q;
80  dVec=linspace(0,H,nDyb);
81  DeltaDyb = dVec(2)-dVec(1);
82
83
84  for i = 1 : vD;
85      zSize = zSizes(i);
86
87      [minSim3, maxSim3, gennemV3, Dsim3, Tsim3,Vmad3, MaxDk3]
            = bifurkationMad(zSize, konstanter);
88      lengD=length(Dsim3);
89
90      s = abs(diff(Dsim3([1,1:end])));
91      swim = s>1e-2;
92      still = ~swim;
93
94      eat = still & Dsim3>(maxSim3+minSim3)*0.5;
95      warmth = still & ~eat;
96
97
98      svom(i) = nnz(swim)/lengD;
99      spis(i) = nnz(eat)/lengD;
100     varm(i) = nnz(warmth)/lengD;
101
102
103
104
105 %      s = abs(sign(diff(Dsim3)));
106 %      swim = length(find(s>0));
107 %      swimP = swim/lengD*100;
```

```
%
%     %De dybder den ligger stille ved:
%     dFsim=diff(Dsim3);
%     jj=find(dFsim==0);
%     still = Dsim(jj);
%
%     %Grænsen hvor den varmer istedet for at jage:
%     % det må være relativt til hvor zSS er henne.
%
%     bound = 100;
%
%     %Spis
%     eat = length(find(sign(still-bound)>0));
%     eatP = eat/lengD*100;
%
%     %Varme
%     v1 = length(find(sign(still-bound)<0));
%     v2 = length(find(sign(still-bound)==0));
%     warmth =v1+v2;
%     warmthP = warmth/lengD*100;
%
%     svom(i) = swimP;
%     spis(i) = eatP;
%     varm(i) = warmthP;

end

%% Plot af fordelingerne for hver z_ss

A = [svom spis varm];
x = zSizes;

figure
harea = area(x,A);
set(harea(1),'FaceColor',[.5 .8 .9])
set(harea(2),'FaceColor',[.7 .9 .1])
set(harea(3),'FaceColor',[.9 1 1])
xlabel('z_{ss}')
ylabel('%')
legend('Svøm', 'Spis', 'Varm')
%gemFigur('BarZss')
%savefig('../Skabelon/figurer/BarZss2',gcf,'pdf')


% http://www.mathworks.se/help/matlab/creating_plots/bar-and
    -area-graphs.ht
% ml
```

```
154 %http://www.mathworks.se/help/matlab/creating_plots/bar-and-
        area-graphs.htm
155 %l
156
157 %%
        ------------------------------------------------------------------

158 %%Tids fordeling for 1 dag for 1 tun ved en given længde.
159 %%-----------------------------------------------------------------

160
161 nSize=22; %Antal størrelser, før var det 15.
162 lSizes = linspace(0.25,2.0,nSize); %Oprindeligt 0.1-1.5
163
164 svomL = zeros(nSize,1);
165 spisL = zeros(nSize,1);
166 varmL = zeros(nSize,1);
167
168 for i=1:nSize
169     lSize=lSizes(i);
170
171     [minSim4, maxSim4, gennemV4, Dsim4, Tsim4, Vsize4] =
            bifurkationSize(lSize, konstanter);
172
173     lengLD=length(Dsim4);
174
175     s = abs(diff(Dsim4([1,1:end])));
176     swim = s>1e-2;
177     still = ~swim;
178
179     eat = still & Dsim4>(maxSim4+minSim4)*0.5;
180     warmth = still & ~eat;
181
182
183     svomL(i) = nnz(swim)/lengLD;
184     spisL(i) = nnz(eat)/lengLD;
185     varmL(i) = nnz(warmth)/lengLD;
186
187
188
189
190
191 %     %Det antal gange den svømmer:
192 %     sL = abs(sign(diff(Dsim4)));
193 %     swimL = length(find(sL>0));
194 %     swimLP = swimL/lengLD*100;
195 %
```

```
%      %De dybder den ligger stille ved:
%      dFsimL=diff(Dsim4);
%      jjL=find(dFsimL==0);
%      stillL = Dsim(jjL);
%
%      %Grænsen hvor den varmer istedet for at jage:
%      bound = 100;
%
%      %Spis
%      eatL = length(find(sign(stillL-bound)>0));
%      eatLP = eatL/lengLD*100;
%
%      %Varme
%      v1L = length(find(sign(stillL-bound)<0));
%      v2L = length(find(sign(stillL-bound)==0));
%      warmthL =v1L+v2L;
%      warmthLP = warmthL/lengLD*100;
%
%      svomL(i) = swimLP;
%      spisL(i) = eatLP;
%      varmL(i) = warmthLP;
end

%% Plot af fordelingerne for hver længde

B = [svomL spisL varmL];
x = lSizes;

figure
harea = area(x,B);
set(harea(1),'FaceColor',[.5 .8 .9])
set(harea(2),'FaceColor',[.7 .9 .1])
set(harea(3),'FaceColor',[.9 1 1])
xlabel('Tunens længde')
ylabel('%')
legend('Svøm', 'Spis', 'Varm')
%gemFigur('BarLength2')
%savefig('../Skabelon/figurer/BarLength3',gcf,'pdf')
```

```
%-------------------------------------------------------------------------

%%Script til at sammenligne løsningen til konstant dybde med
     løsningen
%lige omkring bifurkationspunktet z_^b
%
%-------------------------------------------------------------------------
```

```
6
7  %%Konstanter
8
9  %Temperatur udviklingen i dybden
10 konstanter.Tmax = 27.5;  %overflade temperatur i grader
11 konstanter.p = -1/100; % temperaturfaldet efter thermoklinen
12 konstanter.A = 17; % Temperaturfaldet over thermoklinen
13 konstanter.Lambda = 2; % overgangen fra thermoklinen til
       temperatur faldet i dybdem
14 konstanter.zTcli = 100; % slut dybden på thermoklinen
15
16 %Svømmehastighed ud fra temperatur i konstantDybde
17 konstanter.Vmin=(2.0*70)/100;
18 konstanter.Vmax = (2.3*70)/100;
19
20 %Værdier for fødetætheden
21 konstanter.f0=0; %fødetætheden ved overfladen
22 konstanter.fSS=1; %
23 konstanter.zSS=500; %dybden hvor der er mest føde
24 konstanter.q=100; %længden af overgangszonen org. 100
25
26 %Værdier til kTemp i varDyb
27 konstanter.klow=0.03/60; %pr minut hos Malte så del med 60
28 konstanter.khigh = 0.3/60; %pr minut hos Malte så del med 60
29 konstanter.Tbb = 0.2; % org. 0.7. forskydning så tunen
       bliver afkølet lidt, inden der skiftes til khigh
30
31
32 %Periode
33 konstanter.Tp = (20*60);  %Én periode er 20 minutter
34
35 %Omega og epsilon
36 konstanter.omega = (2*pi)/konstanter.Tp; %Vinkel frekvens
37
38 %Start betingelser
39 konstanter.D0 = 300; %ud fra Olivier s. 125.
40 konstanter.T0 = 15; %ud fra Olivier s. 125.
41 konstanter.E0 = 0; % energien den har til at starte med er
       nul.
42
43 konstanter.TidI = konstanter.Tp*10; % skal køres i et helt
       antal perioder
44
45 %Konstanter til Phi=svingPhi i varDybPhi
46 %Den rette linje ud fra de to punkter
47 konstanter.Tlow = 17.5;
48 konstanter.Thigh = 22.5;
```

```
49 konstanter.Dlinje = 500;
50
51 konstanter.U0=1.5;
52
53 %Tunens størrelse
54
55 %%Tunens udgifter
56 %Svømme effekt
57 konstanter.beta=2;
58 konstanter.alpha=0;
59 %Varmetab
60 konstanter.c=0; %varmekapacitet
61
62 %%
       ----------------------------------------------------------------
63 %Når z_ss er større end z^b
64 %%---------------------------------------------------------------
65
66 aZ=5;
67 zSizes=linspace(61,75,aZ); %Nye værdier for længden
68 plotZd=zeros(aZ,20000);
69 plotZt=zeros(aZ,20000);
70
71
72 %Tunens optimale dybde ved konstant dybde
73 [ z,Temp2, U, f ,E, MaxE, MaxD ] = konstantDybde(konstanter)
       ;
74
75 %Vand temperaturen til den givne dybde
76 Temp=ambientTemp(MaxD,konstanter);
77
78
79 %Simulering af tunens bevægelse
80
81 %Simulering af tunens bevægelse
82
83 for i=1:aZ
84     konstanter.zSS=zSizes(i);
85    [minSim4, maxSim4, gennemV4, Dsim4, Tsim4,Vmad4, MaxDk4]
          = bifurkationMad(konstanter.zSS, konstanter);
86
87     plotZd(i,:)=Dsim4;
88     plotZt(i,:)=Tsim4;
89 end
90
```

```
91  %% Temperaturkurven
92  zz=linspace(0,konstanter.zSS);
93  TempKurve=ambientTemp(zz,konstanter);
94
95  %%Plot
96  figure
97  hold on
98  plot(TempKurve, zz,'k', 'linewidth',1.5)
99  plot(plotZt(1,:),plotZd(1,:), 'Color', 'b', 'linewidth',1.5)
100 plot(plotZt(2,:),plotZd(2,:),'Color', 'r','linewidth',1.5)
101 plot(plotZt(3,:),plotZd(3,:),'Color', 'm','linewidth',1.5)
102 plot(plotZt(4,:),plotZd(4,:),'Color', 'c','linewidth',1.5)
103 plot(plotZt(5,:),plotZd(5,:),'Color', 'g','linewidth',1.5)
104 %
105 %plot(plotLt(1,end),plotLd(1,end),'bX', 'linewidth',2.5)
106 %plot(plotLt(2,end),plotLd(2,end),'rX', 'linewidth',2.5)
107 %plot(plotLt(3,end),plotLd(3,end),'mX', 'linewidth',2.5)
108 %plot(plotLt(4,end),plotLd(4,end),'cX', 'linewidth',2.5)
109 %plot(plotLt(5,end),plotLd(5,end),'gX', 'linewidth',2.5)
110 %plot(Temp,MaxD, 'rX', 'linewidth',2.5)
111 axis ij
112 xlabel('Temperatur')
113 ylabel('Dybde')
114 legend('T_a', ['z_{ss} = ' num2str(zSizes(1))],['z_{ss} = '
        num2str(zSizes(2))], ['z_{ss} = ' num2str(zSizes(3))], ['
        z_{ss} = ' num2str(zSizes(4))], ['z_{ss} = ' num2str(
        zSizes(5))],'Location','NorthWest' )
115 title('z_{ss} større end bifurkations punktet, z_b')
116 %savefig('../Skabelon/figurer/adfardZssM9',gcf,'pdf')
117
118 %%
        ------------------------------------------------------------------

119 %Når z_ss er mindre end z^b
120 %%-----------------------------------------------------------------------

121
122 adZ=45;             %Ny værdi for z_ss
123
124 konstanter.zSS=adZ; %Den gamle sættes lig den nye
125
126
127 %Tunens optimale dybde ved konstant dybde
128 [ z,Temp2, U, f ,E, MaxE, MaxD ] = konstantDybde(konstanter)
        ;
129
130 %Vand temperaturen til den givne dybde
```

```
131 Temp=ambientTemp(MaxD,konstanter);
132
133 %Simulering af tunens bevægelse
134 [minSim2, maxSim2, gennemV2, Dsim2, Tsim2,Vmad2, MaxDk2] =
        bifurkationMad(konstanter.zSS, konstanter);
135 %[minSim2, maxSim2, gennemV2, Dsim2, Tsim2,Vmad2, middel2,
        afvig2, MaxDk2] = bifurkationMad(konstanter.zSS,
        konstanter);
136
137 %Find startværdien, lidt væk fra Temp punktet
138
139 %Sorter det første ballade mht startpunktet fra.
140 dFsim=diff(Dsim2);
141 %jj=find(dFsim==0);
142 %jj=jj(1);
143
144 %sorDsim=Dsim2(jj:end); %sorteret fra
145 %sorTsim=Tsim2(jj:end);
146 %iq=find(sorDsim<MaxD+2); %Værdier lidt fra maximum, starten
        på simuleringen
147
148 % Den del af simuleringen der skal plottes
149 PlotD=Dsim2(1:end);
150 PlotT=Tsim2(1:end);
151 %PlotD=Dsim2(iq(1):end);
152 %PlotT=Tsim2(iq(1):end);
153 %PlotD=Dsim2(iq(1):(iq(1)+5500));
154 %PlotT=Tsim2(iq(1):(iq(1)+5500));
155
156
157 %Temperaturkurven
158 zz=linspace(0,adZ+10);
159 TempKurve=ambientTemp(zz,konstanter);
160
161
162 %% Plot
163 figure
164 hold on
165 plot(TempKurve, zz,'k', 'linewidth',1.5)
166 plot(PlotT,PlotD, 'linewidth',1.5)
167 plot(Temp,MaxD, 'rX', 'linewidth',2.5)
168 plot(Tsim2(end),Dsim2(end),'mX', 'linewidth',2.5)
169 axis ij
170 xlabel('Temperatur')
171 ylabel('Dybde')
172 legend('T_a','Simulering af tunens adfærd','Optimal dybde
        ved konstant dybde','Slut punkt for simuleringen', '
```

```
    Location','NorthWest')
title('z_{ss} mindre end bifurkations punktet, z_b')
%gemFigur('adfardZssB7') %tidligere B1 og B3
%gemFigur('adfardZssN1')
%savefig('../Skabelon/figurer/adfardZssN2',gcf,'pdf')

%%
nDyb=50;
nTemp=45;

H=konstanter.zSS+5*konstanter.q;
dVec=linspace(0,H,nDyb);

TempVec=ambientTemp(dVec, konstanter);
TempGrid=linspace(TempVec(end),TempVec(1),nTemp);

figure
contour(TempGrid,dVec,Vmad2');
hold on
plot(ambientTemp(MaxD, konstanter),MaxD,'rx', 'linewidth
    ',2.0)
plot(Tsim2(end),Dsim2(end),'mx','linewidth',2.0)

axis ij
xlabel('Temperatur')
ylabel('Dybde')
colorbar
legend('V_{\infty}','Optimal dybde ved konstant dybde','
    Slutpunkt for simuleringen', 'Location', 'SouthWest')
%savefig('../Skabelon/figurer/Vkurvekk',gcf,'pdf')
%savefig('../Skabelon/figurer/Vkurvekk2',gcf,'pdf')

%%
    ------------------------------------------------------------------

%Når længden af tunen er mindre end s^b
%%-----------------------------------------------------------------



lSize=0.15; %Ny værdi for længden


%Tunens optimale dybde ved konstant dybde
[ z,Temp2, U, f ,E, MaxE, MaxD ] = konstantDybde(konstanter)
    ;

```

```
213 %Vand temperaturen til den givne dybde
214 Temp=ambientTemp(MaxD,konstanter);
215
216 %Simulering af tunens bevægelse
217 [minSim, maxSim, gennemV3, Dsim3, Tsim3, Vsize] =
        bifurkationSize(lSize, konstanter);
218
219
220 PlotD=Dsim3(1:end);
221 PlotT=Tsim3(1:end);
222
223 %Temperaturkurv en
224 zz=linspace(0,600);
225 TempKurve=ambientTemp(zz,konstanter);
226
227
228 %%Plot
229 figure
230 hold on
231 plot(TempKurve, zz,'k', 'linewidth',1.5)
232 plot(PlotT,PlotD, 'linewidth',1.5)
233 plot(Temp,MaxD, 'rX', 'linewidth',2.5)
234 plot(Tsim3(end),Dsim3(end),'mX', 'linewidth',2.5)
235 axis ij
236 xlabel('Temperatur')
237 ylabel('Dybde')
238 legend('T_a','Simulering af tunens adfærd','Optimal dybde
        ved konstant dybde','Slut punkt for simuleringen', '
        Location','NorthWest')
239 title('L_{b} mindre end bifurkations punktet, s_b')
240 %savefig('../Skabelon/figurer/adfardLbM',gcf,'pdf')
241
242 %%
        ------------------------------------------------------------------

243 %Når længden af tunen er større end s^b
244 %%-----------------------------------------------------------------

245
246 aDyb=5;
247 lSizes=linspace(0.23,3.7,aDyb); %Nye værdier for længden
248 plotLd=zeros(aDyb,20000);
249 plotLt=zeros(aDyb,20000);
250
251 %%Tunens optimale dybde ved konstant dybde
252 [ z,Temp2, U, f ,E, MaxE, MaxD ] = konstantDybde(konstanter)
        ;
```

```
253
254 %Vand temperaturen til den givne dybde
255 Temp=ambientTemp(MaxD,konstanter);
256
257 %Simulering af tunens bevægelse
258
259 for i=1:aDyb
260     lSize=lSizes(i);
261
262     [minSim, maxSim, gennemV3, Dsim3, Tsim3, Vsize] =
            bifurkationSize(lSize, konstanter);
263
264     plotLd(i,:)=Dsim3;
265     plotLt(i,:)=Tsim3;
266
267 end
268
269 %Temperaturkurven
270 zz=linspace(0,600);
271 TempKurve=ambientTemp(zz,konstanter);
272
273 %%Plot
274 figure
275 hold on
276 plot(TempKurve, zz,'k', 'linewidth',1.5)
277 plot(plotLt(1,:),plotLd(1,:), 'Color', 'b', 'linewidth',1.5)
278 plot(plotLt(2,:),plotLd(2,:),'Color', 'r','linewidth',1.5)
279 plot(plotLt(3,:),plotLd(3,:),'Color', 'm','linewidth',1.5)
280 plot(plotLt(4,:),plotLd(4,:),'Color', 'c','linewidth',1.5)
281 plot(plotLt(5,:),plotLd(5,:),'Color', 'g','linewidth',1.5)
282 %
283 %plot(plotLt(1,end),plotLd(1,end),'bX', 'linewidth',2.5)
284 %plot(plotLt(2,end),plotLd(2,end),'rX', 'linewidth',2.5)
285 %plot(plotLt(3,end),plotLd(3,end),'mX', 'linewidth',2.5)
286 %plot(plotLt(4,end),plotLd(4,end),'cX', 'linewidth',2.5)
287 %plot(plotLt(5,end),plotLd(5,end),'gX', 'linewidth',2.5)
288 %plot(Temp,MaxD, 'rX', 'linewidth',2.5)
289 axis ij
290 xlabel('Temperatur')
291 ylabel('Dybde')
292 legend('T_a', ['L_b = ' num2str(lSizes(1))],['L_b = '
        num2str(lSizes(2))], ['L_b = ' num2str(lSizes(3))], ['L_b
         = ' num2str(lSizes(4))], ['L_b = ' num2str(lSizes(5))],'
        Location','NorthWest')
293 title('L_{b} større end bifurkations punktet, s_b')
294 %savefig('../Skabelon/figurer/adfardLbS2',gcf,'pdf') %
        gammelt plot: adfardLbS
```

```
295
296 %0.2300    1.0975    1.9650    2.8325    3.7000
297
298 %%
299 PlotD=Dsim3(1:end);
300 PlotT=Tsim3(1:end);
301
302 %Temperaturkurv en
303 zz=linspace(0,600);
304 TempKurve=ambientTemp(zz,konstanter);
305
306
307 %%Plot
308 figure
309 hold on
310 plot(TempKurve, zz,'k', 'linewidth',1.5)
311 plot(PlotT,PlotD, 'linewidth',1.5)
312 plot(Temp,MaxD, 'rX', 'linewidth',2.5)
313 plot(Tsim3(end),Dsim3(end),'mX', 'linewidth',2.5)
314 axis ij
315 xlabel('Temperatur')
316 ylabel('Dybde')
317 legend('Ambient temperatur','Simulering af tunens adfærd','
    Optimal dybde ved konstant dybde','Slut punkt for
    simuleringen', 'Location','NorthWest')
318 title('L_{b} større end bifurkations punktet, s_b')
319 %savefig('../Skabelon/figurer/adfardLbS',gcf,'pdf')
```

```
1  %%Script til at opstille og løse det dynamiske udtryk for
       tunens adfærd.
2
3  clc, clear, %close all
4
5  %% Konstanter
6
7
8  %Temperatur udviklingen i dybden
9  konstanter.Tmax = 27.5;  %overflade temperatur i grader
10 konstanter.p = -1/100; % temperaturfaldet efter thermoklinen
11 konstanter.A = 17; % Temperaturfaldet over thermoklinen
12 konstanter.Lambda = 2; % overgangen fra thermoklinen til
       temperatur faldet i dybdem
13 konstanter.zTcli = 100; % slut dybden på thermoklinen
14
15 %Svømmehastighed ud fra temperatur i konstantDybde
16 konstanter.Vmin=(2.0*70)/100;
17 konstanter.Vmax = (2.3*70)/100;
```

```
18
19 %Værdier for fødetætheden
20 konstanter.f0=0; %fødetætheden ved overfladen
21 konstanter.fSS=1; %
22 konstanter.zSS=45; %dybden hvor der er mest føde
23 konstanter.q=100; %længden af overgangszonen org. 100
24
25 %Værdier til kTemp i varDyb
26 konstanter.klow=0.03/60; %pr minut hos Malte så del med 60
27 %Sensitivitets analyse: ændre klow til:
28 %konstanter.klow=0.03/60-(0.03/60)/100*5;
29 %konstanter.klow=0.03/60*10;
30 konstanter.khigh = 0.3/60; %pr minut hos Malte så del med 60
31 %konstanter.khigh = 0.3/60*10;
32 konstanter.Tbb = 0.2; % org. 0.7. forskydning så tunen
       bliver afkølet lidt, inden der skiftes til khigh
33
34
35 %Periode
36 konstanter.Tp = (20*60);  %Én periode er 20 minutter
37
38 %Omega og epsilon
39 konstanter.omega = (2*pi)/konstanter.Tp; %Vinkel frekvens
40
41 %Start betingelser
42 konstanter.D0 = 300; %ud fra Olivier s. 125.
43 konstanter.T0 = 15; %ud fra Olivier s. 125.
44 konstanter.E0 = 0; % energien den har til at starte med er
       nul.
45
46 konstanter.TidI = konstanter.Tp*10; % skal køres i et helt
       antal perioder
47
48 %Konstanter til Phi=svingPhi i varDybPhi
49 %Den rette linje ud fra de to punkter
50 konstanter.Tlow = 17.5;
51 konstanter.Thigh = 22.5;
52 konstanter.Dlinje = 500;
53
54 konstanter.U0=1.5;
55
56 %% Model funktioner, grids mm:
57
58 %Grids
59 nDyb=50;
60 nTemp=45;
61
```

```
62 %Skift: (indeksering)
63 Op=[1,1:(nDyb-1)]; %Rækkevektor med dobbelt start værdi
64 Ned=[2:nDyb,nDyb]; % Rækkevektor med dobbelt slut værdi
65
66 varme = [2:nTemp,nTemp]; %Rækkevektor med dobbelt slut
       nummer
67 kulde = [1,1:(nTemp-1)]; %Rækkevektor med dobbelt start
       nummer
68
69 %Tidsskridt
70 tidS=1;
71
72 %H=500;
73 H=konstanter.zSS+5*konstanter.q;
74 dVec=linspace(0,H,nDyb); %50 punkter fordelt lige over de
       500 meters dybde
75 MadVec=fodetaet(dVec, konstanter); %Fødetætheden i hvert
       dybdepunkt
76 MadVec2 = 1./(1+exp(-(dVec/H-0.85)*10));
77 TempVec=ambientTemp(dVec, konstanter); % vandtemp. i hvert
       dybdepunkt
78
79 %Plot af Temperatur og fødetæthed afhængigt af dybden
80 figure(1)
81 plot(MadVec,dVec,'r')
82 hold on
83 plot(TempVec, dVec)
84 plot(MadVec2, dVec, 'g')
85 hold off
86 axis ij
87 ylabel('Dybde')
88 legend('Fødetæthed','Temperatur', 'Uffes fødetæthed')
89
90 TempGrid=linspace(TempVec(end),TempVec(1),nTemp);
91 %Temperaturen stiger igennem vektoren
92
93 DeltaDyb = dVec(2)-dVec(1); %Forskellen mellem 2 dybde
       punkter.
94 DeltaTemp= TempGrid(2)-TempGrid(1); %Forskellen mellem 2
       temp punkter
95
96 %% State space
97 [DD,TT]=meshgrid(dVec,TempGrid);
98 %DD: Søjle 1 indeholder 45 rækker med værdien dVec(1),
99 %TT: Række 1 indeholder 50 kolonner med værdien TempGrid(1).
100
101 %% Temperatur-, energi- og fødefunktioner i state space
```

```
    punkterne
Tamb=meshgrid(TempVec,TempGrid);
%Hver kolonne k indeholder 45 kopier af TempVec(k).
FodeS=meshgrid(MadVec,TempGrid);
%Hver kolonne k indeholder 45 kopier af MadVec(k).

Usvom=svomHastighed(TT, konstanter);
%Svømmehastigheden til den ambiente temp.
EnergiH=FodeS.*Usvom;
%Energihøst i statespace punkterne


%Varmekoefficienten evalueret i statespace og der ses på
    temp forskellen
k1 = kTemp(TT,DD, konstanter);
k= k1.*(Tamb-TT);


figure(2) %Plot af varmekoefficienten
subplot(2,1,1)
imagesc(dVec,TempGrid,k) %farven er 3. dim.
colorbar
title('Varmekoefficient')
xlabel('Dybde')
ylabel('Temperatur')

subplot(2,1,2) %Plot af Energihøsten
imagesc(dVec,TempGrid, EnergiH)
colorbar
title('Energihøst')
xlabel('Dybde')
ylabel('Temperatur')

%% Info til plot af når Tamb=TT
tt = linspace(TempGrid(1),TempGrid(end),1000);
dd = linspace(dVec(1),dVec(end),1000);
at = ambientTemp(dd,konstanter);

tdiff = (repmat(tt,1000,1)-repmat(at',1,1000)).^2;
[~,idx] = min(tdiff);
ttdyb = dd(idx);

%% Optimal adfærd ud fra statespace ved baglæns iterationer

%Slut tilstand. På dette tidspunkt er det ikke længere
    muligt at have en
%fremtidig høste, det er ved solnedgang.
```

```
146 V = zeros(nTemp, nDyb);
147
148 maxIt=2e6;
149 Q=ones(1,maxIt)*Inf;
150 gV= NaN(1,maxIt);
151 nV= NaN(1,maxIt);
152 w=[];
153 W=[];
154
155 i=1;
156
157 tic %tids tæller
158 while Q(i) >1e-14 && i<maxIt
159
160     Vgam=V;
161
162     %Værdien, V, opdateres ud fra kulde eller varme.
163     %Upwind discretization anvendes - forklar.
164     V = V + max(0,k)*tidS.*(V(varme,:)-V)/DeltaTemp - min(0,
            k)*tidS.*(V(kulde,:)-V)/DeltaTemp;
165     %k*(Ta-T) * forskellen i fitness fra et step til det
            næste. Første led
166     %i formlens max udtryk.
167
168
169     %Vælg nu den optimale adfærd. Dyk, svøm op og jag
170     %For hvert time step bevæges en grid cell når der
            svømmes eller dykkes
171     V=V+EnergiH.*tidS + tidS*max(max(0,(V(:,Op)-V)/DeltaDyb
            .*Usvom),(V(:,Ned)-V)/DeltaDyb.*Usvom);
172
173     % Den optimale adfærd findes ved at se hvilket udtryk
            der er størst.
174     % I det nye udtryk kan den spise samtidigt med at den
            svømmer
175
176     i=i+1;
177     %Q(i)=norm(V-Vgam)/norm(V);
178     deltaV=V-Vgam;
179     Q(i)=max(deltaV(:)) - min(deltaV(:));
180
181     %Gem til udregninger af mu
182     nV(i)=norm(deltaV(:));
183     gV(i)=mean(V(:));
184
185     if mod(i,100)==0  %Gem for hver hundrede
186         w = [w;i];
```

```
187         W = [W V(:)];
188         disp(i);
189     end
190
191 end
192 toc %Tidstæller
193
194 %% Udregner mu overall for V, V(D,T,t)=V(T,D)_u + mu*t
195 gennemV=mean(deltaV(:))/tidS;
196
197 nV=nV(~isnan(nV));
198 gV=gV(~isnan(gV));
199 iL=length(gV);
200
201 [P,S]=polyfit(1:iL,gV,1); %P(1) er hældningen
202
203 %% Plot
204 figure
205 subplot(1,2,1)
206 plot(nV)
207 axis tight
208
209 subplot(1,2,2)
210 plot(gV)
211 axis tight
212
213 %% Plot middel ændring for hvert grit punkt i V
214
215 Q1=NaN(nTemp,nDyb);
216 Q2 = NaN(nTemp,nDyb);
217
218 for j=1:nDyb*nTemp
219     [P1,S1] = polyfit(w,W(j,:)',1);
220     Q1(j) = P1(1);   % Hældning
221     Q2(j) = P1(2);   % Konstant
222 end
223 %%
224 figure
225 imagesc(dVec, TempGrid, Q1(end-1:end))
226 axis square;
227 colorbar
228
229 figure
230 imagesc(dVec,TempGrid,Q2(end)); axis square; title('Konstant
      led, V_{\infty}');
231 colorbar
232
```

```
233 %%
234 % Vis billeder
235 figure,
236 imagesc(dVec,TempGrid,Q1); axis square; title('Hældning, \
        gamma');
237 colorbar
238 %savefig('../Skabelon/figurer/gammaLos1',gcf,'pdf')
239 %gemFigur('gammaLos')
240
241 figure
242 %subplot 122,
243 imagesc(dVec,TempGrid,Q2); axis square; title('Konstant led,
         V_{\infty}');
244 colorbar
245 %savefig('../Skabelon/figurer/VuendligLos1',gcf,'pdf')
246 %gemFigur('VuendligLos')
247
248
249 %% Plot af den optimale adfærd
250
251         figure(3)
252
253         subplot(1,2,1)
254         imagesc(TempGrid,dVec,V'); %v'=ombytning af rækker
                og søjler
255         title('Energihøst')
256         xlabel('Tunens temperatur')
257         ylabel('Dybde')
258         %colorbar
259
260         spis=max(V(:,Op),V(:,Ned)) < V+EnergiH./DeltaDyb*
                tidS;
261         %spis=max(V(:,Op),V(:,Ned)) < V+EnergiH*tidS;
                Oprindelig
262         %Giver 1 eller 0, alt afhængigt af om det er sandt
                eller falskt.
263         %Hvis det ikke giver bedre "fitness" at ændre dybde
                så spis.
264
265         SvomOp= V(:,Op) > V(:,Ned);
266         %Forskellen mellem nu og næste step er negativ. Den
                taber
267         %"fitness", svøm derfor op. Hvis sandt, er der 1
                taller ellers
268         % nul og dermed svøm ned.
269
270         subplot(1,2,2)
```

```
271         imagesc(TempGrid,dVec,(spis+(1-spis).*(SvomOp+2))');
272         % Hvis spis er værdien=1, hvis svøm op er værdien
                =3, hvis svøm ned
273         % er værdien =2.
274         hold on;
275         plot(tt,ttdyb,'-w'); %Når Tamb-Tt=0
276         hold off;
277         title('Optimal adfærd');
278         xlabel('Tunens temperatur')
279         ylabel('Dybde')
280         %farve=colorbar;
281         %set(farve,'visible','off')
282         %gemFigur('2111OptimalAd500m')
283 %%
284 %Plot af konvergerings variablen
285 %Plot de værdier der er forskellige fra inf
286 Q_ikke_inf = Q(Q~=inf);
287 figure
288 semilogy(2:i, Q_ikke_inf) %Logaritmisk akse
289 %plot(2:i,Q_ikke_inf);
290 xlabel('Antal iterationer')
291 ylabel('Q(i)')
292 %legend('Q(i)=max(deltaV(:)) - min(deltaV(:));')
293 %title(['Energihøst raten = ' num2str(gennemV)])
294 %gemFigur('1101KonValue500m') %for z_ss lig 500
295 %gemFigur('2911KonValue500m')
296
297
298 %% Simulering af tunens bevægelse
299
300 D0 = konstanter.D0;
301 T0 = konstanter.T0;
302
303
304 AntalSim = 20000; %Antal tidsskridt
305
306 Dsim = zeros(AntalSim,1); %Vektorer
307 Tsim = zeros(AntalSim,1);
308 dVdZsim = zeros(AntalSim,1);
309
310
311 Dsim(1) = D0; %Først punkt sættes til startværdier
312 Tsim(1) = T0;
313 [DVEC,TEMPGRID] = meshgrid(dVec,TempGrid);
314
315 [DVEC,TEMPGRID] = meshgrid(dVec,TempGrid);
316
```

```
317 %Overall
318 dVdZ=(V(:,Ned)-V(:,Op))./(DeltaDyb*2);
319
320 %Overgangen mellem at spise og svømme udglattes
321 mu = -1+(2./(1+exp(-dVdZ.*10)));
322
323 %figure
324 %plot(dVdZ,mu)
325
326 for i=1:(AntalSim-1)
327
328 k1 = kTemp(Tsim(i),Dsim(i), konstanter);
329 dTdtsim= k1.*(ambientTemp(Dsim(i),konstanter)-Tsim(i));
330 Tsim(i+1) = Tsim(i) + dTdtsim * tidS;
331
332 hastighedSim = svomHastighed(Tsim(i), konstanter);
333
334
335 dVdZsim(i)=interp2(DVEC, TEMPGRID, dVdZ, Dsim(i), Tsim(i), '
    cubic',0);
336
337 Dsim(i+1) = Dsim(i)+tidS*hastighedSim*(-1+(2./(1+exp(-
    dVdZsim(i).*100))));
338 Dsim(i+1) = max(0,Dsim(i+1));
339 Dsim(i+1) = min(H,Dsim(i+1));
340 end
341
342 %% Plot af simuleringen
343 figure
344
345 subplot(2,1,1)
346 plot(Dsim)
347 axis ij
348 %set(gca, 'yticklabels',{})
349 axis([0 AntalSim -50 550])%Akse fra 50 til minus 500 på y-
    aksen. X-akse 0:Nsim
350 xlabel('Tidsskridt')
351 ylabel('Dybde')
352
353 subplot(2,1,2)
354 plot(Tsim,'r')
355 axis([0 AntalSim 0 30])
356 hold on
357 plot(ambientTemp(Dsim,konstanter))
358 hold off
359 xlabel('Tidsskridt')
360 legend('Tunens temperatur','Ambient temperatur')
```

```
361 %savefig('../Skabelon/figurer/2301Simulering',gcf,'pdf')
362 %gemFigur('2201Simulering')
363
364 %% Plot af bevægelsen, ændret så det er med den rigtige
        simulering
365 SVOMOP=zeros(size(dVdZ));
366 SVOMOP(dVdZ<-5e-2)=-1;
367 SVOMOP(dVdZ>5e-2)=1;
368 figure
369 %imagesc(TempGrid,dVec,SVOMOP');
370 imagesc(TempGrid,dVec,dVdZ');
371 hold on
372 plot(Tsim,Dsim,'-m','linewidth',2);
373 plot(Tsim(1),Dsim(1),'or');
374 text(Tsim(1),Dsim(1),'Start','Color', 'w');
375 xlabel('Tunens temperatur')
376 ylabel('Dybde')
377 legend('Tunens adfærd', 'Location', 'SouthWest')
378 caxis([-0.5 0.5]);
379 colorbar
380 %savefig('../Skabelon/figurer/2701MovePlot',gcf,'pdf')
381 %savefig('../Skabelon/figurer/2301MovePlot',gcf,'pdf')
382 %gemFigur('2201MovePlot')
383
384 %% Plot af flad V kurve, når den konstante dybde er optimal
385
386
387 %Energi
388 %U(Ta(z))*f(z)
389 TaPP=ambientTemp(dVec, konstanter);
390 Upp=svomHastighed(TaPP,konstanter);
391 fpp=fodetaet(dVec,konstanter);
392 Epp=Upp.*fpp;
393
394 %Konstant Dybde
395 [a44,b44,c44,d44,e44,f44,g44] = konstantDybde(konstanter);
396
397 %Vinf(z,Ta(z')) hvor z' er dybden tunen slutter på
398
399 %%
400 figure
401 contour(TempGrid,dVec,V');
402 hold on
403 scale = 1/max(Epp)*10+min(TempGrid);
404 %plot(Epp*scale,dVec, 'linewidth',2.0);
405 plot(ambientTemp(g44, konstanter),g44,'rx', 'linewidth',2.0)
406 %plot(interp2(DVEC,TEMPGRID,V,dVec(:), ambientTemp(Dsim(end)
```

```
    ,konstanter)),dVec,'k')
plot(Tsim(end),Dsim(end),'mx','linewidth',2.0)
%plot(interp2(DVEC,TEMPGRID,V/1300,dVec(:), Tsim(end)),dVec
    ,'k', 'linewidth',1.3)
%contour(dVec,TempGrid, V)
axis ij
xlabel('Temperatur')
ylabel('Dybde')
%axis([0 1.4 0 600])
colorbar
legend('V_{\infty}','Optimal dybde ved konstant dybde','
    Slutpunkt for simuleringen', 'Location', 'SouthWest')
%savefig('../Skabelon/figurer/Vkurvekk',gcf,'pdf')
%%
figure;
imagesc(TempGrid,dVec,(spis+(1-spis).*(SvomOp+2))'); hold on
    ;
plot(Tsim,Dsim,'-m');
plot(Tsim(1),Dsim(1),'or');
text(Tsim(1),Dsim(1),'Start');
xlabel('Tunens temperatur')
ylabel('Dybde')
legend('Tunens adfærd')
%gemFigur('2211MovePlot')


%% Plot af den nye mu/varphi funktion

%[minSim2, maxSim2, gennemV2, Dsim2, Tsim2,Vmad2, MaxDk2] =
    bifurkationMad(konstanter.zSS, konstanter);


dVdZ2 = linspace(-15,15,2000); %Flere x-værdier ca. 1000
mu2 = -1+(2./(1+exp((-dVdZ2).*10)));
figure
plot(dVdZ2,mu2)
axis([-15 15 -1.2 1.2])
xlabel('$$\triangle V_{\infty}$$','Interpreter','latex')
ylabel('$$\varphi$$','Interpreter','latex')
%gemFigur('varPhiPlot')
%savefig('../Skabelon/figurer/varPhiPlot',gcf,'pdf')

%% Plot af bifurkationerne udført samtidigt fra kørel på
    Think Link
load bifurkation2D_result;
[SS,DD] = meshgrid(sValues,zDepths);
figure, surf(DD,SS,Dmax-Dmin,'EdgeColor','none','LineStyle
```

```
    ','none','FaceLighting','phong');
xlabel('z_{ss}')
ylabel('L_b')
zlabel('Dybde differens')
%savefig('../Skabelon/figurer/bifurkation2D',gcf,'pdf')
```

```
%%-----------------------------------------------------------------------

%%Script til at køre de to bifurkations funktioner.
%Plot af hvordan max og min dybde ændrer sig når z_ss og
    tunens længde
%ændrer sig
%%-----------------------------------------------------------------------


%%Konstanter

%Temperatur udviklingen i dybden
konstanter.Tmax = 27.5;  %overflade temperatur i grader
konstanter.p = -1/100; % temperaturfaldet efter thermoklinen
konstanter.A = 17; % Temperaturfaldet over thermoklinen
konstanter.Lambda = 2; % overgangen fra thermoklinen til
    temperatur faldet i dybdem
konstanter.zTcli = 100; % slut dybden på thermoklinen

%Svømmehastighed ud fra temperatur i konstantDybde
konstanter.Vmin=(2.0*70)/100;
konstanter.Vmax = (2.3*70)/100;

%Værdier for fødetætheden
konstanter.f0=0; %fødetætheden ved overfladen
konstanter.fSS=1; %
konstanter.zSS=500; %dybden hvor der er mest føde
konstanter.q=100; %længden af overgangszonen org. 100

%Værdier til kTemp i varDyb
konstanter.klow=0.03/60; %pr minut hos Malte så del med 60
konstanter.khigh = 0.3/60; %pr minut hos Malte så del med 60
konstanter.Tbb = 0.2; % org. 0.7. forskydning så tunen
    bliver afkølet lidt, inden der skiftes til khigh


%Periode
konstanter.Tp = (20*60);  %Én periode er 20 minutter

%Omega og epsilon
konstanter.omega = (2*pi)/konstanter.Tp; %Vinkel frekvens
```

```
37
38 %Start betingelser
39 konstanter.D0 = 300; %ud fra Olivier s. 125.
40 konstanter.T0 = 15; %ud fra Olivier s. 125.
41 konstanter.E0 = 0; % energien den har til at starte med er
       nul.
42
43 konstanter.TidI = konstanter.Tp*10; % skal køres i et helt
       antal perioder
44
45 %Konstanter til Phi=svingPhi i varDybPhi
46 %Den rette linje ud fra de to punkter
47 konstanter.Tlow = 17.5;
48 konstanter.Thigh = 22.5;
49 konstanter.Dlinje = 500;
50
51 konstanter.U0=1.5;
52
53 %Tunens størrelse
54
55 %%Tunens udgifter
56 %Svømme effekt
57 konstanter.beta=2;
58 konstanter.alpha=0;
59 %Varmetab
60 konstanter.c=0; %varmekapacitet
61
62 %% Bifurkationer ved at ændre på fødetætheden
63 %
64 vD=15; %Antal størrelser
65 zSizes = linspace(-50,350,vD); %Interval som zSS skal være
       imellem
66 %zSizes = linspace(0,150,vD); %Interval som zSS skal være
       imellem for zoom
67
68 Dmin = zeros(vD,1);
69 Dmax = zeros(vD,1);
70 KonD = zeros(vD,1);
71 EngZ = zeros(vD,1);
72
73 Vbox= zeros(vD,2);
74
75 tic;
76 for i = 1:vD; % matlabpool skrives i command winduet for
       parfor
77    zSize = zSizes(i);
78
```

```
79      %[minSim, maxSim, gennemV] = bifurkationMad((1+loopVar)
            *50);
80       %[minSim, maxSim, gennemV] = bifurkationMad(loopVar);
             Prøv denne på
81       %DTU
82
83      [minSim, maxSim, gennemV, Dsim, Tsim,Vmad, MaxDk] =
            bifurkationMad(zSize, konstanter);
84      Dmin(i) = minSim;
85      Dmax(i) = maxSim;
86
87      KonD(i) = MaxDk;
88      EngZ(i) = gennemV;
89      %Vbox(i,:) = [middel afvig];
90  end
91  tid=toc
92
93  %Plot af bifurkationerne
94  %figure
95  %plot(zSizes,Dmin, 'bx-')
96  %hold on
97  %plot(zSizes,Dmax, 'rx-')
98  %ylabel('Dybde')
99  %xlabel('Tunens længde, meter')
100 %legend('Minimumsdybde','Maksimumsdybde')
101 %axis([(min(zSizes)-0.1) (max(zSizes)+0.1) -50 (max(Dmax)
        +50)])
102 %title(['Plot når spredningen på fødefeltet= ' num2str(
        konstanter.q)])
103 %gemFigur('BifurLength2')
104
105 %%Konstant dybde
106 kk=500;
107 kp=100;
108 kZss=linspace(-50,350,kk);
109 DkonsP = zeros(kk,1);
110
111 for i = 1:kk;
112
113     konstanter.zSS = kZss(i);
114     H=konstanter.zSS+5*konstanter.q;
115     dybV8=linspace(0,H,kk);
116
117     [ zK8,TempK8, UK8, fK8 ,EK8, MaxEk8, MaxDk8 ] =
            konstantDybdeB(konstanter, dybV8);
118     DkonsP(i) = MaxDk8;
119 end
```

```
120
121
122
123
124 %%Bifurkations plot med konstant dybde
125 figure
126 plot(zSizes,Dmin, 'bx-')
127 hold on
128 plot(zSizes,Dmax, 'rx-')
129 hold on
130 plot(kZss, DkonsP,'k-')
131 %plot(zSizes,KonD, 'kx-')
132 ylabel('Dybde')
133 xlabel('z_{SS}')
134 legend('Minimumsdybde','Maksimumsdybde', 'Konstant dybde')
135 axis([(min(zSizes)-0.1) (max(zSizes)+0.1) -50 (max(Dmax)+50)
        ])
136 %gemFigur('BifurKonstantD8')
137 %savefig('../Skabelon/figurer/BifurZoom90',gcf,'pdf')
138
139 %savefig('../Skabelon/figurer/BifurKonstantD89',gcf,'pdf')
140 %saveFile= 'BiSize82'; %Indsæt filnavn
141 %save(saveFile, 'konstanter', 'Dmin', 'Dmax','Tsim', '
        gennemV', 'Vsize')
142
143 %% Plot af gennemsnits energihøsten som funktion af zSS.
144 figure
145 plot(zSizes, EngZ)
146 xlabel('z_{ss}')
147 ylabel('\gamma')
148 %gemFigur('EnergiHzSS8Zoom')
149 %savefig('../Skabelon/figurer/BifurKonstantE89',gcf,'pdf')
150
151 %Forskel i energihøst når z_ss varieres
152 %Energihøst for tunen når z_ss er 80:
153 zL=EngZ(end-13);
154
155 %Energihøst for tunen når z_ss er 100
156 zM=EngZ(end);
157
158 %% Bifurkationer ved at ændre på tunens længde/størrelse
159
160 nSize=10; %Antal størrelser, før var det 80.
161 lSizes = linspace(0.1,2.0,nSize); %Oprindeligt 0.1-1.5
162
163 Dmin = zeros(nSize,1);
164 Dmax = zeros(nSize,1);
```

```
165 EngLe = zeros(nSize,1);
166
167
168 tic
169 for i=1:nSize
170     lSize=lSizes(i);
171
172     [minSim, maxSim, gennemV3, Dsim, Tsim, Vsize] =
            bifurkationSize(lSize, konstanter);
173     Dmin(i) = minSim;
174     Dmax(i) = maxSim;
175     EngLe(i) = gennemV3;
176 end
177 toc
178
179 %% Plot af gennemsnits energihøsten som funktion af længden.
180 figure
181 plot(lSizes, EngLe)
182 xlabel('L_b')
183 ylabel('\gamma')
184 %gemFigur('EnergiHLength8')
185 %savefig('../Skabelon/figurer/BifurLengthE89',gcf,'pdf')
186
187
188 %% Bifurkations plot
189 figure
190 plot(lSizes,Dmin, 'bx-')
191 hold on
192 plot(lSizes,Dmax, 'rx-')
193 ylabel('Dybde')
194 xlabel('Tunens længde, meter')
195 legend('Minimumsdybde','Maksimumsdybde')
196 axis([(min(lSizes)-0.1) (max(lSizes)+0.1) -50 (max(Dmax)+50)
        ])
197 %gemFigur('BifurLength81')
198 %savefig('../Skabelon/figurer/BifurLength89',gcf,'pdf')
199
200 %saveFile= 'BiLengh82'; %Indsæt filnavn
201 %save(saveFile, 'konstanter', 'Dmin', ' dmax','Tsim', '
        gennemV3', 'Vsize')
```

```
1
2 function [minSim, maxSim, gennemV, Dsim, Tsim,V]=
      bifurkationSize(Lb, konstanter)
3 %-------------------------------------------------------------------------

4 %%Beregner min og max dybde som tunen har når den udfører
```

```
    optimal adfærd
%Lb er tunens længde i meter (body length)
%-------------------------------------------------------------------------


%konstanter.Lb=Lb;

%% Model funktioner, grids mm:

%Grids
nDyb=50;
nTemp=45;


%Skift: (indeksering)
Op=[1,1:(nDyb-1)]; %Rækkevektor med dobbelt start værdi
Ned=[2:nDyb,nDyb]; % Rækkevektor med dobbelt slut værdi

varme = [2:nTemp,nTemp]; %Rækkevektor med dobbelt slut
    nummer
kulde = [1,1:(nTemp-1)]; %Rækkevektor med dobbelt start
    nummer

%Tidsskridt
tidS=1;

H=konstanter.zSS+5*konstanter.q;
dVec=linspace(0,H,nDyb); %50 punkter fordelt lige over de
    500 meters dybde
MadVec=fodetaet(dVec, konstanter); %Fødetætheden i hvert
    dybdepunkt
TempVec=ambientTemp(dVec, konstanter); % vandtemp. i hvert
    dybdepunkt

TempGrid=linspace(TempVec(end),TempVec(1),nTemp);
%Temperaturen stiger igennem vektoren

DeltaDyb = dVec(2)-dVec(1); %Forskellen mellem 2 dybde
    punkter.
DeltaTemp= TempGrid(2)-TempGrid(1); %Forskellen mellem 2
    temp punkter

%% State space
[DD,TT]=meshgrid(dVec,TempGrid);
%DD: Søjle 1 indeholder 45 rækker med værdien dVec(1),
%TT: Række 1 indeholder 50 kolonner med værdien TempGrid(1).

```

```
43 %% Temperatur-, energi- og fødefunktioner i state space
       punkterne
44 Tamb=meshgrid(TempVec,TempGrid);
45 %Hver kolonne k indeholder 45 kopier af TempVec(k).
46 FodeS=meshgrid(MadVec,TempGrid);
47 %Hver kolonne k indeholder 45 kopier af MadVec(k).
48
49 Usvom=svomHastighedSize(TT, konstanter,Lb);
50 %Svømmehastigheden til den ambiente temp. med den givne
       længde Lb.
51 EnergiH=FodeS.*Usvom;
52 %Energihøst i statespace punkterne
53
54
55 %Varmekoefficienten evalueret i statespace og der ses på
       temp forskellen
56 k1 = kTemp(TT,DD, konstanter);
57 k= k1.*(Tamb-TT);
58
59 %% Optimal adfærd ud fra statespace ved baglæns iterationer
60
61 %Slut tilstand. På dette tidspunkt er det ikke længere
       muligt at have en
62 %fremtidig høste, det er ved solnedgang.
63 V = zeros(nTemp, nDyb);
64
65 maxIt=1e6;
66 Q=ones(1,maxIt)*Inf;
67 i=1;
68
69 while Q(i) >1e-8 && i<maxIt
70
71     Vgam=V;
72
73     %Værdien, V, opdateres ud fra kulde eller varme.
74     %Upwind discretization anvendes - forklar.
75     V = V + max(0,k)*tidS.*(V(varme,:)-V)/DeltaTemp - min(0,
           k)*tidS.*(V(kulde,:)-V)/DeltaTemp;
76     %k*(Ta-T) * forskellen i fitness fra et step til det
           næste. Første led
77     %i formlens max udtryk.
78
79
80     %Vælg nu den optimale adfærd. Dyk, svøm op og jag
81     %For hvert time step bevæges en grid cell når der
           svømmes eller dykkes
82     V=V+EnergiH.*tidS + tidS*max(max(0,(V(:,Op)-V)/DeltaDyb
```

```
        .*Usvom)...
83          ,(V(:,Ned)-V)/DeltaDyb.*Usvom);
84
85      % Den optimale adfærd findes ved at se hvilket udtryk
            der er størst.
86      % I det nye udtryk kan den spise samtidigt med at den
            svømmer
87
88      i=i+1;
89      %Q(i)=norm(V-Vgam)/norm(V);
90      deltaV=V-Vgam;
91      Q(i)=max(deltaV(:)) - min(deltaV(:));
92
93  end
94  gennemV=mean(deltaV(:))/tidS;
95
96  spis=max(V(:,Op),V(:,Ned)) < V+EnergiH./DeltaDyb*tidS;
97          %spis=max(V(:,Op),V(:,Ned)) < V+EnergiH*tidS;
                Oprindelig
98          %Giver 1 eller 0, alt afhængigt af om det er sandt
                eller falskt.
99          %Hvis det ikke giver bedre "fitness" at ændre dybde
                så spis.
100
101 SvomOp= V(:,Op) > V(:,Ned);
102         %Forskellen mellem nu og næste step er negativ. Den
                taber
103         %"fitness", svøm derfor op. Hvis sandt, er der 1
                taller ellers
104         % nul og dermed svøm ned.
105
106 %% Simulering af tunens bevægelse
107
108 D0 = konstanter.D0;
109 T0 = konstanter.T0;
110
111
112 AntalSim = 20000; %Antal tidsskridt, dvs. hvis den kun skal
        være for en halv dag så: 60*60*12.
113 %AntalSim = 51000;
114
115 Dsim = zeros(AntalSim,1); %Vektorer
116 Tsim = zeros(AntalSim,1);
117
118 Dsim(1) = 0; %Først punkt sættes til startværdier. Ændret
        for at bifur plot passer
119 Tsim(1) = 27.5; % Originalt var det D0 og T0.
```

```
120
121 [DVEC,TEMPGRID] = meshgrid(dVec,TempGrid);
122
123 %Overall
124 dVdZ=(V(:,Ned)-V(:,Op))./(DeltaDyb*2);
125
126 %Overgangen mellem at spise og svømme udglattes
127 mu = -1+(2./(1+exp(-dVdZ.*10)));
128
129 for i=1:(AntalSim-1)
130
131   %dTdtsim = interp2(DVEC,TEMPGRID,k,Dsim(i)*(1-1e-6),Tsim(i
        ),'linear');
132   k1 = kTemp(Tsim(i),Dsim(i), konstanter);
133   dTdtsim= k1.*(ambientTemp(Dsim(i),konstanter)-Tsim(i)); %
        dTdt i formlen
134   Tsim(i+1) = Tsim(i) + dTdtsim * tidS;
135
136   hastighedSim = svomHastighedSize(Tsim(i), konstanter,Lb);
137
138   dVdZsim=interp2(DVEC, TEMPGRID, dVdZ, Dsim(i), Tsim(i), '
        cubic',0);
139
140   Dsim(i+1) = Dsim(i)+tidS*hastighedSim*(-1+(2./(1+exp(-
        dVdZsim.*10))));
141   Dsim(i+1) = max(0,Dsim(i+1)); %sikre at tunen bliver inden
         for området.
142   Dsim(i+1) = min(H,Dsim(i+1));
143 end
144
145
146
147 %Fjern den første del så det bliver det rigtige minimum
148       ddF=Dsim(AntalSim/2:end);
149       minSim = min(ddF);
150       maxSim = max(ddF);
151
152 end
```

```
1
2 function [minSim, maxSim, gennemV, Dsim, Tsim, V, MaxDk]=
      bifurkationMad(fodDyb, konstanter)
3 %function [minSim, maxSim, gennemV, Dsim, Tsim, V, middel,
      afvig, MaxDk]= bifurkationMad(fodDyb, konstanter)
4
5 %------------------------------------------------------------------------
```

```
6  %%Beregner min og max dybde som tunen har når den udfører optimal adfærd
7  %fodDyb er den dybde hvor der er mest føde, hvor gauss klokken har sit max
8  %-------------------------------------------------------------------------

9
10 konstanter.zSS=fodDyb; %sæt konstanten lig den nye dybde
11
12 %%Model funktioner, grids mm:
13
14 %Grids
15 nDyb=50;
16 nTemp=45;
17
18
19 %Skift: (indeksering)
20 Op=[1,1:(nDyb-1)];                  %Rækkevektor med dobbelt start værdi
21 Ned=[2:nDyb,nDyb];                  %Rækkevektor med dobbelt slut værdi
22
23 varme = [2:nTemp,nTemp];            %Rækkevektor med dobbelt slut nummer
24 kulde = [1,1:(nTemp-1)];            %Rækkevektor med dobbelt start nummer
25
26 %Tidsskridt
27 tidS=1;
28
29 %H=500;
30 H=konstanter.zSS+5*konstanter.q;
31 dVec=linspace(0,H,nDyb);             %50 punkter fordelt lige over de 500 meters dybde
32 MadVec=fodetaet(dVec, konstanter);  %Fødetætheden i hvert dybdepunkt
33 TempVec=ambientTemp(dVec, konstanter); % vandtemp. i hvert dybdepunkt
34
35 TempGrid=linspace(TempVec(end),TempVec(1),nTemp);
36                                     %Temperaturen stiger igennem vektoren
37
38 DeltaDyb = dVec(2)-dVec(1);         %Forskellen mellem 2 dybde punkter.
39 DeltaTemp= TempGrid(2)-TempGrid(1); %Forskellen mellem 2 temp punkter
```

```
40
41 %% Den konstante dybde til en given zSS.
42 [ zK,TempK, UK, fK ,EK, MaxEk, MaxDk ] = konstantDybdeB(
       konstanter,dVec);
43
44 %inkluder 500 punkter istedet for at gøre det ud fra dVec
45
46 %% State space
47 [DD,TT]=meshgrid(dVec,TempGrid);
48 %DD: Søjle 1 indeholder 45 rækker med værdien dVec(1),
49 %TT: Række 1 indeholder 50 kolonner med værdien TempGrid(1).
50
51 %% Temperatur-, energi- og fødefunktioner i state space
       punkterne
52
53 Tamb=meshgrid(TempVec,TempGrid);    %Hver kolonne k
       indeholder 45 kopier af TempVec(k).
54 FodeS=meshgrid(MadVec,TempGrid);    %Hver kolonne k
       indeholder 45 kopier af MadVec(k).
55
56
57 Usvom=svomHastighed(TT, konstanter);    %Svømmehastigheden
       til den ambiente temp.
58 EnergiH=FodeS.*Usvom;                   %Energihøst i
       statespace punkterne
59
60
61 %Varmekoefficienten evalueret i statespace og der ses på
       temp forskellen
62 k1 = kTemp(TT,DD, konstanter);
63 k= k1.*(Tamb-TT); %dTdt i formlen
64
65 %% Optimal adfærd ud fra statespace ved baglæns iterationer
66
67 %Slut tilstand. På dette tidspunkt er det ikke længere
       muligt at have en
68 %fremtidig høste, det er ved solnedgang.
69 V = zeros(nTemp, nDyb);
70
71 maxIt=1e6;
72 Q=ones(1,maxIt)*Inf;
73 i=1;
74
75 while Q(i) >1e-14 && i<maxIt %normalt er -8
76
77     Vgam=V;
78
```

```
79      %Værdien, V, opdateres ud fra kulde eller varme.
80      %Upwind discretization anvendes
81      V = V + max(0,k)*tidS.*(V(varme,:)-V)/DeltaTemp - min(0,
            k)*tidS.*(V(kulde,:)-V)/DeltaTemp;
82      %k*(Ta-T) * forskellen i fitness fra et step til det
            næste. Første led
83      %i formlens max udtryk.
84
85
86      %Vælg nu den optimale adfærd. Dyk, svøm op og jag
87      %For hvert time step bevæges en grid cell når der
            svømmes eller dykkes
88      V=V+EnergiH.*tidS + tidS.*max(max(0,(V(:,Op)-V)/DeltaDyb
            .*Usvom)...
89           ,(V(:,Ned)-V)/DeltaDyb.*Usvom);
90
91      % Den optimale adfærd findes ved at se hvilket udtryk
            der er størst.
92      % I det nye udtryk kan den spise samtidigt med at den
            svømmer
93
94      i=i+1;
95      %Q(i)=norm(V-Vgam)/norm(V);
96      deltaV=V-Vgam;
97      Q(i)=max(deltaV(:)) - min(deltaV(:));
98
99  end
100 gennemV=mean(deltaV(:))/tidS;
101
102 %spis=max(V(:,Op),V(:,Ned)) < V+EnergiH./DeltaDyb*tidS; %opr
        .
103          spis=max(V(:,Op),V(:,Ned)) < V+EnergiH./DeltaDyb*
                 tidS;
104          %Giver 1 eller 0, alt afhængigt af om det er sandt
                 eller falskt.
105          %Hvis det ikke giver bedre "fitness" at ændre dybde
                 så spis.
106
107 SvomOp= V(:,Op) > V(:,Ned);
108          %Forskellen mellem nu og næste step er negativ. Den
                 taber
109          %"fitness", svøm derfor op. Hvis sandt, er der 1
                 taller ellers
110          % nul og dermed svøm ned.
111
112 %% Simulering af tunens bevægelse
113
```

```
D0 = konstanter.D0; %startværdier
T0 = konstanter.T0;

AntalSim = 2e4; %Antal tidsskridt, før d. 9/1 var det 20000

Dsim = zeros(AntalSim,1); %Vektorer
Tsim = zeros(AntalSim,1);

%Dsim(1) = D0; %Først punkt sættes til startværdier
%Tsim(1) = T0;

Dsim(1) = 0; %Først punkt sættes til startværdier
Tsim(1) = 27.5;

[DVEC,TEMPGRID] = meshgrid(dVec,TempGrid);

%Overall
dVdZ=(V(:,Ned)-V(:,Op))./(DeltaDyb*2);

%Overgangen mellem at spise og svømme udglattes
mu = -1+(2./(1+exp(-dVdZ.*10)));

%figure
%plot(dVdZ,mu)

for i=1:(AntalSim-1)

k1 = kTemp(Tsim(i),Dsim(i), konstanter);
dTdtsim= k1.*(ambientTemp(Dsim(i),konstanter)-Tsim(i));
Tsim(i+1) = Tsim(i) + dTdtsim * tidS;

hastighedSim = svomHastighed(Tsim(i), konstanter);

dVdZsim=interp2(DVEC, TEMPGRID, dVdZ, Dsim(i), Tsim(i), '
    cubic',0);

Dsim(i+1) = Dsim(i)+tidS*hastighedSim*(-1+(2./(1+exp(-
    dVdZsim.*10))));
Dsim(i+1) = max(0,Dsim(i+1)); %sikre at tunen bliver inden
    for området.
Dsim(i+1) = min(H,Dsim(i+1));
end

 %Finder minimum efter det første er taget væk
 ddF=Dsim(AntalSim/2:end);
 minSim = min(ddF);
       maxSim = max(ddF);
```

```
158
159 end
```

```
1  %%-------------------------------------------------------------------------
2  %%Script fra Uffe Høgsbro Thygesen.
3  %%% Vertuna - optimal vertical movements of a vertual tuna
4  %%-------------------------------------------------------------------------
5  %clear all
6  %close all
7
8  %%% Model parameters
9  KappaHeat = 0.02;
10 KappaCool = 0.005;
11
12
13 %%% Grids
14 nZ = 50;
15 nT = 45;
16
17 %%% Indeces for shift
18 UpI = [1,1:(nZ-1)];
19 DoI = [2:nZ,nZ];
20
21 HeatI = [2:nT,nT];
22 CoolI = [1,1:(nT-1)];
23
24 %%% Time step
25 timestep = 1;
26
27 %%% Water column
28 H = 350;
29 Zvec = linspace(0,H,nZ);
30 Rhovec = 1./(1+exp(-(Zvec/H-0.85)*10));
31 Tvec = 28 - 18*1./(1+exp(-(Zvec/H-1/2)*10));
32
33 figure(1)
34
35 subplot(2,2,1)
36 plot(Rhovec,-Zvec)
37 hold on
38 plot(Tvec,-Zvec)
39 hold off
40
41 Zgrid = Zvec;
42 Tgrid = linspace(Tvec(end),Tvec(1),nT);
```

```
43
44 deltaZ = Zgrid(2)-Zgrid(1);
45 deltaT = Tgrid(2)-Tgrid(1);
46
47 %%% Make state space
48 [ZZ,TT] = meshgrid(Zgrid,Tgrid);
49
50 %% Construct matrices of ambient temperature and food as
       functions on state space
51 Tamb = meshgrid(Tvec,Tgrid);
52 RhoAmb = meshgrid(Rhovec,Tgrid);
53
54 %% Feeding rate as function on state space
55 FeedingRate = RhoAmb .* (TT-5);
56
57 %% Rate of change of body temperature as function on state
       space
58 dTdt = KappaHeat*max(0,Tamb-TT) + KappaCool*min(0,Tamb-TT);
59
60 disp('Diagnostocs: Time step control')
61 disp(max(abs(dTdt(:)))*timestep/deltaT)
62
63 subplot(2,2,2)
64 imagesc(Zgrid,Tgrid,dTdt)
65 colorbar
66
67 subplot(2,2,3)
68 imagesc(Zgrid,Tgrid,FeedingRate)
69 colorbar
70
71 figure(2)
72
73 %%% Optimal gain as function on state space
74 %%% Terminal condition "sunset"; no future harvest
75 V = zeros(nT,nZ);
76
77 %% Backward iteration: Propagate value function backwards in
        time
78 for i=1:1000,
79
80   % Update value function due to heating/cooling
81   % Use first order upwind discretization
82   V = V + max(0,dTdt)*timestep.*(V(HeatI,:)-V)/deltaT  - min
       (0,dTdt)*timestep.*(V(CoolI,:)-V)/deltaT;
83
84    %Ændring af V(:,op)
85     %Vop=V+((TT-5).*(V(:,UpI)-V).*timestep)./deltaZ;
```

```
86      %Vned=V+((TT-5).*(V(:,DoI)-V).*timestep)./deltaZ;
87
88    % Choose optimal behaviour between foragin, surfacing,
          diving.
89    % When surfacing or diving, move one grid cell per time
          step (!)
90    V = max(max(V+FeedingRate*timestep,V(:,UpI)),V(:,DoI));
91      %V = max(max(V+FeedingRate*timestep,Vop),Vned);
92
93
94    if(mod(i,10)==0),
95      subplot(1,2,1)
96      imagesc(Tgrid,Zgrid,V');
97      title('Energy gain');
98      xlabel('Body Temperature')
99      ylabel('Depth')
100     colorbar
101 %
102      Feed = max(V(:,UpI),V(:,DoI)) < V+FeedingRate*timestep;
103      MoveUp = V(:,UpI) > V(:,DoI);
104
105     subplot(1,2,2)
106     imagesc(Tgrid,Zgrid,(Feed + (1-Feed).*(MoveUp+2))');
107     title('Optimal behavior');
108     xlabel('Body Temperature')
109     ch = colorbar;
110     set(ch,'visible','off')
111     pause(1)
112   end
113
114 end
115 %%
116 %%% Simulate
117
118 Z0 = 150;
119 T0 = 15;
120
121 Nsim = 2000;
122
123 Zsim = zeros(Nsim,1);
124 Tsim = zeros(Nsim,1);
125
126 Zsim(1) = Z0;
127 Tsim(1) = T0;
128
129 for i=1:(Nsim-1)
130   dTdtsim = interp2(Zgrid,Tgrid,dTdt,Zsim(i)*(1-1e-6),Tsim(i
```

```
      ));
131
132   Tsim(i+1) = Tsim(i) + dTdtsim * timestep;
133
134   FeedSim = interp2(Zgrid,Tgrid,Feed,Zsim(i),Tsim(i)*(1-1e
          -6),'nearest');
135   MoveUpSim = interp2(Zgrid,Tgrid,MoveUp,Zsim(i),Tsim(i)
          *(1-1e-6),'nearest');
136
137   if ~FeedSim,
138     if MoveUpSim,
139       Zsim(i+1) = Zsim(i) - deltaZ;
140     else
141       Zsim(i+1) = Zsim(i) + deltaZ;
142     end
143   else
144     Zsim(i+1) = Zsim(i);
145   end
146
147 end
148
149 figure
150
151 subplot(2,1,1)
152 plot(-Zsim)
153 axis([0 Nsim -500 50])
154
155 subplot(2,1,2)
156 plot(Tsim)
157 axis([0 Nsim 0 30])
158 hold on
159 plot(interp1(Zvec,Tvec,Zsim*0.999))
160 hold off
```

```
1
2  %%-------------------------------------------------------------------------
3  %%Data importering af test data fra Toby
4  %%-------------------------------------------------------------------------
5
6  %SKAL DET FØLGENDE SLETTES?
7  %Omformet til data fil
8
9   %D= importdata('data.txt',' ',1);
10  %data=D.data;
11  %fisk_nr = str2num(cell2mat(D.textdata(2:end,1)));
```

```
12  %dato = D.textdata(2:end,2);
13  %tid = D.textdata(2:end,3);
14  %datocell = cellstr([cell2mat(dato) repmat(' ',length(tid)
        ,1) cell2mat(tid)]);
15  %datovec = datenum(datocell);
16  %X = [fisk_nr, datovec, data];
17  %save('data.mat','X');
18
19 %load data;
20 %fisk_nr = X(:,1);
21 %tid = X(:,2);
22 %data = X(:,3:5);
23 %dataMinus=-1*data;
24
25 %Data plot
26 %figure,
27 %plot(tid,dataMinus(:,3));
28 %datetick('x','mmmyy','keepticks');
29 %legend(['Tun nr: ' num2str(fisk_nr(1))]);
30 %gemFigur('AlTestData');
31
32 %Fra midt dag og 24 timer frem
33 %t=(24*60)/4; %måling hver 4 minut
34 %figure
35 %plot(tid(1:t,1),dataMinus(1:t,3));
36 %datetick('x','mmmyy','keepticks');
37
38 %Kun 1 dag
39 %figure
40 %tidM=datevec(tid);
41 %EnEnkeltDag=find(tidM(:,3)==13 & tidM(:,2)==10);
42 %plot(tid(EnEnkeltDag,1),dataMinus(EnEnkeltDag,3));
43 %datetick('x','HH:MM','keepticks');
44 %legend('D. 13/10/2001')
45 %gemFigur('13oktoberPlot2');
46
47 %% Data importering af Toby data
48
49 %99190.txt
50 %99-224.txt
51 %99243.txt
52
53 filnavn = '99190.txt';
54 D = importdata(filnavn,',',2); %data starter først efter to
       rækker
55
56 %Omformet til data fil
```

```
57 data = D.data; % D.data indeholder en matrix med de sidste 5
       kolonner
58
59 %Dato omformning
60 datestring = cat(1,D.textdata(3:end,1)); % Saml alle datoer
       i kolonne af cell array. Spring de to første rækker over
61 tid = datenum(datestring,'dd/mm/yyyy HH:MM:ss');
62 % Konverter til datenum (sekunder siden 1970)
63 % Jeg sætter her datoformatet eksplicit for at være sikker
       på
64 % at den gør det rigtigt
65
66 %dato2 = datenum(D.textdata(3:end,1)); %D.textdata er et
       cell array hvor første kolonne indeholder dato
67
68 %gem data
69 Y= [tid, data];
70 save('data2.mat','Y');
71
72 %%Load data
73 load data2
74 tid2 = Y(:,1)+10/24;
75 data = Y(:,2:6);
76 dataMinus = -1*data;
77
78 %%Plot data tun, al data
79 figure
80 plot(tid2,data(:,1));
81 axis ij
82 datetick('x','mmmyy','keepticks'); % x for på x-aksen
83 legend(['tun nr: ' num2str(filnavn)]);
84 %gemFigur('Tun99243AL')
85 %gemFigur('Tun99-224AL')
86 %gemFigur('Tun99190AL')
87
88 %Plot for tidlig morgen og 2 dage frem
89 t2=(24*60*5)/4;
90 figure
91 scatter(tid2(1:t2),data(1:t2,1));
92 axis ij
93 datetick('x','mmmyy','keepticks');
94 legend(['tun nr: ' num2str(filnavn)]); % der er noget galt
       med tun 99190
95 %gemFigur('Tun99243Endag')
96 %gemFigur('Tun99-224Endag')
97 %gemFigur('Tun99190Endag')
98
```

```
99  %Plot 200 dage inde og et døgn frem
100 ts=(200*28*60)/4;
101 t3=(24*60)/4;
102 figure
103 plot(tid2(ts:ts+t3),data(ts:ts+t3,1));
104 axis ij
105 datetick('x','HH:MM','keepticks');
106 legend(['tun nr: ' num2str(filnavn)]);
107 %gemFigur('Tun99243Tohundrede')
108 %gemFigur('Tun99-224Tohundrede')
109 %gemFigur('Tun99190Tohundrede')
```

```
1  %%-------------------------------------------------------------------------
2  %%Udregner gennemsnitlig energihøst og viser plots for alle
       tre modeller i
3  % den statiske del af projektet
4  %%-------------------------------------------------------------------------
5
6  %%Konstanter
7
8  clc, clear, close all
9
10 %Temperatur udviklingen i dybden
11 konstanter.Tmax = 27.5;  %overflade temperatur i grader
12 konstanter.p = -1/100; % temperaturfaldet efter thermoklinen
13 konstanter.A = 17; % Temperaturfaldet over thermoklinen
14 konstanter.Lambda = 2; % overgangen fra thermoklinen til
       temperatur faldet i dybdem
15 konstanter.zTcli = 100; % slut dybden på thermoklinen
16
17 %Svømmehastighed ud fra temperatur i konstantDybde
18 konstanter.Vmin=(2.0*70)/100;
19 konstanter.Vmax = (2.3*70)/100;
20
21 %Værdier for fødetætheden
22 konstanter.f0=0; %fødetætheden ved overfladen
23 konstanter.fSS=1; %
24 konstanter.zSS=500; %dybden hvor der er mest føde, er dybF
       -100.
25 konstanter.q=100; %længden af overgangszonen
26
27 %Værdier til kTemp i varDyb
28 konstanter.klow=0.03/60; %pr minut hos Malte så del med 60
29 konstanter.khigh = 0.3/60; %pr minut hos Malte så del med 60
30 konstanter.Tbb = 0.2; %forskydning så tunen bliver afkølet
```

```
    lidt, inden der skiftes til khigh
31
32
33  %Periode
34  konstanter.Tp = (20*60);  %Én periode er 20 minutter
35
36  %Omega og epsilon
37  konstanter.omega = (2*pi)/konstanter.Tp; %Vinkel frekvens
38
39  %Start betingelser
40  konstanter.D0 = 300; %ud fra Olivier s. 125.
41  konstanter.T0 = 15; %ud fra Olivier s. 125.
42  konstanter.E0 = 0; % energien den har til at starte med er
        nul.
43
44  konstanter.TidI = konstanter.Tp*10; % skal køres i et helt
        antal perioder
45
46  %Konstanter til Phi=svingPhi i varDybPhi
47  %Den rette linje ud fra de to punkter
48  konstanter.Tlow = 17.5;
49  %konstanter.Tlow = -54.2; %test
50  %konstanter.Tlow = 12.995; %test
51
52
53  konstanter.Thigh = 22.5;
54  %konstanter.Thigh = 14.7; % test
55  %konstanter.Thigh = 25.1;
56
57
58
59  konstanter.Dlinje = 500; % I rapport = Dspis
60  %y-værdi til linjen i varDybPhi
61  %konstanter.Dlinje = 498.6; %test
62
63
64  konstanter.U0=1.5;
65
66
67  %% Optimer på de to værdier for dybde temperatur
68
69  %x_start=[500;17.5;22.5];
70  x_start=[17.5;22.5];
71  options = optimset('fminsearch');
72  %options = optimset(options,'Display','iter','PlotFcns', {
        @optimplotfval});
73  options = optimset(options,'TolFun',1e-1,'TolX',1e-1,'
```

```
    Display','iter','PlotFcns', {@optimplotfval,
    @optimplotLinje});
e_stjerne = fminsearch(@(x) SvingLogiskWrapper(x,konstanter)
    , x_start, options);

%% Variabel dybde med logisk sving
konstanter.Thigh = e_stjerne(2);
konstanter.Tlow = e_stjerne(1);

[tl, Xl, gennemL] = SvingLogisk(konstanter);
%%
minTemp33=min(Xl(100:end,2))

%% Konstant dybde

[ dybdeVektor, TempFunktion, UsvommeHastighed, fodetaet2 ,
    Energihost, MaxE, MaxD ] = konstantDybde(konstanter);
MaxTk = ambientTemp(MaxD, konstanter);

%%Dybde og energihøst
figure
hold on
plot(dybdeVektor, Energihost);
hold on
plot(MaxD, MaxE, 'rx','linewidth',2);
ylabel('Energihøst')
xlabel('Dybde')
%gemFigur('KonstantDybdePlot22') Til rapport
%savefig('../Skabelon/figurer/KonstantDybdePlot23',gcf,'pdf
    ')
%% Temperatur- og fødekurve
dybX=1:1500;
TunTemp=1:25;

Ta1=ambientTemp(dybX, konstanter); %Vandets temperatur
fode1=fodetaet(dybX, konstanter); %Fødetætheden

figure
plot(dybX,Ta1)
axis ij



%Linjen der skiller de to zoner for logiske svingninger
Ttemp2 = [konstanter.Tlow, konstanter.Thigh];
Ddyb2 = [konstanter.Dlinje, 0];
p2 = polyfit(Ttemp2, Ddyb2,1);
```

```
116 Lzone2 = p2(2) + p2(1).*TunTemp;
117 %%
118 % %k-værdien
119 % dybX2=1:60:1500;
120 % k2 = kTemp(TunTemp, dybX2, konstanter);
121 % %Hvis denne skal plottes skal dybX og TunTemp være lige
        lange.
122 %
123 % % figure
124 % plot((TunTemp-Ta1(:,1:25)),k2)
125 % xlabel('T-T_a')
126 % ylabel('k')
127 % legend('k_{low}=0.0005, k_{high}=0.0050, T_{bb}=0.7')
128 % %gemfigur('OptimeringLssK')
129
130 %% Variabel dybde med sinus
131
132 L_n = 100;
133 %konstanter
134 [ts, Xs, gennemS]=SvingSinus(L_n, konstanter);
135 %konstanter
136 %
137 figure
138 plot(ts,Xs(:,1))
139 axis ij
140 xlabel('Tid')
141 ylabel('Dybde')
142
143 %Varier epsilon
144
145 %% Samlet plot til at få overblik
146
147 %Plot
148 figure
149 subplot(3,3,1)
150 plot(tl,Xl(:,1))
151 axis ij
152 title('Logiske svingninger')
153 xlabel('Tid')
154 ylabel('Dybde')
155 %
156 subplot(3,3,2)
157 plot(tl,Xl(:,2))
158 title('Logiske svingninger')
159 xlabel('Tid')
160 ylabel('Temperatur')
161 %
```

```
subplot(3,3,3)
plot (tl, Xl(:,3))
title('Logiske svingninger')
xlabel('Tid')
ylabel('Energihøst')
%
subplot(3,3,4)
hold on
plot(Ta1, dybX,'b')
hold on
plot(MaxTk,MaxD,'xk')
%plot(TunTemp, Lzone2, 'g')
axis ij
xlabel('Temperatur')
ylabel('Dybde')
legend('Ambient temperatur','Den konstante dybde for tunen
    ','Location', 'SouthEast')
%
subplot(3,3,5)
plot(fode1,dybX,'r')
xlabel('Fødetæthed')
ylabel('Dybde')
axis ij
%
subplot(3,3,6)
hold on
text(0,0.8,['Logisk E-værdi=' num2str(gennemL)])
hold on
text(0,0.6, ['Konstant dybde E-værdi=' num2str(MaxE)])
hold on
text(0,0.4, ['Sinus E-værdi=' num2str(gennemS)])
hold on
text(0,0.2,['Optimerede værdier: Tlow= ' num2str(konstanter.
    Tlow) ', Thigh= ' num2str(konstanter.Thigh)])
axis off

subplot(3,3,7)
plot(ts,Xs(:,1))
axis ij
title('Sinus svingninger')
xlabel('Tid')
ylabel('Dybde')

subplot(3,3,8)
plot(ts,Xs(:,2))
axis ij
title('Sinus svingninger')
```

```
207 xlabel('Tid')
208 ylabel('Temperatur')
209
210 subplot(3,3,9)
211 plot(ts,Xs(:,3))
212 title('Sinus svingninger')
213 xlabel('Tid')
214 ylabel('Energihøst')
215
216 %% Plot til rapport af bevægelsen mm hvor det er samlet
217
218 %dybde
219 figure
220 hold on
221 plot(ts,Xs(:,1),'r')
222 plot(tl,Xl(:,1),'b')
223 axis ij
224 xlabel('Tid')
225 ylabel('Dybde')
226 legend('Sinussvingninger','Logiske svingninger')
227 %gemFigur('dybdeSinLog1')
228 %savefig('../Skabelon/figurer/dybdeSinLog2',gcf,'pdf')
229
230
231 %Temperatur
232 figure
233 hold on
234 plot(ts,Xs(:,2),'r')
235 plot(tl,Xl(:,2),'b')
236 axis ij
237 xlabel('Tid')
238 ylabel('Temperatur')
239 legend('Sinussvingninger','Logiske svingninger')
240 %gemFigur('tempSinLog1')
241 %savefig('../Skabelon/figurer/tempSinLog2',gcf,'pdf')
242
243
244 %Energi
245 figure
246 hold on
247 plot(ts,Xs(:,3),'r')
248 plot (tl, Xl(:,3),'b')
249 xlabel('Tid')
250 ylabel('Energihøst')
251 legend('Sinussvingninger','Logiske svingninger')
252 %gemFigur('energiSinLog1')
253 %savefig('../Skabelon/figurer/energiSinLog2',gcf,'pdf')
```

```
254
255
256 %% Plot af fødetætheden til rapport
257
258
259 figure
260 hold on
261 text(konstanter.fSS+0.02, 1400,'f_{ss}','Color', 0.4.*[1 1
        1],'fontsize',15);
262 plot([konstanter.fSS konstanter.fSS], [konstanter.zSS
        1500],'--','Color',0.4.*[1 1 1],'linewidth',2.0)
263 %
264 text( 0, konstanter.zSS-50,'z_{ss}','Color', 0.4.*[1 1 1],'
        fontsize',15);
265 text( 0.02, konstanter.f0 + 40,'f_{0}','Color', 0.4.*[1 1
        1],'fontsize',15);
266 plot([0 konstanter.fSS], [konstanter.zSS konstanter.zSS
        ],'--','Color',0.4.*[1 1 1],'linewidth',2.0)
267 %
268 text( 0.42, 412.5,'q','Color', 0.4.*[1 1 1],'fontsize',15);
269 plot([0.4 0.4], [367 490],'--','Color',0.4.*[1 1 1],'
        linewidth',2.0)
270 %
271 fPP=plot(fode1,dybX,'r','linewidth',2.3);
272 xlabel('Fødetæthed')
273 ylabel('Dybde')
274 axis([0 1.1 0 1500])
275 axis ij
276 legend(fPP,'Fødetætheden','Location', 'SouthWest')
277 %gemfigur('DybdeTempOrg')
278 %savefig('../Skabelon/figurer/DybdeTempOrg2',gcf,'pdf')
279
280 %% Plot af ambient temperatur til rapport
281
282 figure
283 hold on
284 text( konstanter.Tmax-2, 1450,'T_{max}','Color', 0.4.*[1 1
        1],'fontsize',15);%r
285 plot([konstanter.Tmax konstanter.Tmax], [-30 1400],'--','
        Color',0.4.*[1 1 1],'linewidth',2.0)%r
286 %
287 %text( 19.5, 1450,'T_{cli}','Color', 0.4.*[1 1 1],'fontsize
        ',15);%m
288 %plot([20.5 20.5], [80 1400],'--','Color',0.4.*[1 1 1],'
        linewidth',2.0) %m
289 %
290 %plot([20.5 konstanter.Tmax], [1250 1250],'--','Color
```

```
    ',0.4.*[1 1 1],'linewidth',2.0)
text( 18, -40,'A','Color', 0.4.*[1 1 1],'fontsize',15);
plot([0.43 12], [1032 -10], '--', 'Color',0.6.*[1 1 1],'
    linewidth',2.0)
plot([12 27.5], [ 5 5 ], '--', 'Color',0.4.*[1 1 1],'
    linewidth',2.0)
%
%
text(0,120,'z_{T{cli}}','Color',0.4.*[1 1 1],'fontsize',15)
    ;%g
plot([2.5 18.4], [100 100],'--','Color',0.4.*[1 1 1],'
    linewidth',2.0) %g
%
plot([2.7 2.7], [700 800],'--','Color',0.4.*[1 1 1],'
    linewidth',2.0)
plot([2.7 4.0], [700 700],'--','Color',0.4.*[1 1 1],'
    linewidth',2.0)
text(1.9,720,'\rho','Color',0.4.*[1 1 1],'fontsize',15);
%
aaB=plot(Ta1,dybX,'b','linewidth',2.3);
axis([-0.3 28 -60 1500])
xlabel('Ambient temperatur')
ylabel('Dybde')
axis ij
legend(aaB,'Ambiente temperatur', 'Location', 'SouthWest')
%gemfigur('DybdeFodeOrg')
%savefig('../Skabelon/figurer/DybdeFodeOrg2',gcf,'pdf')

%% Konstant dybde plot
% %Dybde og temperatur
% figure
% plot(TempFunktion,dybdeVektor);
% xlabel('Temperatur')
% ylabel('Dybde')
% axis ij
%
% %Dybde og fødetæthed
% figure
% plot(fodetaet,dybdeVektor);
% xlabel('Fødetæthed')
% ylabel('Dybde')
% axis ij
%
% %Dybde og energihøst
% figure
% hold on
% plot(Energihost,dybdeVektor);
```

```
% hold on
% plot(MaxE, MaxD, 'rx');
% xlabel('Energihøst')
% ylabel('Dybde')
% axis ij

%% Variabel dybde med sinus

%L_n = 50;
%konstanter
%[ts, Xs, gennemS]=SvingSinus(L_n, konstanter);
%konstanter
%
%figure
%plot(ts,Xs(:,1))
%axis ij
%xlabel('Tid')
%ylabel('Dybde')
%gemFigur('dybdeVarDybEps1')
%gemFigur('dybdeVarDybEps') %her er k ikke ændret til en
    funktion


% figure
% plot(ts,Xs(:,2))

%HVIS dette plot skal laves skal der tages konstanter fra
    SvingSinus og
%sættes ind i den variable konstanter. Det er plot for at se
     at antal
%perioder passer.
% figure
% hold on
% scatter(ts,Xs(:,2))
% hold on
% plot(ts(row1),Xs(row1,2),'rX')
% plot(ts(row), Xs(row,2), 'rx')
% axis
% xlabel('Tid')
% ylabel('Temperatur')
%gemFigur('dybdeVarTempEps1')
%gemFigur('dybdeVarTempEps') %her er k ikke ændret til en
    funktion


% figure
% plot(ts,Xs(:,3))
```

```
374 % xlabel('Tid')
375 % ylabel('Energihøst')
376 %gemFigur('dybdeVarEnergiEps1')
377 %gemFigur('dybdeVarEnergiEps') %her er k ikke ændret til en
    funktion
```

```
1  function [ z,Temp, U, f ,E, MaxE, MaxD ] = konstantDybde(
       konstanter)
2  %For konstant dybde kan energihøst mm findes.
3  %   z er dybden. Temp er temperaturen afhængigt af dybden, U
       er
4  %   svømmehastigheden, f er fødetætheden, E er energihøsten,
       MaxE er
5  %   maksimal energihøst, og MaxD er dybden til den maksimale
       energihøst
6
7  %% Find svømmehastighed ud fra temperatur
8
9  %punkter
10 a=[17.5, 21.5];
11 b=[konstanter.Vmin,konstanter.Vmax];
12
13 %Linear regression. Svømmehastighed ud fra vand temperaturen
14 p=polyfit(a,b,1);
15 x=(0:1:25);
16 y=p(2)+p(1).*x;
17 %OVERVEJ OM DET SKAL ÆNDRET HVIS svomHastighed.m ER ET ANDET
       UDTRYK!!
18
19 %% Temperatur udvikling i dybden
20
21 z=linspace(0,700,1000);
22 %z=linspace(0,1200,1200);
23 Temp=ambientTemp(z,konstanter);
24 %Temp = konstanter.Tmax+((konstanter.p).*z-konstanter.A).*z
       .^2./(konstanter.zTcli+z.^2);
25
26 %% Fødetæthed
27 f=fodetaet(z,konstanter);
28
29 %Fødetætheden hvor der også er føde meget dybt nede.
30 %f =konstanter.fSS./(1+exp((-z+(konstanter.zSS))./konstanter
       .q));
31
32
33
34 %% Energihøst E=f*u
```

```
35
36 T=Temp;
37 %U=p(1)*T+p(2); % svømmehastighed
38 U=svomHastighed(T, konstanter);
39
40 E=f.*U;
41
42 %Maksimum
43 MaxE=max(E); %maksimal energihøst
44 [Y,I]=max(E);
45 MaxD=z(I); %dybden hvor er er maksimal enerihøst
46
47
48 end
```

```
1  function [ z,Temp, U, f ,E, MaxE, MaxD ] = konstantDybdeB(
       konstanter,z)
2  %For konstant dybde kan energihøst mm findes.
3  %   z er dybden. Temp er temperaturen afhængigt af dybden, U
       er
4  %   svømmehastigheden, f er fødetætheden, E er energihøsten,
       MaxE er
5  %   maksimal energihøst, og MaxD er dybden til den maksimale
       energihøst
6
7  % Da det er konstant dybde er tunens temp lig ambient temp.
8
9
10 %% Temperatur udvikling i dybden
11
12 Temp=ambientTemp(z,konstanter);
13
14 %% Fødetæthed
15
16 f=fodetaet(z,konstanter);
17
18 %% Svømmehastighed
19
20 U=svomHastighed(Temp, konstanter);
21
22 %% Energihøst E=f*u
23
24 E=f.*U;
25
26 %Maksimum
27 MaxE=max(E); %maksimal energihøst
28 [Y,I]=max(E);
```

```
MaxD=z(I); %dybden hvor er er maksimal enerihøst


end
```

```
function [t,X,GennemsnitE] = SvingSinus(L_new1, konstanter)
%%Funktion udregner den gennemsnitlige energihøst.
%L_new1 er den afstand som tunen svømmer op og ned
%t er tiden og X indeholder dybde, temperatur og energihøst.

konstanter.L = L_new1;

[t,X] = ode45(@(t,X) varDyb(t,X,konstanter), [0,konstanter.
    TidI], [konstanter.D0;konstanter.T0;konstanter.E0]);

%%
%Gennemsnitsværdi for energihøsten

%t og delta t
row1 = 100; %fordi løsningen i starten opfører "mærkeligt"
    er det t(100)

%antal perioder
n=floor(((max(t)-t(row1))/(2*pi))*konstanter.omega);

tDelta = n*(2*pi)/konstanter.omega; %det skal være et  helt
    antal perioder
t2 = t(row1)+tDelta;

%Rækken som er (delta t + t)
[a,row] = min(abs(t-t2));

%Gennemsnitsværdien for E
GennemsnitE = (X(row,3)-X(row1,3))/tDelta;

end
```

```
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68 %Phi= 2/10000.*sin(45*t);
69 %sPhi = sin(Phi);
70
71 %figure
72 %plot(t,sPhi)
73 %xlabel('t')
74 %ylabel('Dybde, z')
75 %gemFigur('svingning')
76
77
78
79 %Temperatur i dybden, z
80 % z=linspace(0,500);
81 % Ta = 27.5+((-1./100)-12).*D.^2./(100+D.^2);
82 %
83 % %Phi vinklen til tiden t
84 % Phi=(2/10000).*sin(45*t);
85 %
86 % %% De 3 koblede diff ligninger
```

```
% %Temperatur for tunen til tiden t
% diff(Tkrop,t)=k.*(Ta-Tkrop);
%
% %Dybden til tiden t
% diff(D,t)= (0.053.*D +0.481).*sin(Phi);


%Energihøsten til tiden t
%f=1./(1+exp((-D+300)./100));%Fødetætheden

%E=(0.053.*D + 0.481)*f;
```

```
function y = SvingLogiskWrapper(x, konstanter)
%Funktion er input til optimeringen af Tlow og Thigh

%konstanter.Dlinje = x(1);
konstanter.Tlow = x(1);
konstanter.Thigh = x(2);


[tl, Xl, gennemL] = SvingLogisk(konstanter);

y = -gennemL;
end
```

```

function [t, X, GenmEphi ] = SvingLogisk(konstanter)
%Funktionen udregner den gennemsnitlige energihøst.
% t er tiden og X indeholder, dybde, temperatur og
    energihøst.


[t,X] = ode45(@(t,X) varDybPhi(t,X,konstanter), [0,
    konstanter.TidI], [konstanter.D0;konstanter.T0;konstanter
    .E0]);

%% Gennemsnitsværdi for energihøsten
%t og delta t
row1 = 100; %fordi løsningen i starten opfører "mærkeligt"
    er det t(100)

%Find omega ud fra simuleringen:
kk=diff(sign(diff(X(row1:end,2))));
p=find(kk~=0);
pl=p(1:end-2);
ps=p(3:end);

```

```
ki=round(mean(ps-pl));

Tpe=t(row1+ki) - t(row1);

omega1=(2*pi)/Tpe;

%antal perioder
n=floor(((max(t)-t(row1))/(2*pi))*omega1);

tDelta = n*(2*pi)/omega1; %det skal være et  helt antal
    perioder
t2 = t(row1)+tDelta;

%Rækken som er (delta t + t)
[a,row] = min(abs(t-t2));

%Gennemsnitsværdien for E
GenmEphi = (X(row,3)-X(row1,3))/tDelta;

end
```

```
function Phi=svingPhi(D,T, konstanter)
%%Funktionen giver vinklen Phi, når tunen udfører logiske
    svingninger.
%D er dybde og T er tunens temperaturen.
%Det er linjen lSkil der er skillelinjen mellem de to zoner
    hvor tunen
%svømmer op eller ned.


Ttemp = [konstanter.Tlow, konstanter.Thigh];
Ddyb = [konstanter.Dlinje, 0];

p = polyfit(Ttemp, Ddyb,1); %grad=1

%Den rette linje der skiller de to zoner
lSkil = p(2) + p(1).*T; % det er tunens temperatur


if D < lSkil % da dybden bliver mindre op af y-aksen.

    Phi = -atan(D);
else

    Phi = atan(konstanter.Dlinje-D);

end
```

```
25
26 end
```

```
1  function Phi=sving(t,D, konstanter)
2  %Funktionen udregner vinklen phi når tunen udfører sinus
       svingninger.
3  %D er tunen dybde, t er tiden.
4
5  U=svomHastighed(D, konstanter);
6
7  Epsi = (konstanter.L*konstanter.omega)/(U);
8
9  Phi= Epsi*sin(konstanter.omega*t);
10
11 end
```

```
1  function dXdt = varDyb( t,X, konstanter )
2  %Udregner de afledte når tunen udfører sinus svingninger
3  %t er tiden, X indeholder dybde, temperatur og energihøst
4
5  %Variable
6  D = X(1);
7  T = X(2);
8  E = X(3);
9
10 %Henvisning til funktioner
11 Ta = ambientTemp(D, konstanter);
12 U = svomHastighed(T, konstanter);
13 f=fodetaet(D, konstanter);
14 Phi=sving(t,D, konstanter);
15 k = kTemp(T,D, konstanter);
16
17
18 % De afledte
19 dTdt = k*(Ta - T);
20 dDdt = U*sin(Phi);
21 dEdt = U*f;
22
23 % Samler de afledte
24 dXdt=[dDdt;dTdt;dEdt];
25
26 end
```

```
1  function f=fodetaet(D, konstanter)
2  %Funktionen udregner fødetætheden som en funktion af dybden
       D.
```

```
3
4  %Fødetæthed, hvor der også er føde meget dybt nede
5  %f=konstanter.f0+konstanter.fSS/(1+exp((-D+(konstanter.zSS))
       /konstanter.q));
6
7  %Fødetætheden som en gausklokke
8  f=konstanter.f0+konstanter.fSS.*exp(-0.5.*((D-konstanter.zSS
       ).^2)./konstanter.q^2);
9
10 end
```

```
1  function Ta=ambientTemp(D, konstanter)
2  %Funktionen udregner vandets temperaturudviklingen i forhold
       til dybden D
3
4  Ta=konstanter.Tmax+(konstanter.p.*D-konstanter.A).*D.^
       konstanter.Lambda./(konstanter.zTcli^konstanter.Lambda+D
       .^konstanter.Lambda);
5
6  end
```

```
1  function k = kTemp(T,D, konstanter)
2  %%k er tunens varmekorefficient, alt afhængigt af om tunen
       er på dybt eller
3  %%lavt vand.
4  %T er tunens temperatur, D er dybden.
5
6  Ta = ambientTemp(D, konstanter);
7
8  k = konstanter.klow+(konstanter.khigh-konstanter.klow)
       .*(1./(1+exp((T-Ta+konstanter.Tbb).*100)));
9
10 end
```

```
1  function U=svomHastighed(T, konstanter)
2  %Funktionen udregner tunens svømmehastighed som funktion af
       tunens
3  %temperatur
4
5  %U=T-5;
6  U=0.053.*T+0.481;
7  %Det er T og ikke Ta. Det er ikke som i konstant dybde
8  %hvor temperaturen udenfor er det samme som temperaturen
       inden i fisken
9
10
```

```
11 end
```

```
1  function U=svomHastighedSize(T, konstanter,Lb)
2  %Funktionen udregner tunens svømmehastighed som funktion af
       tunens
3  %temperatur og længde.
4  %Lb er tunens længde i meter
5
6  %U=T-5;
7  U=(0.053.*T+0.481).*Lb./0.74;
8  %Det er T og ikke Ta. Det er ikke som i konstant dybde
9  %hvor temperaturen udenfor er det samme som temperaturen
       inden i fisken
10
11
12 end
```

# Litteratur


[1] C.W. Clark and M. Mangel. *Dynamic state variable models in ecology: methods and applications*. Oxford University Press, USA, 2000.

[2] P.B. Dimitri. *Dynamic programming and optimal control*, volume 1. Athena Scientific Belmont, MA, 2000.

[3] K. Evans, NP Clear, T. Patterson, JS Gunn, and J. Hampton. Behaviour and habitat preferences of bigeye tuna (thunnus obesus) tagged in the western coral sea. *Migration and habitat preferences of bigeye tuna, Thunnus obesus, on the east coast of Australia. CSIRO Report FRPC*, 109:47–71, 1999.

[4] G. Ferretti. Cold and muscle performance. *International journal of sports medicine*, 13:S185, 1992.

[5] J.H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. Springer, 2002.

[6] Florida Fish and Wildlife Conservation Commission. Saltwater fish measurement guidelines. `http://myfwc.com/fishing/saltwater/recreational/fish-measurement/$#$fork"`, November 2012.

[7] Primary Industries Fishing and NSW Government Aquaculture. Bigeye tuna thunnus obesus. `http://www.dpi.nsw.gov.au/fisheries/recreational/saltwater/sw-species/bigeye-tuna`, Januar 2013.

[8] A. Gilat. *MATLAB: an introduction with applications*. Wiley, 2005.

[9] R. Grimshaw. *Nonlinear ordinary differential equations*, volume 2. Blackwell Scientific Publications, 1990.

[10] F.S. Hillier, G.J. Lieberman, and M. Hillier. *Introduction to operations research*, volume 6. McGraw-Hill New York, 2005.

[11] A.I. Houston and J.M. McNamara. *Models of adaptive behaviour: an approach based on state*. Cambridge University Press, 1999.

[12] J.B. Jørgensen, K. Madsen, H.B. Nielsen, and M. Rojas. Introduction to optimization and data fitting. *Informatics and Mathematical Modeling, Technical University of Denmark*, 2007.

[13] D.L. Kramer. The behavioral ecology of air breathing by aquatic animals. *Canadian Journal of Zoology*, 66(1):89–94, 1988.

[14] H Malte, C. Larsen, M. Musyl, and R. Brill. Differential heating and cooling rates in bigeye tuna (Thunnus obesus lowe): a model of non-steady state heat exchange. *J Exp Biol*, 210(Pt 15):2618–26, 2007.

[15] O. Maury. How to model the size-dependent vertical behaviour of bigeye (Thunnus obesus) tuna in its environment. *Collect. Vol. Sci. Pap, ICCAT*, 57(2):115–126, 2005.

[16] University of Florida Department of Mathematics. ode23, ode45. `http://www.math.ufl.edu/help/matlab/ode23.html`, 1994.

[17] World Register of Marine Species. Thunnus obesus (lowe, 1839). `http://www.marinespecies.org/aphia.php?p=taxdetails&id=127028`, January 2013.

[18] N.K. Poulsen. Dynamic optimization, optimal control. *Informatics and Mathematical Modeling, Technical University of Denmark*, 2012.

[19] D.W. Stephens and J.R. Krebs. *Foraging theory*. Princeton University Press, 1987.

[20] S. Strogatz. *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry and engineering*. Perseus Books Group, 2000.

[21] R.V.V. Vidal and H.F. Ravn. *Notes on static and dynamic optimization*. Imsor, 1977.

[22] Wikipedia. Bigeye tuna. `http://en.wikipedia.org/wiki/Bigeye_tuna`, December 2012.

[23] Wikipedia. Simplex. `http://en.wikipedia.org/wiki/Simplex`, Januar 2013.

[24] Y. Yazici. Operator splitting methods for differential equations. 2010.