University of Amsterdam

MSc Stochastics and Financial Mathematics

Master Thesis

Functional form based optimization for invasive species management

---

Author:
Nikki Levering

Supervisor:
prof. dr. M.R.H. Mandjes
dr. B. Patch
dr. K. Hock

Examination date:
August 29, 2019

Korteweg-de Vries Institute for
Mathematics

Abstract

We consider habitat networks where an invasive species has entered the network at a certain patch and is able to disperse through the network, colonizing other patches and thereby harming biodiversity. Methods to limit this colonization can often only intervene on certain patches, not on the network as a whole. The objective of managers is to minimize the risk of colonization of high-value patches, given a total allocation of intervention resources that can be divided between patches. The objective function in invasive species management models often has no closed-form expression and current simulation optimization techniques do not yield a fast approximation to the optimal solution. In this thesis a functional form based optimization method is described for the optimization of the invasive species model under study. This approach connects a detailed model of the system incorporating realistic features with a tractable approximate model that can be optimized by fast numerical optimization methods. Performance of this methodology is illustrated by means of an example: a model for crown-of-thorns starfish (COTS), a species behaving like an invasive species, on the Great Barrier Reef is constructed together with an appropriate approximation model. With the use of this approximation model the COTS model is optimized by means of this approach. A set of numerical experiments is conducted to compare the results in terms of optimality gap and computation time with known optimization techniques. These numerical experiments conducted on example networks show great promise, since the optimal value can be reached with a computation time improving on that of a stochastic approximation method.

Title: Functional form based optimization for invasive species management
Author: Nikki Levering, nikki.levering@student.uva.nl, 10787992
Supervisor: prof. dr. M.R.H. Mandjes, dr. B. Patch, dr. K. Hock
Second Examiner: dr. J.L. Dorsman
Examination date: August 29, 2019

Korteweg-de Vries Institute for Mathematics
University of Amsterdam
Science Park 105-107, 1098 XG Amsterdam
`http://kdvi.uva.nl`

Contents

1. Introduction

The biodiversity of natural environments is damaged by the introduction of invasive species to the ecosystem. Most natural environments can be viewed as habitat networks, since they are separated into geographically distinct habitat patches and individuals can migrate between patches. An invasive species enters the ecological system at a patch in the network and spreads through the network, harming the biological communities at the patches by e.g. predating on the native species. So, managing the biodiversity in environments requires decision making in the time and place of intervention to constrain the expansion of the invasive species. Most intervention methods will only reduce the expansion of an invasive species on a certain subset of patches in the habitat network, and thus not on the whole network. Since the resources for intervention are limited (e.g., the budget), this raises the question of where in the habitat network intervention resources should be committed.

An example that will be analyzed is the problem of crown-of-thorns starfish (COTS) outbreaks on the Great Barrier Reef (GBR). It should be noted that COTS have always been present on the GBR, making it a pest and not an invasive species. COTS do however behave as an invasive species, making it a relevant example for invasion-like dynamics, and one of the best documented of such dynamics in the marine ecosystems. COTS spread through the GBR and are environmentally harmful, since studies have shown that on the GBR, the world's largest coral reef system, COTS are among the most important causes of coral reef decline [21]. One of the reasons that COTS are one of the most important causes of reef decline are COTS outbreaks in which COTS densities on a reef can increase rapidly from low to high densities. Moreover, adult COTS have the ability to kill entire corals, where most other coral-predators only cause tissue-loss or injuries [31]. Noting that the GBR is $344,400\,\mathrm{km}^2$ and contains more than 3000 individual reefs [15], it can be concluded that eradication of COTS on the entire system is impossible. Currently intervention resources are dedicated to reducing COTS numbers though culls on individual outbreaks on reefs, so only a limited area is targeted by interventions. If the locations are chosen well then there is more hope of reducing the decline of coral cover. The Australian government generally spends annually around AUD $3 million on controlling the range expansion of COTS [27], and this sum has been substantially increased in the last year (K. Hock, personal communication, August 13 2019).

When COTS larvae arrive on a reef, they tend to settle and colonize it. Reefs which are currently free of COTS can be colonized through larval dispersal. COTS reproduce by spawning, and the larvae are dispersed to other geographically separate reefs via ocean currents. Some of the reefs have high economic (due to tourism, for example) or ecological value. The goal of the current study is to take actions that prevent these

'high-value' patches from being colonized by COTS. The interventions on reefs involve employing divers which inject adult COTS with detergent to kill them. Since COTS are a native pest species and as such cannot be legally eradicated, the true objective of controlling the range expansion of COTS will actually be to keep the COTS density on the high-value patches below a certain level. We will however approach the COTS management from the perspective of invasive species management, implying the studied objective to be eradication of all COTS from the high-value patches of the network.

Assume we can allocate a total effort among the patches and denote with x the control effort allocation vector. When a patch is allocated effort this implies intervention takes place on this patch and the growth of the COTS population is negatively affected by every unit of effort allocated to the patch. Define w_i as the value of patch i, with $w_i = 1$ for high-value patches and $w_i = 0$ otherwise. Let $Z_i(x, t)$ be an indicator variable on the event that adult COTS are present at patch i at the end of management interval t given a feasible control effort allocation x. Then the objective can be written as:

$$\min_{x \in \mathcal{X}} \max_i \; w_i \mathbb{E} Z_i(x, t). \tag{1.1}$$

Just as in the context of COTS outbreaks on the GBR, we will assume the objective in invasive species management to be of the form

$$\min_{x \in \mathcal{X}} \; f(x),$$

with $f(x) := \mathbb{E}F(x)$ a function $\mathcal{X} \to \mathbb{R}$ where $\mathcal{X}$ represents the set of feasible control decisions. The goal is to find the optimum x^*, i.e. $f(x^*) \leq f(x)$ for all $x \in \mathcal{X}$. The complexity of the problem can make it intractable, which means the problem is often considered solved if one can find x' such that $f(x')$ is close to optimal. There are two common approaches to solve this kind of problems: analytically based and simulation based.

Analytic methods use a surrogate model to solve the problem. So instead of the complex stochastic network, they consider an approximating network with objective

$$\min_{x \in \mathcal{X}} \; \bar{f}(x). \tag{1.2}$$

and solve this problem. When the surrogate model is a good approximation for the original network the solution will approximate the objective (1.1). This is also a limitation in the use of analytic methods: one has to choose a surrogate model that captures enough of the relevant features of the system and for which (1.2) is a good approximation for (1.1). When a surrogate model is chosen that doesn't incorporate adequate features, the analytic results might be misleading.

In the simulation-based optimization setting it is assumed that one can simulate the random variable $F(x)$ for which $f(x) = \mathbb{E}F(x)$ to obtain a sequence of samples $(\hat{f}(x^{(n)}), n \in \{1, \ldots, M\})$ for some $M \in \mathbb{N}$. These samples are then used to generate a sequence $(x^{(n)})$ for which $x^{(n)} \to x^*$ as $n \to \infty$. There are simulation methods for which convergence has been proven. A widely used simulation method for which this is the

case is stochastic approximation [3]. A limitation of stochastic approximation and other simulation-based optimization methods is however the fact that optimization suffers from high computational costs. Invasive species networks are often high-dimensional models, so performing simulation-based optimization becomes impractical. Furthermore, high computational costs make it infeasible to gain meaningful insights from the model (e.g., by performing sensitivity analysis).

Besides these two general methods there have been methods proposed for specific problems. For the problem of COTS on the GBR there have been some recent publications on proposed solutions. In [39] a linear, discrete time, stochastic dynamic connectivity model for marine species is considered. At every time step the number of COTS is multiplied with a birth rate, and this new number is multiplied with the probability of successful dispersal from patch i to patch j and the probability of post-settlement survival. With some simplifying assumptions the long-term stochastic growth rate is computed and bounded from above. The method presented by [39] has a few limitations. In the first place there is no mention of an objective function or a specific method on how to get population persistence. Secondly, in the model there is no dependence for the COTS on available coral. Lastly, there is no distinction in the different age classes for COTS, even though literature reveals that only larvae can disperse [31] and only adults (COTS of age 2+) can reproduce. This is not the case in [27], where equations are used to describe the discrete-time population dynamics and where both different age groups and the relation between coral and COTS is considered. However, in [27] the model is not a connectivity model and moreover does not consider the fact that model parameters may change over time which is considered important by biologists [39] . Another problem in the use of [27] is that just as in [39] there is only a model proposed, no specific solution.

A specific solution for the COTS problem on the GBR is given in [20]. In [20] the GBR is modelled as a network in which each reef is either colonized or uncolonized by COTS. The model incorporates dynamic connectivity by letting the configuration of the network not be deterministic. Every time step the ENZI value of every patch is computed, where ENZI stands for 'expected non-zero impact' and is defined as

$$\mathrm{ENZI}_t(S, N_i) = 1 - \prod_{i=0}^{k-1} [1 - r_t(N_i) v_t(N_i)],$$

with S an occupied source patch, N_i a set of empty patches under consideration, k the number of network nodes, $r_t(N_i)$ the probability that patch N_i is occupied at time t and $v_t(N_i)$ the perceived value of N_i at time t for the management objective. By determining the largest ENZI value of the different occupied source patches the patch of intervention can be determined. For simplicity and tractability this model parsimoniously still leaves out some key features inherent to stochastic habitat networks. Firstly the model does not account for the life-cycle of COTS or the predator-prey relation between COTS and coral, just as [39]. Moreover, the procedure is binary in that it only distinguishes between colonized and uncolonized, implying that it does not allow for density-dependent dispersal probabilities. Data does however show that dispersal is often density-dependent [2]. A final limitation we will mention, which is also a limitation in [39] but is incorporated

in [27], is that it does not account for carrying capacity.

Since both the analytic and simulation methods have their limitations, just like the specific approaches proposed for the COTS problem, our approach will be different. For solving (1.1) we will use a hybrid method similar to the one proposed in [12] and [30] for stochastic processing network applications, that uses techniques from both analytic-based and simulation-based methods. In [12] it is assumed that the objective function f has no known closed-form, but that in the set of potential actions the function can be evaluated by simulation of a random variable $F(x)$ for which $f(x) = \mathbb{E}F(x)$. Just as in the analytic-based method a set of functions $\{\tilde{f}(\cdot, \Lambda)\}$ is constructed which should contain elements that well approximate f. Assuming the existence of such a set of functions we now describe the in [12] presented sequential approach to finding Λ such that $\tilde{f}(\cdot, \Lambda) \approx f$. Given an initial $x^{(1)}$ we sample $F(x^{(1)})$ and then find the value of $\Lambda^{(1)}$ such that $f(x^{(1)}, \Lambda^{(1)}) = F(x^{(1)})$. With $\Lambda^{(1)}$ given $\tilde{f}(\cdot, \Lambda^{(1)})$ is optimized with respect to x. With this value of x we start the procedure again: we evaluate $F(x^{(2)})$ by means of simulation and decide on a value of $\Lambda^{(2)}$ for which $f(x^{(2)}, \Lambda^{(2)}) = F(x^{(2)})$. Then $\tilde{f}(\cdot, \Lambda)$ is optimized with respect to x. Iterating these steps gives a sequence $(\Lambda^{(n)}, n \in \mathbb{N}_0)$ with a corresponding sequence of functions $(\tilde{f}(\cdot, \Lambda^{(n)}), n \in \mathbb{N}_0)$. We want that $x^{(n)} \to x'$ for x' close to x^*. In [12] it is proven that this hybrid approach converges for a specific class of stochastic processing networks.

The approach as proposed in [12] needs to be adapted to our setting of invasive species problems. In this thesis we illustrate how the methodology can be used to guide real world intervention policies for the control of invasive species, by applying the methodology to the problem of COTS on the GBR. In order to apply the methodology for the specific problem of COTS on the GBR we first develop a model incorporating important real world parameters. Since it will be used in a functional form optimization framework this model purposefully favours inclusion of important ecological features over simplicity and incorporates many features absent in earlier discussed models of [39, 27, 20]. To counterbalance this complex and feature-rich model we will also develop a simpler tractable model. Our optimization approach balances between the strength of these models, leveraging the simple model for speed and the complex model for approximating the natural dynamics. We will test the method using some example networks and will show that it can outperform other relevant optimization methods.

The chapters are organized as follows. In Chapter 2 an outline of the COTS model and the objective are presented. To solve the objective a functional form approach as proposed by [12] and briefly discussed above will be used, a methodology that will be explained in Chapter 3. Chapter 4 is used to describe how to apply the methodology of [12] to our COTS model as presented in Chapter 2. In Chapter 5 three optimization algorithms are reviewed. Two optimization algorithms, backtracking line search and the NITRO algorithm, will be used in applying the functional form based methodology to the COTS model. One step of the functional form approach will consist of optimizing a function with constraints that can be evaluated by fast numerical methods and these two optimization algorithms are designed for this kind of optimization. The third optimization algorithm reviewed in Chapter 5 will be a stochastic approximation algorithm and this algorithm will be used to get an indication of how well the functional form based al-

gorithm performs, by comparing the results of the functional form based algorithm with the results of the reviewed stochastic approximation algorithm. In Chapter 6 computational experiments are performed, both to verify if the methodology gives the expected outcomes and moreover to compare the methodology to the stochastic approximation algorithm presented in Chapter 5. Chapter 6 also illustrates the usefulness of being able to perform fast optimization by describing how to perform sensitivity analysis on the model parameters. In Chapter 7 both concluding remarks and possible future research interests are given.

2. Detailed invasive species network model

The aim is to illustrate how the methodology of functional form based optimization can be used in real life management of invasive species and pests. This illustration will be done by means of an explicit example, which as described in the introduction will be the problem of controlling crown-of-thorn starfish (COTS) outbreaks in the Great Barrier Reef (GBR). In this chapter a COTS model incorporating important real world parameters is presented together with the objective. Since our main focus will be on the description of the methodology, the COTS model as presented below is not an attempt to develop a new model of COTS dynamics on the GBR. The model was, however, created with the intention of resembling the real life situation and can be considered as a reasonable representation of the ecological dynamics as all features presented in the model have references in studies on COTS or the GBR. The purpose of the model is to be used as an example proof of concept for the use of functional form optimization for invasive species management. How functional form optimization, a methodology that will be described in Chapter 3, can be used to solve the objective as presented in Section 2.2 will be explained in Chapter 4.

2.1. Outline COTS model

The COTS model is a metapopulation model in which each reef is a habitat. Let $\mathcal{N} = \{1, \dots, N\}$ be the set of patches with $N = 3806$ [21] and $\mathcal{I}$ the set of high-value patches. The number of patches in $\mathcal{I}$ depends on the criteria set for patches to be of high-value. In [20] 18 patches are considered high-value patches, where they are considered high-value since they belong to a small region of the GBR where COTS outbreaks are usually first detected and moreover are valuable for tourism. The population sizes at each patch at time t will be denoted by a vector $n(t) = (n_1(t), \dots, n_N(t))$ with

$$n_i(t) = (a_i(t), j_i(t), l_i(t), c_i(t))^T \quad \forall i \in \mathcal{N},$$

and state space $\mathcal{S}^N$ with $\mathcal{S} = \mathbb{Z}^4_{\geq 0}$. Here $a_i(t), j_i(t)$ and $l_i(t)$ are the relative densities of adults, juveniles and settler (age-0) COTS measured in a unit of space on patch i at time step t respectively and $c_i(t)$ the coral cover per m^2 on patch i. It should be noted that everything in the model is subject to uncertainty, since despite intense current and past research efforts many parameters concerning COTS and coral ecology remain poorly known and speculative, but the model should nevertheless be considered useful for insights.

Feature	Reference(s)
Three age groups need to be considered	[27, 11]
Larvae are able to migrate	[20, 21, 39, 37]
COTS larvae are buoyant and passive	[21, 31]
Adults don't migrate	[20, 27]
COTS spawn, with spawning peak around January	[27, 4, 24]
Coral reproduces sexually and asexually	[19]
Connectivity between colonies varies randomly in time	[20, 24, 38, 27, 39]

Table 2.1.: Features incorporated in the COTS model

Since the number of patches is high and there are multiple events possible on every patch, simulation of a continuous model would be too time-consuming. Moreover, conducting studies to estimate rates in a continuous model is much harder and in our model not realistic, since COTS activity is hard to study and a lot of questions on COTS activity are still unresolved [31]. Thus the model as presented will be a discrete-time connectivity model. Yearly time-steps will be used, since they are adequate to represent major events in the life-cycle of the COTS. COTS become sexually mature and reproduce when about two years old [27], which they do via spawning - releasing their gametes into the water column where the fertilization occurs. Studies have shown that there is a peak in the spawning of COTS during the Australian summer (November-February) [4]. The fertilized eggs develop into larvae and are then dispersed among reefs by wind and ocean currents. After 11-22 days, the larvae settle if they encounter an appropriate reef habitat [27]. Since the spawning period occurs seasonally the time steps are chosen to be per year and represent the period between two Australian summers.

Since there is a predator-prey relation between COTS density and coral cover, in the development of the connectivity model for the COTS outbreaks both the density of COTS and coral cover are modelled. Three main age groups for COTS are considered: settlers (year 0, also called larvae in the first two weeks), juveniles (year 1) and adults (year 2+). Motivation for considering these three age groups is the fact that adults (year 2+) are capable of reproduction, while settlers and juveniles are not [27]. Moreover, after birth, larvae have a probability of dispersal to other patches, due to ocean currents that cause larval dispersal [21]. Studies have shown that juveniles and adults are also capable of migrating to other patches of habitat within the same reef, though the extent at which they will actually migrate between reefs is inconclusive. Adult COTS in good condition and with enough coral to feed on will have little reason to move, whereas adult COTS that are starving are probably unlikely to succeed in travelling between two reefs with a large distance between them [31]. Due to this we assume that juveniles and adults do not move between reefs and the only movement of COTS between reefs is due to larval dispersal.

For the age groups of COTS considered, there are three main natural events on each patch: reproduction, migration and death. Features of these events incorporated in the model are given in Table 2.1, together with their references. Defining $A_i(t), J_i(t), L_i(t)$

as random variables denoting the density of adult COTS, the density of juvenile COTS and the density of COTS settlers measured in certain units of density and $C_i(t)$ as the coral cover respectively on patch $i \in \mathcal{N}$ at time step t, the model can be written as (with the notation for stochastic metapopulation models as used in e.g. [14]):

$$\begin{aligned} A_i(t) &= S_i^a(t,x) + S_i^j(t,x), \\ J_i(t) &= S_i^l(t), \\ L_i(t) &= X_i(t), \\ C_i(t) &= S_i^c(t) + X_i^c(t), \end{aligned}$$

with

$$\begin{aligned} S_i^a(t,x) &\sim \text{Bin}(A_i(t-1), p_i^a(t,x,B(t),C_i(t-1))), \\ S_i^j(t,x) &\sim \text{Bin}(J_i(t-1), p_i^j(t,x,B(t),C_i(t-1))), \\ S_i^l(t) &\sim \text{Bin}(L_i(t-1), p_i^l(t,B(t),C_i(t-1))), \\ X_i(t) &\sim \text{Poisson}_{K_i - N_i(t-1)}(\sum_{j=1}^{N} f_{ji}(A_j(t-1),B(t),t)), \\ S_i^c(t) &\sim \text{Bin}(f_i^c(C_i(t-1)), p_i^c(t,B(t),A_i(t-1))), \\ X_i^c(t) &\sim \text{Poisson}_{K_i^c - C_i(t-1)}(\sum_{j=1}^{N} f_{ji}^c(C_j(t-1),B(t),t)). \end{aligned}$$

where $\text{Poisson}_m(\lambda)$ is a mean-λ Poisson distribution truncated at $m \in \mathbb{N}$ and

$$N_i(t-1) = A_i(t-1) + J_i(t-1) + L_i(t-1)$$

denotes the total COTS density on patch i at time step $t-1$. Descriptions of the parameter functions are given in Table 2.2. The random variables $S_i^a(t,x), S_i^j(t,x)$ and $S_i^l(t)$ respectively represent the relative density of adult, juvenile and settler COTS present on patch i that survive time step t given a decision to allocate effort x. The quantities S_i^a and S_i^j depend on x since the intervention will change the survival probabilities for adults and juveniles, whereas S_i^l is not dependent on x since age-0 COTS will be too small to be eradicated by divers [40] (see also Remark 2.2.1). The quantity $S_i^c(t)$ represents the coral cover either surviving time step t or increasing the cover at time step t by means of asexual reproduction. $X_i(t)$ and $X_i^c(t)$ respectively represent the growth of settler density and the growth of coral cover on patch i by dispersal at time step t. Since the amount of suitable habitat available on any individual patch is limited, both coral cover and COTS density have a carrying capacity at every patch, denoted K_i for COTS and K_i^c for coral at patch i. Now $X_i(t)$ and $X_i^c(t)$ are modelled as truncated Poisson random variables, truncated at the leftover capacity $K_i - N_i(t-1)$ (or $K_i^c - C_i(t-1)$ in case of coral cover) of patch i. A distinction is made between $X_i^c(t)$ and $X_i(t)$, since it is not necessarily true that for instance the dispersal probability to i is the same for COTS and coral larvae. This is due to differences in the timing of spawning as well as life history

Parameter function	Description
$p_i^a(\cdot)$	Survival probability adult COTS patch i
$p_i^j(\cdot)$	Survival probability juvenile COTS patch i
$p_i^l(\cdot)$	Survival probability COTS settlers patch i
$p_i^c(\cdot)$	Survival probability of coral cover patch i
$f_i^c(\cdot)$	Internal growth rate coral cover
$f_{ji}(\cdot)$	Mean COTS settlers dispersing from j to i when unlimited capacity
$f_{ji}^c(\cdot)$	Mean COTS settlers dispersing from j to i when unlimited capacity
$b_i(\cdot)$	Reproduction rate COTS on patch i
$b_i^c(\cdot)$	Sexual reproduction rate coral on patch i
$p_{ij}(\cdot)$	Dispersal probability from i to j for COTS settlers
$p_{ij}^c(\cdot)$	Dispersal probability from i to j for coral larvae
$p_{ij,d}(\cdot)$	Survival probability of COTS dispersing from i to j
$p_{ij,d}^c(\cdot)$	Survival probability for coral dispersing from i to j

Table 2.2.: Descriptions of parameter functions COTS model.

characteristics (e.g., time spent in the water, development rate, etc.) of the settlers (K. Hock, personal communication, August 13 2019). The process $(B(t), t \in \mathbb{N}_0)$ represents the background process, which contains information on all the ecological factors at time step t, e.g. temperature and salinity. The background process $(B(t), t \in \mathbb{N}_0)$ contains the information about all these factors and can thus be seen as a vector which at time step t gives values for e.g. the temperature and salinity on all reefs.

The functions f_{ji} and f_{ji}^c respectively represent the mean larval density and coral density dispersing from patch j to patch i at a certain time step when there would be unlimited capacity. They are given by the expressions

$$f_{ji}(t) = A_j(t-1) \cdot b_j(B(t), t) \cdot p_{ji}(A_j(t-1), B(t), t) \cdot p_{ji,d}(B(t), t),$$
$$f_{ji}^c(t) = C_j(t-1) \cdot b_j^c(B(t), t) \cdot p_{ji}^c(C_j(t-1), B(t), t) \cdot p_{ji,d}^c(B(t), t),$$

where the descriptions of the parameter functions can again be found in table 2.2. For these parameter functions and the ones previously presented explicit expressions or estimations are needed, but it is outside the scope of this thesis to present these explicit expressions or estimates for the parameter functions. Given that the factors of the parameter functions covary, exhibit thresholds and vary over time and across space, estimating the parameters is a study on its own. Multiple studies on the dispersal probabilities have already been conducted, resulting in the program Connie 3. Further details of this research can be found in Appendix A.

2.2. Objective

As mentioned in the introduction, the management objective is to prevent the COTS from colonizing high-value patches. Controlling the expansion of COTS is performed by trained divers manually eradicating COTS.

Remark 2.2.1. Since there are thousands of unknown COTS possibly existing on each reef, many inaccessible to control efforts, it is unknown whether the control efforts as currently applied do have any sort of impact on COTS populations beyond perhaps very localized reduction. In the rest of the thesis we will however assume that the control efforts are having an impact and reduce COTS population growth.

If one would perform COTS control on the high-value patches by attempting to minimize the total risk of colonization by adult COTS for the high-value patches, it could happen that some of the high-value patches have a high risk just to reduce the expectation of others. To prevent this, it will be assumed that the management objective is to minimize the maximum risk of adult COTS colonization on the high-value patches at a certain time step t. Thus, in a mathematical framework, the objective function f is given by

$$f(x) = \max_i \ w_i \mathbb{E} Z_i(t, x),$$

where, as defined in the introduction, w_i is the value of patch i, with $w_i = 1$ for high-value patches and $w_i = 0$ otherwise and $Z_i(t, x)$ is an indicator variable on the presence of adult COTS at time step t given a decision to allocate effort x:

$$Z_i(t, x) = \begin{cases} 0 & \text{if } A_i(t) = 0 \text{ given x} \\ 1 & \text{otherwise.} \end{cases}$$

Here $x = [x_1, \ldots, x_N]$ is a vector and x_i the allocated effort to patch $i \in \mathcal{N}$. The allocated effort rate is measured in a certain units of human hours and it will be assumed that if patch $i \in \mathcal{N}$ is allocated a rate x_i, this will decrease the survival probability of adults and juveniles on patch i with x_i, the result having a lower bound of 0. Thus, p_i^a and p_i^j will for all $1 \leq i \leq N$ be estimated and then x_i will be subtracted (with lower bound 0) from this estimate. The objective function has to be minimized with respect to x subject to the constraint $x \in \mathcal{X}$ subject to a total budget for intervention activities across the entire network. The objective in this situation can be expressed as

$$\begin{aligned} \min \quad & \max_i \ w_i \mathbb{E} Z_i(t, x) \\ \text{subject to} \quad & 1^T x \leq C \end{aligned} \tag{2.1}$$

or equivalent as

$$\begin{aligned} \min \quad & \max_i \ w_i \mathbb{P}_x(A_i(t) > 0) \\ \text{subject to} \quad & 1^T x \leq C, \end{aligned}$$

where 1 is a column vector of length N consisting of ones. An optimal effort allocation x^* is an effort allocation solving the objective, so x^* is defined as

$$x^* = \operatorname*{argmin}_{x : 1^T x \leq C} f(x)$$

By the complexity of the presented COTS model the objective (2.1) does not have a closed-form expression. As argued in Chapter 1, earlier presented approaches in solving the objective have their limitations. We will present a hybrid method similar to the one proposed in [12] and [30] for stochastic processing network applications and adapt it to our setting of invasive species problems to solve the objective. In the next chapter first an overview of the hybrid method used in [12] and [30] will be given in Section 3.1, after which Section 3.2 will explain how to adapt this method to our setting of invasive species models.

Remark 2.2.2. We assume that the decision on where to employ the effort is made every year. Since optimizing the model for $t = 1$ does only output a management decision that minimizes the risk of colonization for the high-value patches the next year and not in the future, this will not be our suggested strategy. Our suggested strategy is optimizing the model for a time t in the future (say e.g. $t = 15$) and use the effort allocation as outputted by the optimization. In the year after, the same procedure can be used, so given the current effort allocation the optimization algorithm is ran with a time t in the future as input resulting in a new effort allocation.

3. The functional form approach

The functional form approach was first presented by Dieker et al. [12] for stochastic processing networks (SPNs) and later expanded by Patch et al. [30] to SPNs with blocking. The method can be used to optimize systems that have no known closed-form objective function but for which objective function values can be evaluated using simulation. It relies on a class of non-complex functions believed to approximate the objective function being available. The idea is to optimize these functions that approximate the objective function instead of the objective function itself. In this chapter we will first give an outline of the functional form approach and then we will describe how to adapt the method to the context of the problem of managing COTS on the GBR such that the methodology can be used to solve the objective presented in Chapter 2.

3.1. Overview functional form approach

In this section we will review the functional form based optimization approach presented in [12] and [30]. The approach was developed to optimize an objective function

$$f : \mathcal{X} \to \mathbb{R},$$

with $\mathcal{X}$ the set of feasible parameters, for which no closed-form expression is known. It is however assumed that for a given input vector $x \in \mathcal{X}$ the objective function f can be evaluated by simulation of a random variable $F(x)$ for which $\mathbb{E}F(x) = f(x)$. Thus, to evaluate $f(x)$ the random variable $F(x)$ is sampled, where the sample will be denoted as $\hat{f}(x)$. The context we are looking at is one where generating samples takes a long time, which implies that optimization routines using many of these samples are inefficient.

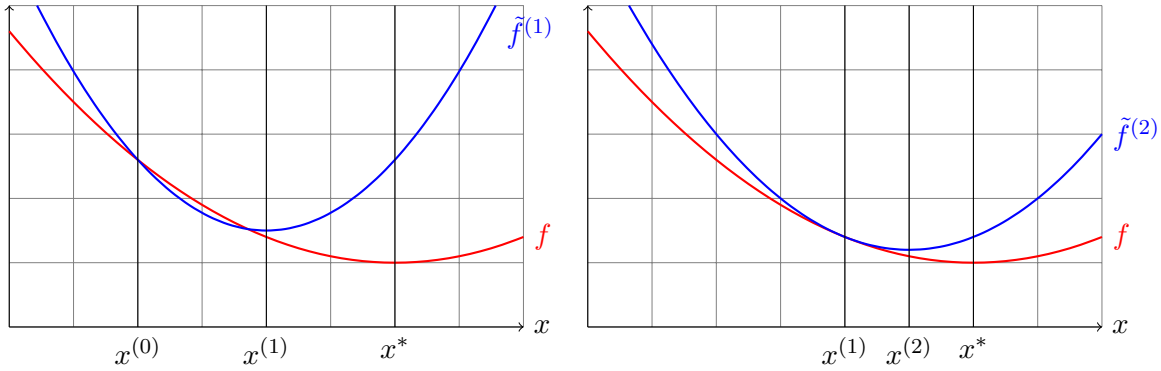

Figure 3.1.: Two iteration steps of the functional form based optimization algorithm.

An explicit example of an objective function without known closed-form expression was given in Chapter 2, in which we argued that the objective for the management of COTS on the GBR is given by

$$\begin{aligned} f(x) = &\max_i \; w_i \mathbb{E} Z_i(t,x) \\ \text{subject to} \quad & 1^T x \leq C, \end{aligned}$$

where 1 is a column vector of length N consisting of ones, w_i the value of the patch with $w_i = 1$ if patch i is high-value and 0 otherwise and $\mathbb{E}Z_i(t,x)$ representing the probability of adult COTS colonization on patch i. Since this expectation depends on multiple environmental factors (temperature, ocean currents, etc.), there is no closed form expression of $\mathbb{E}Z_i(t,x)$ in terms of x and thus no closed form expression of f. However, given the choice of x, f can be evaluated by simulation of a random variable $F(x)$ for which $\mathbb{E}F(x) = f(x)$. In our COTS model the random variable $F(x)$ will be chosen to be a crude Monte Carlo estimator, which we can sample using the COTS model presented in Section 2.1.

The goal is to optimize the function f and thus to determine the solution

$$x^* = \operatorname*{argmin}_{x \in \mathcal{X}} f(x),$$

which functional form based optimization does by efficient approximation. The methodology is an iteration which starts with an initial guess $x = x^{(0)}$ (chosen random or based on expert opinion) and assumes that a local fit $\tilde{f}^{(1)}$ of f is available around $x^{(0)}$, where $\tilde{f}^{(1)}$ is a function that can be evaluated by fast numerical methods or is of closed-form. Optimization of $\tilde{f}^{(1)}$ then gives the next iterate $x^{(1)}$. Since $\tilde{f}^{(1)}$ is a local fit of f around $x^{(0)}$, optimizing $\tilde{f}^{(1)}$ moves the iteration in the right direction. Furthermore, since $\tilde{f}^{(1)}$ can be evaluated by fast numerical methods or is of closed form, optimization of $\tilde{f}^{(1)}$ will be significantly faster than the optimization of f. Assuming there is a function $\tilde{f}^{(2)}$ which

- is a local fit of f around $x^{(1)}$,
- can be evaluated by fast numerical methods or is of closed-form,

the procedure can now be repeated: $\tilde{f}^{(2)}$ is optimized to find the next iterate $x^{(2)}$, which should again be a move in the direction of x^*. In Figure 3.1 these first two steps are drawn for a possible function $f : \mathbb{R} \to \mathbb{R}$ and initial value $x^{(0)}$. Iterating the procedure we get a sequence $\{x^{(n)}\}$. In [12] they prove that in a specific SPN the sequence $\{x^{(n)}\}$ converges for $n \to \infty$.

The methodology as explained above relies on the fact that for every x there exists a simple function $\tilde{f}$ that is a local fit of f around x. In functional form based optimization the knowledge of the structure and the behaviour of the network are used in defining a function $\tilde{f}(\cdot, \Lambda)$ which depends both on the network parameter x and an additional parameter Λ. Λ is either a scalar or a matrix and is used as auxiliary variable: for

every x the parameter Λ is chosen such that $\tilde{f}(\cdot, \Lambda)$ fits f well around x. Thus, given an initial $x^{(0)}$, a parameter value $\Lambda^{(1)}$ is chosen such that $\tilde{f}^{(1)} := \tilde{f}(\cdot, \Lambda^{(1)})$ fits f well locally around $x^{(0)}$. Then, as the procedure described above, $\tilde{f}^{(1)}$ is optimized to get the next iterate $x^{(1)}$. Given $x^{(1)}$ a parameter value $\Lambda^{(2)}$ is chosen such that $\tilde{f}^{(2)} := \tilde{f}(\cdot, \Lambda^{(2)})$ fits f well locally around $x^{(1)}$. $\tilde{f}^{(2)}$ is optimized to get the next iterate $x^{(2)}$, etc.

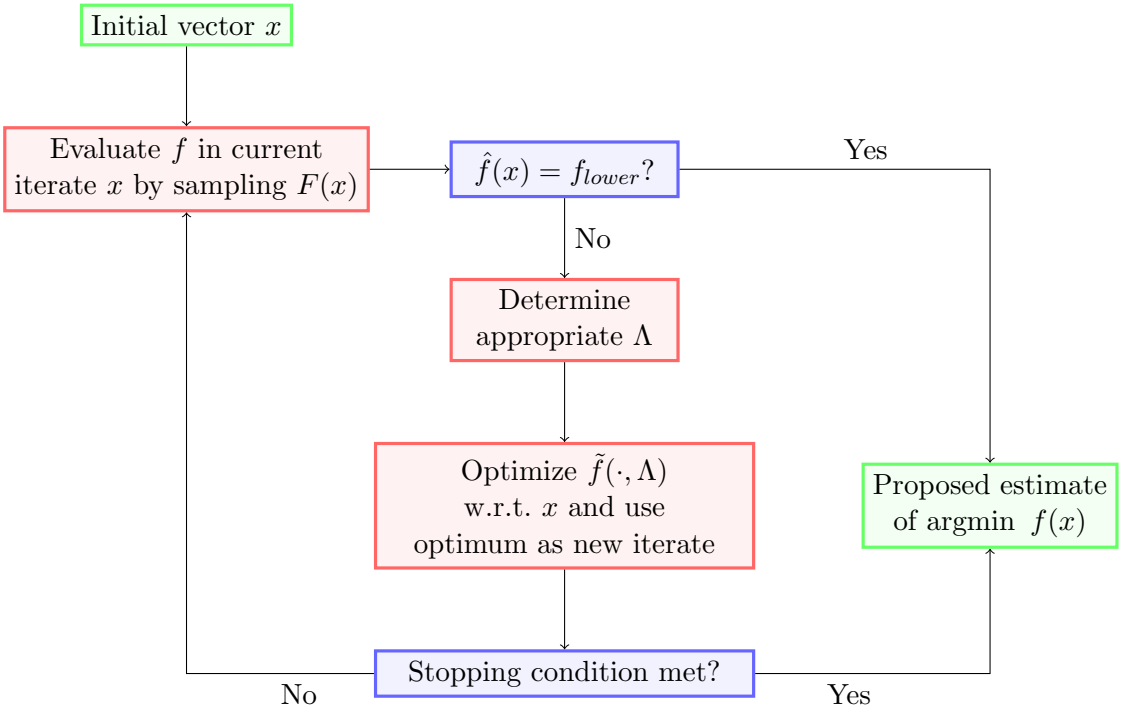

Figure 3.2.: Scheme for the use of surrogate models in the functional form approach

Remark 3.1.1. Determining a value of Λ such that $\tilde{f}(\cdot, \Lambda)$ approximates $f(\cdot)$ well around a given x can be done in different ways, but it is assumed that an evaluation of the value of f in x is used. In [30] an appropriate Λ is found solving

$$f(x) = \tilde{f}(x, \Lambda).$$

Since $f(x)$ can only be evaluated by a sample of the random variable $F(x)$, denoted as $\hat{f}(x)$, this equation becomes

$$\hat{f}(x) = \tilde{f}(x, \Lambda).$$

In our application of functional form based optimization a different method is used in determining a proper Λ (Section 4.1), which also uses one sample of the random variable $F(x)$ in each iteration. The fact that only one sample $\hat{f}$ per iteration is used to determine the right direction is what makes the functional form based optimization algorithm far

more efficient than purely simulation-based methods, which need more simulation values for the approximation of the Jacobian for determination of the direction.

In our implementation the iteration of functional form based optimization is repeated until there is an $n \in \mathbb{N}$ such that $||x^{(n+1)} - x^{(n)}|| < \epsilon$ for some pre-specified $\epsilon \in \mathbb{R}_{>0}$ or otherwise if $n > \tilde{N}$ for some $\tilde{N} \in \mathbb{N}$. Since the objective function in our study will be lower bounded (the expectation of an indicator function is a probability and thus bounded below by 0), the iteration will also be stopped when this lower bound f_{lower} is reached. The outline of the iterative algorithm can thus be described as in Algorithm 1, the dynamics are shown in Figure 3.2.

Result: approximation of optimizer x^* of f
Initialization: $n = 1$, choose $\epsilon > 0$, $\tilde{N} \in \mathbb{N}$ and $x = x^{(0)}$;
while $n < \tilde{N}$ *and* $||x^{(n+1)} - x^{(n)}|| < \epsilon$ **do**

(i) Evaluate the sample $\hat{f}(x^{(n-1)})$;

if $\hat{f}(x^{(n-1)}) = f_{lower}$ **then**

break

end

(ii) Determine $\Lambda^{(n)}$ using $\hat{f}(x^{(n-1)})$

(iii) Compute

$$x^{(n)} = \underset{x \in \mathcal{X}}{\operatorname{argmin}}\, \tilde{f}(x, \Lambda^{(n)})$$

(iv) $n = n + 1$

end

Algorithm 1: Functional form based optimization.

Remark 3.1.2. As argued in [30] $\tilde{f}(\cdot, \Lambda^{(n)})$ is chosen such that it fits f well around $x^{(n-1)}$, which makes it likely that the algorithm moves in the right direction in each iteration step. Since the direction is only determined by the shape of $\tilde{f}(\cdot, \Lambda^{(n)})$, this argument implies that $\tilde{f}(\cdot, \Lambda^{(n)})$ should only fit the shape of f well around $x^{(n-1)}$. Thus, only the shapes of $\tilde{f}(\cdot, \Lambda^{(n)})$ and f should align around $x^{(n-1)}$. This can also be noted from the use of the functional form based optimization method in Figure 3.3, in which $\tilde{f}(\cdot, \Lambda^{(n)})$ does not fit f well locally, but both functions have aligned shapes. Therefore, when in the remainder of this thesis it is mentioned that $\tilde{f}$ fits f, it is implied that the shapes of $\tilde{f}$ and f align.

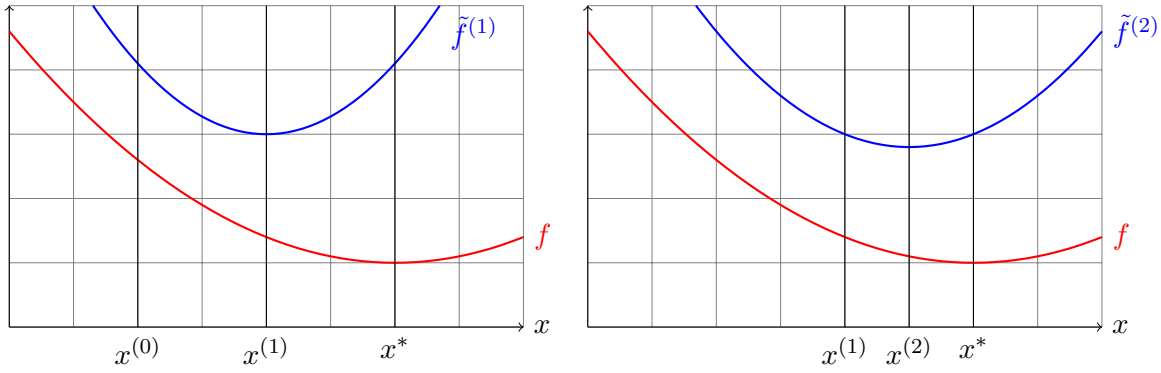

Figure 3.3.: Two iteration steps of the functional form based optimization algorithm.

3.2. Functional form for invasive species models

Before the functional form approach can be applied to an invasive species model, we have to find a function family $\{\tilde{f}(x,\Lambda)\}_\Lambda$ containing elements that approximate our objective function f. The closed-form expression of f is unknown, so the question becomes how one can find an appropriate $\{\tilde{f}(x,\Lambda)\}_\Lambda$ only using the stochastic invasive species model. One method to do this, the method we will apply, is by the use of *surrogate models*. A surrogate model or metamodel is a model that mimics the model of study, but is cheaper to evaluate computationally. Since it mimics the model of study the behaviour of the output $\tilde{f}$ of the surrogate model will be close to the behaviour of the output f of the model of study. Then, instead of optimizing the original model, we can optimize the surrogate model with respect to x. We will give the surrogate model an extra parameter, Λ. As explained in the previous section, this parameter Λ will be used to tune the surrogate model and make it a good fit for the original model for given values of x.

With a detailed and surrogate model given, we start by getting the parameters and initial values from the data. The parameters are used both for the invasive species model and the surrogate model. With an initial vector x chosen the steps of the functional form based optimization algorithm are now as follows. Simulation of the invasive species model is used to obtain a sample $\hat{f}(x)$ for the initial vector x. From this simulation the parameter Λ for the surrogate model is determined. With this value of Λ the objective function of the surrogate model is optimized, which can be done by fast numerical methods by the simplicity of the surrogate model. This optimization gives a new vector x which is used in a new simulation run of the detailed model. This loop is now iterated until one of the stopping conditions is met.

In Chapter 2 we presented both a COTS model and an objective function for controlling the range expansion of COTS. In this chapter an outline of the functional form based optimization algorithm was given, as presented in [12] and [30]. The goal of this thesis is to show how to use the functional form based optimization algorithm in solving the objective for the management of COTS on the GBR given by (2.1). The remaining

chapters of this thesis are dedicated to the use of the functional form based optimization algorithm on this objective. To use the algorithm first an appropriate surrogate model needs to be presented, which will be done in Chapter 4. Chapter 4 will also be used to explain how to determine an appropriate Λ to tune the surrogate model such that the objective function $\tilde{f}$ fits f around a given x. Chapter 5 discusses how the optimization of the surrogate model can be performed to find a new iterate x in the functional form based optimization algorithm. With these two steps of the algorithm explained, the algorithm can be performed. Results of the performance of the functional form based optimization algorithm in solving the objective (2.1) are presented in Chapter 6.

4. Finding the appropriate functional form

In the previous chapter an outline of the functional form based optimization algorithm was given, as presented in [12] and [30]. In this thesis we illustrate how this methodology can be used to guide real world intervention policies for the control of invasive species, by applying the methodology to the problem of COTS on the GBR. The methodology relies on the fact that for every x there exists a non-complex function $\tilde{f}$ that is a local fit of f around x. In this chapter we will present a class of non-complex functions $\tilde{f}$ to fit the objective function f of COTS management given in (2.1) by the use of a surrogate or approximate model. This chapter will also be used to explain how to determine an appropriate Λ to tune the surrogate model such that the objective function $\tilde{f}$ fits f around a given x.

4.1. The approximate model

The approximate model, with $\Lambda = (\lambda_{ij})$ a matrix, will be given by

$$\begin{aligned} A_i(t) &= S_i^a(t,x) + S_i^j(t,x), \\ J_i(t) &= S_i^l(t), \\ L_i(t) &= \sum_{j=1}^{N} X_{ji}(t), \end{aligned}$$

with

$$\begin{aligned} S_i^a(t,x) &\sim \text{Bin}(A_i(t-1), p_i^a(t,x,B(t))), \\ S_i^j(t,x) &\sim \text{Bin}(J_i(t-1), p_i^j(t,x,B(t))), \\ S_i^l(t) &\sim \text{Bin}(L_i(t-1), p_i^l(t,B(t))), \\ X_{ji}(t) &\sim \text{Poisson}(\lambda_{ji} A_j(t-1)), \end{aligned}$$

where $X_{ji}(t)$ is a random variable representing COTS settlers dispersing from patch j to contribute to the population growth on patch i at time step t and λ_{ji} a rate for the expected contribution from j to i per unit of population density at patch j. As can be seen this model is closely related to the model presented in Chapter 2. There are however two important differences. First, this model does not consider coral cover as a factor in the survival probabilities. Second, the model assumes a linear input of $A_j(t-1,x)$ as parameter in the Poisson process, which we assume is non-truncated. Both changes

were made to fasten the process of computing the parameters and thus the evaluation of the objective function.

This approximation model only partially overcomes the problems encountered in optimization of the invasive species model under study, since there is still no closed-form expression of $\mathbb{E}Z_i(t,x)$ in this approximation model. One can estimate $\mathbb{E}Z_i(t,x)$ by multiple model runs of the proposed approximation model, as is done in the original model. This will however significantly enlarge the computational costs of the approximate model, while speed is important. It turns out that, even though $\mathbb{E}Z_i(t,x)$ can only be estimated by multiple model runs, the moments of $\mathbb{E}A_i(t)$ given x can be computed explicitly. With the first moment we can define

$$\tilde{f}(x,\Lambda) = \max_i \ \mathbb{E}_x A_i(t). \tag{4.1}$$

Of course, the value of $\tilde{f}(\cdot,\Lambda)$ will then be a maximum expectation, while the value of $f(\cdot)$ will be a maximum probability. We do however keep in mind, as explained in Chapter 3, that only the shapes of f and $\tilde{f}(\cdot,\Lambda)$ need to match around a certain x to guide the optimum in the right direction. It is believed that the shapes of $\tilde{f}(\cdot,\Lambda)$ and $f(\cdot)$ will align, since a small colonization probability indicates that in a lot of simulation runs there will be no adults present, which indicates a low expected adult COTS density.

Remark 4.1.1. The objective function $\tilde{f}$ of the approximate model only uses the location of $A_i(t)$ and not the spread. However, the objective function f of the detailed COTS model is a probability and therefore depends both on location and spread. This implies that in the case of high (co)variance the shapes of $\tilde{f}$ and f will not be aligned and thus that other choices of the objective function $\tilde{f}$ based on both location and spread might improve results. Since such an objective function $\tilde{f}$ has not yet been found, our proposed approximate model will use (4.1) as objective function. In the computations below it will however be shown how to compute the covariance matrix of $A_i(t)$, such that one can use the covariance matrix once an appropriate objective function $\tilde{f}$ is constructed depending both on location and spread.

With the approximate model as presented the mean and variance can be computed exact with repeated use of the tower property and matrix difference equations. It will however turn out that implementation of these computations is rather slow, given that the matrices used in these matrix difference equations are quite large. For this reason a second method, computing the mean and variance in an approximate way using a MGF, will also be presented and this method will be used in the implementation of the approximate model. The exact computations will however be shown for completeness.

Remark 4.1.2. The computations in the rest of the chapter will depend on x, but since we will assume x to be given, x will be omitted in the notation.

Define the mean vector at time step t as

$$\mu_t := (\mathbb{E}A_1(t), \mathbb{E}A_2(t), \ldots, \mathbb{E}A_N(t))^T.$$

Define furthermore the filtration $\mathbb{F} = (\mathcal{F}_t)_{t\in\mathbb{N}_0}$ with the σ-algebras $\mathcal{F}_t$ defined as

$$\mathcal{F}_t = \sigma((A_k(s), J_k(s), L_k(s)), \quad \forall 1 \le k \le N, s \le t).$$

Use of the tower property now yields

$$\begin{aligned}
&\mathbb{E}A_i(0) = A_i(0)\\
&\mathbb{E}A_i(1) = \mathbb{E}[\mathbb{E}[S_i^a(1) + S_i^j(1)|\,\mathcal{F}_0]] = \mathbb{E}[p_i^a(1)A_i(0) + p_i^j(1)J_i(0)] = p_i^a(1)A_i(0) + p_i^j(1)J_i(0)\\
&\mathbb{E}A_i(2) = \mathbb{E}[\mathbb{E}[S_i^a(2) + S_i^j(2)|\,\mathcal{F}_1]] = \mathbb{E}[p_i^a(2)A_i(1) + p_i^j(2)J_i(1)]\\
&\qquad = p_i^a(2)\mathbb{E}[A_i(1)] + p_i^j(2)\mathbb{E}[\mathbb{E}[S_i^l(1)|\,\mathcal{F}_0]] = p_i^a(2)\mathbb{E}[A_i(1)] + p_i^j(2)p_i^l(1)L_i(0).
\end{aligned}$$

In the same way the following holds for $t \ge 3$

$$\mathbb{E}[A_i(t)] = \mathbb{E}[\mathbb{E}[S_i^a(t) + S_i^j(t)|\,\mathcal{F}_{t-1}]] = \mathbb{E}[p_i^a(t)A_i(t-1) + p_i^j(t)J_i(t-1)].$$

Now since

$$\mathbb{E}[J_i(t)] = \mathbb{E}[\mathbb{E}[S_i^l(t)|\,\mathcal{F}_{t-1}]] = p_i^l(t)\mathbb{E}[L_i(t-1)] \tag{4.2}$$

$$\mathbb{E}[L_i(t)] = \mathbb{E}[\mathbb{E}[\sum_{j=1}^{N} X_{ji}(t)|\,\mathcal{F}_{t-1}]] = \sum_{j=1}^{N} \lambda_{ji}\mathbb{E}[A_j(t-1)] \tag{4.3}$$

this results in

$$\mathbb{E}[A_i(t)] = p_i^a(t)\mathbb{E}[A_i(t-1)] + p_i^j(t)p_i^l(t-1)\sum_{j=1}^{N}\lambda_{ji}\mathbb{E}[A_j(t-3)].$$

This gives us a recursive relation for $\mathbb{E}[A_i(t)]$. This recursive relation can be rewritten in terms of a matrix difference equation (MDE), which can solved. Noting that we had defined

$$\mu_t := (\mathbb{E}A_1(t), \mathbb{E}A_2(t), \ldots, \mathbb{E}A_N(t))^T$$

we get the MDE

$$\begin{pmatrix}\mu_t\\ \mu_{t-1}\\ \mu_{t-2}\end{pmatrix} = \begin{pmatrix}P^a(t) & 0 & \Lambda\\ I & 0 & 0\\ 0 & I & 0\end{pmatrix}\cdot\begin{pmatrix}\mu_{t-1}\\ \mu_{t-2}\\ \mu_{t-3}\end{pmatrix}$$

with I the $N\times N$-identity matrix, 0 the $N\times N$-zero matrix,

$$P^a(t) = \mathrm{diag}(p_1^a(t), p_2^a(t), \ldots, p_N^a(t))$$

and

$$\Lambda = \begin{pmatrix}
p_1^j(t)p_1^l(t-1)\lambda_{11} & p_1^j(t)p_1^l(t-1)\lambda_{21} & \ldots & p_1^j(t)p_1^l(t-1)\lambda_{N1}\\
p_2^j(t)p_2^l(t-1)\lambda_{12} & p_2^j(t)p_2^l(t-1)\lambda_{22} & & p_2^j(t)p_2^l(t-1)\lambda_{N2}\\
\vdots & \vdots & \ddots & \vdots\\
p_N^j(t)p_N^l(t-1)\lambda_{1N} & p_N^j(t)p_N^l(t-1)\lambda_{2N} & \ldots & p_N^j(t)p_N^l(t-1)\lambda_{NN}
\end{pmatrix}.$$

With the notation

$$A_t = \begin{pmatrix} \mu_t \\ \mu_{t-1} \\ \mu_{t-2} \end{pmatrix}, \qquad B_t = \begin{pmatrix} P^a(t) & 0 & \Lambda \\ I & 0 & 0 \\ 0 & I & 0 \end{pmatrix},$$

the MDE can now be written as

$$A_t = B_t \cdot A_{t-1}$$

which gives as solution

$$A_t = B_t B_{t-1} B_{t-2} \cdots B_4 A_3.$$

The mean vector μ_t is now given by the first N rows of the given solution A_t.

The covariance matrix at time step t, Σ_t, can also be found using recursion and an MDE. Since for $1 \leq i, j \leq N$ by definition

$$\mathrm{Cov}(A_i(t), A_j(t)) = \mathbb{E}[A_i(t)A_j(t)] - \mathbb{E}[A_i(t)]\mathbb{E}[A_j(t)] \tag{4.4}$$

and μ_t is known, our focus will be on computing $\mathbb{E}[A_i(t)A_j(t)]$. For $i \neq j$ and $t \geq 3$ use of the tower property results in:

$$\begin{aligned}
\mathbb{E}[A_i(t)A_j(t)] &= \mathbb{E}[\mathbb{E}[(S_i^a(t) + S_i^j(t))(S_j^a(t) + S_j^j(t))|\,\mathcal{F}_{t-1}]] \\
&= p_i^a(t)p_j^a(t)\mathbb{E}[A_i(t-1)A_j(t-1)] + p_i^a(t)p_j^j(t)\mathbb{E}[A_i(t-1)J_j(t-1)] \\
&\quad + p_i^j(t)p_j^j(t)\mathbb{E}[J_i(t-1)J_j(t-1)].
\end{aligned}$$

To get a recursive relation, first the last term is rewritten:

$$\begin{aligned}
\mathbb{E}[J_j(t-1)J_i(t-1)] &= \mathbb{E}[\mathbb{E}[S_i^l(t-1)S_j^l(t-1)|\,\mathcal{F}_{t-2}]] \\
&= p_i^l(t-1)p_j^l(t-1)\mathbb{E}[L_i(t-2)L_j(t-2)] \\
&= p_i^l(t-1)p_j^l(t-1)\mathbb{E}[\mathbb{E}[\big(\sum_{k=1}^{N} X_{ki}(t-2)\big)\big(\sum_{m=1}^{N} X_{mj}(t-2)\big)|\,\mathcal{F}_{t-3}]] \\
&= p_i^l(t-1)p_j^l(t-1)\mathbb{E}[\big(\sum_{k=1}^{N} \lambda_{ki}A_k(t-3)\big)\big(\sum_{m=1}^{N} \lambda_{mj}A_m(t-3)\big)] \\
&= p_i^l(t-1)p_j^l(t-1)\sum_{k=1}^{N}\sum_{m=1}^{N} \lambda_{ki}\lambda_{mj}\mathbb{E}[A_k(t-3)A_m(t-3)].
\end{aligned}$$

In the same way it holds that

$$\begin{aligned}
\mathbb{E}[A_i(t)J_j(t)] &= \mathbb{E}[\mathbb{E}[(S_i^a(t)+S_i^j(t))S_j^l(t)|\,\mathcal{F}_{t-1}]] \\
&= p_i^a(t)p_j^l(t)\mathbb{E}[A_i(t-1)L_j(t-1)] + p_i^j(t)p_j^l(t)\mathbb{E}[J_i(t-1)L_j(t-1)] \\
&= p_i^a(t)p_j^l(t)\sum_{k=1}^{N}\lambda_{kj}\big(p_i^a(t-1)\mathbb{E}[A_i(t-2)A_k(t-2)] \\
&\qquad + p_i^j(t-1)\mathbb{E}[A_k(t-2)J_i(t-2)]\big) \\
&\quad + p_i^j(t)p_j^l(t)p_i^l(t-1)\sum_{k=1}^{N}\lambda_{kj}\big(\sum_{l=1}^{N}\lambda_{li}p_k^a(t-2)\mathbb{E}[A_l(t-3)A_k(t-3)] \\
&\qquad + p_k^j(t-2)\mathbb{E}[A_l(t-3)J_k(t-3)]\big),'
\end{aligned} \tag{4.5}$$

since it is straightforward to deduce that

$$\mathbb{E}[A_i(t-1)L_j(t-1)] = \sum_{k=1}^{N}\lambda_{kj}\big(p_i^a(t-1)\mathbb{E}[A_i(t-2)A_k(t-2)] + p_i^j(t-1)\mathbb{E}[A_k(t-2)J_i(t-2)]\big)$$

$$\begin{aligned}
\mathbb{E}[J_i(t-1)L_j(t-1)] &= p_i^l(t-1)\sum_{k=1}^{N}\lambda_{kj}\big(\sum_{l=1}^{N}\lambda_{li}p_k^a(t-2)\mathbb{E}[A_l(t-3)A_k(t-3)] \\
&\qquad + p_k^j(t-2)\mathbb{E}[A_l(t-3)J_k(t-3)]\big).
\end{aligned}$$

These results do specifically hold in the case $i \neq j$. Usage of the tower property for the case $i = j$ gives the following ($t \geq 3$):

$$\begin{aligned}
\mathbb{E}[A_i(t)^2] &= \mathbb{E}[\mathbb{E}[S_i^a(t)^2 + 2S_i^a(t)S_i^j(t) + S_i^j(t)^2|\,\mathcal{F}_{t-1}]] \\
&= p_i^a(t)(1-p_i^a(t))\mathbb{E}[A_i(t-1)] + p_i^a(t)^2\mathbb{E}[A_i(t-1)^2] \\
&\quad + 2p_i^a(t)p_i^j(t)\mathbb{E}[A_i(t-1)J_i(t-1)] + p_i^j(t)(1-p_i^j(t))\mathbb{E}[J_i(t-1)] + p_i^j(t)^2\mathbb{E}[J_i(t-1)^2].
\end{aligned}$$

By (4.2) we know

$$\mathbb{E}[J_i(t-1)] = p_i^l(t-1)\sum_{j=1}^{N}\lambda_{ji}\mathbb{E}[A_j(t-3)].$$

Furthermore, it can easily be shown that (4.5) does extend to the case $i = j$. To get a recursive relation, the second moment of $J_i(t)$ in the derivation of $\mathbb{E}[A_i(t)^2]$ has to be rewritten. Again using the tower property and the relations between the random variables, one finds that

$$\begin{aligned}
\mathbb{E}[J_i(t)^2] &= p_i^l(t)\sum_{k=1}^{N}\lambda_{ki}\mathbb{E}[A_k(t-2)] + p_i^l(t)^2\sum_{k=1}^{N}\lambda_{ki}^2\mathbb{E}[A_k(t-2)^2] \\
&\quad + p_i^l(t)^2\sum_{k=1}^{N}\sum_{l=1,l\neq k}^{N}\lambda_{ki}\lambda_{li}\mathbb{E}[A_k(t-2)A_l(t-2)].
\end{aligned}$$

Summing all terms it can now be concluded that

$$\begin{aligned}\mathbb{E}[A_i(t)^2] &= p_i^a(t)(1-p_i^a(t))\mathbb{E}[A_i(t-1)] + p_i^a(t)^2\mathbb{E}[A_i(t-1)^2] + 2p_i^a(t)p_i^j(t)\mathbb{E}[A_i(t-1)J_i(t-1)] \\ &\quad + p_i^j(t)p_i^l(t-1)\sum_{k=1}^{N}\lambda_{ki}\mathbb{E}[A_k(t-3)] \\ &\quad + p_i^j(t)^2p_i^l(t-1)^2\sum_{k=1}^{N}\sum_{l=1}^{N}\lambda_{ki}\lambda_{li}\mathbb{E}[A_k(t-3)A_l(t-3)].\end{aligned}$$

Altogether the following system of recursive relations has been derived, with $i \neq j$ as condition for the second relation

$$\begin{aligned}\mathbb{E}[A_i(t)^2] &= p_i^a(t)(1-p_i^a(t))\mathbb{E}[A_i(t-1)] + p_i^a(t)^2\mathbb{E}[A_i(t-1)^2] + 2p_i^a(t)p_i^j(t)\mathbb{E}[A_i(t-1)J_i(t-1)] \\ &\quad + p_i^j(t)p_i^l(t-1)\sum_{k=1}^{N}\lambda_{ki}\mathbb{E}[A_k(t-3)] \\ &\quad + p_i^j(t)^2p_i^l(t-1)^2\sum_{k=1}^{N}\sum_{l=1}^{N}\lambda_{ki}\lambda_{li}\mathbb{E}[A_k(t-3)A_l(t-3)] \\ \mathbb{E}[A_i(t)A_j(t)] &= p_i^a(t)p_j^a(t)\mathbb{E}[A_i(t-1)A_j(t-1)] + p_i^a(t)p_j^j(t)\mathbb{E}[A_i(t-1)J_j(t-1)] \\ &\quad + p_j^a(t)p_i^j(t)\mathbb{E}[A_j(t-1)J_i(t-1)] \\ &\quad + p_i^j(t)p_j^j(t)p_i^l(t-1)p_j^l(t-1)\sum_{k=1}^{N}\sum_{m=1}^{N}\lambda_{ki}\lambda_{mj}\mathbb{E}[A_k(t-3)A_m(t-3)] \\ \mathbb{E}[A_i(t)J_j(t)] &= p_i^a(t)p_j^l(t)\sum_{k=1}^{N}\lambda_{kj}\big(p_i^a(t-1)\mathbb{E}[A_i(t-2)A_k(t-2)] + p_i^j(t-1)\mathbb{E}[A_k(t-2)J_i(t-2)]\big) \\ &\quad + p_i^j(t)p_j^l(t)p_i^l(t-1)\sum_{k=1}^{N}\lambda_{kj}\big(\sum_{l=1}^{N}\lambda_{li}p_k^a(t-2)\mathbb{E}[A_l(t-3)A_k(t-3)] \\ &\quad + p_k^j(t-2)\mathbb{E}[A_l(t-3)J_k(t-3)]\big).\end{aligned}$$

These recursive relations we can again be rewritten as MDE, which, using the notation

$$\begin{aligned}\mu_t^{AA} &:= \begin{pmatrix}\mathbb{E}A_1(t)^2 & \mathbb{E}A_1(t)A_2(t) & \dots & \mathbb{E}A_1(t)A_N(t) & \mathbb{E}A_2(t)A_1(t) & \dots & \dots & \mathbb{E}A_N(t)A_N(t)\end{pmatrix}^T \\ \mu_t^{AJ} &:= \begin{pmatrix}\mathbb{E}A_1(t)J_1(t) & \mathbb{E}A_1(t)J_2(t) & \dots & \mathbb{E}A_1(t)J_N(t) & \mathbb{E}A_2(t)J_1(t) & \dots & \dots & \mathbb{E}A_N(t)J_N(t)\end{pmatrix}^T\end{aligned}$$

is given by

$$\begin{pmatrix}\mu_t^{AA}\\ \mu_{t-1}^{AA}\\ \mu_{t-2}^{AA}\\ \mu_t^{AJ}\\ \mu_{t-1}^{AJ}\\ \mu_{t-2}^{AJ}\end{pmatrix} = \begin{pmatrix}P^{aa}(t) & 0 & \Lambda'(t) & M(t) & 0 & 0\\ I & 0 & 0 & 0 & 0 & 0\\ 0 & I & 0 & 0 & 0 & 0\\ 0 & \Lambda_1 & \Lambda_2 & 0 & \Lambda_3 & \Lambda_4\\ 0 & 0 & 0 & I & 0 & 0\\ 0 & 0 & 0 & 0 & I & 0\end{pmatrix} \cdot \begin{pmatrix}\mu_{t-1}^{AA}\\ \mu_{t-2}^{AA}\\ \mu_{t-3}^{AA}\\ \mu_{t-1}^{AJ}\\ \mu_{t-2}^{AJ}\\ \mu_{t-3}^{AJ}\end{pmatrix} + \begin{pmatrix}C(t)\\ 0\\ 0\\ 0\\ 0\\ 0\end{pmatrix}$$

where the explicit expressions for the block matrices can easily be derived from the recursive relations. Now, in the same way as for the mean vector $\mu(t)$, the given MDE can be solved to find the covariance matrix. Denoting

$$D_t = \begin{pmatrix} \mu_t^{AA} \\ \mu_{t-1}^{AA} \\ \mu_{t-2}^{AA} \\ \mu_t^{AJ} \\ \mu_{t-1}^{AJ} \\ \mu_{t-2}^{AJ} \end{pmatrix}, \qquad E_t = \begin{pmatrix} P^{aa}(t) & 0 & \Lambda'(t) & A_j(t) & 0 & 0 \\ I & 0 & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 & 0 \\ 0 & \Lambda_j^1 & \Lambda_j^2 & 0 & \Lambda_j^3 & \Lambda_j^4 \\ 0 & 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & 0 & I & 0 \end{pmatrix}, \qquad C_t = \begin{pmatrix} C(t) \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

the MDE reads

$$D_t = E_t \cdot D_{t-1} + C_t$$

which has the solution

$$D_t = E_t E_{t-1} E_{t-2} \cdots E_3 D_2 + E_t E_{t-1} E_{t-2} \cdots E_4 C_3 + \cdots + E_t E_{t-1} C_{t-2} + E_t C_{t-1} + C_t.$$

The covariance matrix Σ_t can now be found using (4.4), the solution D_t and the solution A_t for the computation of μ_t.

The method we are using for optimizing the presented objective function by using an approximate model is only relevant if evaluating and optimizing the approximate model is significantly faster than for the real model. This means it is important that the computation of the mean vector and covariance matrix can be done fast. It turns out that we can speed up the algorithm of solving the MDE for the covariance matrix as presented above by using symmetry. In the computation of the covariance matrix μ^{AA} is computed, which we defined as

$$\mu_t^{AA} := \begin{pmatrix} \mathbb{E}A_1(t)^2 & \dots & \mathbb{E}A_1(t)A_N(t) & \mathbb{E}A_2(t)A_1(t) & \dots & \dots & \mathbb{E}A_N(t)A_N(t) \end{pmatrix}^T.$$

It can be noted that for $i \neq j$ both $\mathbb{E}[A_i(t)A_j(t)]$ and $\mathbb{E}[A_j(t)A_i(t)]$ are evaluated. Since these terms are the same, defining

$$\mu_t' := \begin{pmatrix} \mathbb{E}A_1(t)^2 & \dots & \mathbb{E}A_1(t)A_N(t) & \mathbb{E}A_2(t)^2 & \mathbb{E}A_2(t)A_3(t) & \dots & \dots & \mathbb{E}A_N(t)A_N(t) \end{pmatrix}^T$$

and rewriting the MDE in terms of μ_t' will give the same information. However, the computations will imply the multiplication of block matrices which have $\frac{1}{2}N(N+1)$ rows and columns instead of N^2, and will thus be faster.

We have used recursion to compute the mean vector and covariance matrix. However, for large N, even when using symmetry, the computational costs can be too high, since the matrices will have large dimensions. Moreover, since we are only interested in the rows related to the high-value patches, the results of the other rows go unused. Thus, a computation method only concerned with the high-value patches would be preferred.

One method that does this will be presented below and this method will be the implemented method for the computation of the objective of the approximate model, since it also overcomes the problem of the use of large dimensional matrices. The method uses recursion on the MGF

$$g_t(\theta) := \mathbb{E}[e^{\sum_{i=1}^N \theta_i^A A_i(t)+\sum_{i=1}^N \theta_i^J J_i(t)+\sum_{i=1}^N \theta_i^L L_i(t)}],$$

with

$$\theta = (\theta_1^A, \ldots, \theta_N^A, \theta_1^J, \ldots, \theta_N^J, \theta_1^L, \ldots, \theta_N^L).$$

Using the known expressions of an MGF of a Poisson or Binomial random variable and using the notation $c(p, w) = 1 - p + pe^w$, it holds that:

$$\begin{aligned}
g_t(\theta) &= \mathbb{E}[\prod_{i=1}^N e^{\theta_i^A A_i(t)+\theta_i^J J_i(t)+\theta_i^L L_i(t)}] \\
&= \mathbb{E}[\mathbb{E}[\prod_{i=1}^N e^{\theta_i^A A_i(t)+\theta_i^J J_i(t)+\theta_i^L L_i(t)}|\mathcal{F}_{t-1}]] \\
&= \mathbb{E}[\mathbb{E}[\prod_{i=1}^N e^{\theta_i^A S_i^a(t)+\theta_i^A S_i^j(t)+\theta_i^J S_i^l(t)+\theta_i^L \sum_{j=1}^N X_{ji}(t)}|\mathcal{F}_{t-1}]] \\
&= \mathbb{E}[\prod_{i=1}^N c(p_i^a(t), \theta_i^A)^{A_i(t-1)} c(p_i^j(t), \theta_i^A)^{J_i(t-1)} c(p_i^l(t), \theta_i^J)^{L_i(t-1)} \prod_{j=1}^N \exp(\lambda_{ji} A_j(t-1)(e^{\theta_i^L} - 1))] \\
&= \mathbb{E}[e^{\sum_{i=1}^N [(\sum_{j=1}^N \lambda_{ij}(e^{\theta_j^L} - 1)) + \log(c(p_i^a(t), \theta_i^A)] A_i(t-1) + \sum_{i=1}^N \log(c(p_i^j(t), \theta_i^A) J_i(t-1) + \sum_{i=1}^N \log(c(p_i^l(t), \theta_i^J)) L_i(t-1)}]
\end{aligned}$$

which implies

$$g_t(\theta) = g_{t-1}(\hat{\theta})$$

with $\hat{\theta} = (\hat{\theta}_1^A, \ldots, \hat{\theta}_N^A, \hat{\theta}_1^J, \ldots, \hat{\theta}_N^J, \hat{\theta}_1^L, \ldots, \hat{\theta}_N^L)$ and

$$\begin{aligned}
\hat{\theta}_i^A &= \sum_{j=1}^N \lambda_{ij}(e^{\theta_j^L} - 1) + \log(1 - p_i^a(t) + p_i^a(t) e^{\theta_i^A}) \\
\hat{\theta}_i^J &= \log(1 - p_i^j(t) + p_i^j(t) e^{\theta_i^A}) \\
\hat{\theta}_i^L &= \log(1 - p_i^l(t) + p_i^l(t) e^{\theta_i^J}).
\end{aligned}$$

Now, since it is assumed that $A_i(0), J_i(0), L_i(0), i = 1, \ldots, N$ are known, this recursion can be used to give values of g_t for given values of θ.

With the MGF it is now possible to approximate moments by taking derivatives of the MGF. Denote with e_i the vector with 1 in the i-th row and 0 elsewhere and let h be

a small number. Then, using finite difference approximations [13], for $i \neq j$ the values

$$\tilde{\mu}_i^t := \frac{g_t(e_i h) - g_t(0)}{h} = \frac{g_t(e_i h) - 1}{h}$$
$$\tilde{\sigma}_i^t := \frac{g_t(e_i h) - 2g_t(0) + g_t(-e_i h)}{h^2} = \frac{g_t(e_i h) - 2 + g_t(-e_i h)}{h^2}$$
$$\tilde{\sigma}_{ij}^t := \frac{g_t(e_i h + e_j h) - g_t(e_i h - e_j h) - g_t(e_j h - e_i h) + g_t(-e_i h - e_j h)}{4h^2}$$

can respectively be used as approximations of $\mathbb{E}A_i(t)$, $\mathbb{E}A_i^2(t)$ and $\mathbb{E}A_i(t)A_j(t)$, which give the mean vector and covariance matrix of the adult COTS densities of the high-value patches and are used in the computation of the objective of the approximate model.

4.2. Connecting approximate and detailed model

In the model dynamics of the surrogate model a matrix $\Lambda = (\lambda_{ji})_{1 \leq i,j \leq N}$ is used. This matrix Λ will be used to tune $\tilde{f}$ such that it fits f around a specific effort allocation x. In the functional form based optimization algorithm an iterate x is given for which an appropriate Λ has to be determined. We will now describe how we use the sampling of $F(x)$ to determine an appropriate Λ.

For $1 \leq i, j \leq N$ the parameter λ_{ji} is an estimation of the rate at which COTS larvae disperse from patch j to patch i with respect to the density of adults at patch j. The following procedure is used to estimate λ_{ji}. Take $1 \leq i \leq N$ and $n = 1$ (the first iteration of the algorithm). One simulation of the detailed COTS model is run, in which for every time step and every $1 \leq j \leq N$ f_{ji} is computed. Now the larval density produced in the system and dispersing to patch i is computed with the values of f_{ji} by sampling the random variable $X_i(t)$. This sample gives only information about the total larval density contributing to the population growth on i, not about the contribution from the larval density dispersing from j to i. We estimate the contribution growth of the larvae arriving from j using the individual f_{ji}. We scale such that $\sum_{j=1}^N f_{ji} = 1$ and let contribution of the larval density arriving from j be according to the weight of the scaled f_{ji}. Then, to let λ_{ji} be the rate per adult density on the patch, this number is divided by the number of adult population units on patch j. Thus:

$$\lambda_{ji}^1(t) = X_i(t) \cdot \frac{f_{ji}}{\sum_{k=1}^N f_{ki}} \cdot \frac{1}{A_j(t-1)}.$$

If $\sum_{j=1}^N f_{ji} = 0$ there are no larvae dispersing to i and we put $\lambda_{ji}^1(t) = 0$. Now λ_{ji}^1 is given by the average of $\lambda_{ji}^1(t)$ over t. Then Λ is defined as $\Lambda := (\lambda_{ji}^1)$.

For $n = 2$ the same procedure is used, but the average of the two matrices found in $n = 1$ and $n = 2$ is used as Λ. For $n = 3$ the same holds: a matrix is found using the procedure described above and Λ is defined as the average of this matrix and the matrix found for $t = 2$. Iteration now results in Λ for $n \geq 4$.

We have presented a function $\tilde{f}$ that can be tuned by the choice of an appropriate matrix Λ to align with the shape of the objective function f around a given effort allocation

x. In the functional form based algorithm this function $\tilde{f}$ is optimized with respect to x to give a new iterate (see Figure 3.2). Since the computation of $\tilde{f}$ in our presented approximate model is deterministic, there are multiple ways to optimize this with respect to the effort allocation x. The optimization method chosen should work in general, since there is no indication that the function that computes this maximum expectation is e.g. linear or convex. In the next chapter we will present two optimization algorithms that can be used to compute the optimal effort allocation in the approximation model: the backtracking line search algorithm (BLS) and the NITRO algorithm. Moreover, to test the performance of our methodology a stochastic version of BLS is presented, which can be used to optimize the invasive species model under study. Comparison of the CPU times of the functional form based optimization and the stochastic BLS will indicate the performance of our functional form based methodology.

5. Optimization algorithms

In this chapter three optimization algorithms will be described. An optimization algorithm is needed in the loop of the functional form based algorithm to get an optimal effort allocation from the objective function $\tilde{f}$. Furthermore, to get an indication of how well the functional form based algorithm performs, computational results need to be analyzed in which the performance of the functional form based approach is compared with other optimization algorithms on the original model. In this chapter three optimization algorithms will be presented: backtracking line search (BLS), the NITRO algorithm and Kiefer-Wolfowitz stochastic approximation combined with backtracking line search (BLS-SA). The first two optimization algorithms are intended for deterministic optimization and will thus be used in the functional form based optimization loop to optimize the deterministic approximate model objective $\tilde{f}$. Kiefer-Wolfowitz stochastic approximation (KW-SA) is widely recognized as the method to optimize an expectation in case of noise: not the expectation f itself but only observations of random variables F can be observed for which $\mathbb{E}F = f$. The KW-SA algorithm will be adapted with the ideas of BLS and used to optimize the original invasive species model and compare the results with our presented methodology.

5.1. Backtracking line search

The idea of the backtracking line search algorithm we use is described in Wright [28]. In line search the goal is to minimize a function $f(\cdot) : \mathbb{R}^n \to \mathbb{R}$ for some $n \in \mathbb{N}$. Given a starting vector x_0 line search iterates according to the iteration

$$x_{k+1} = x_k + \alpha_k p_k.$$

Here the vector p_k gives the search direction and the scalar α_k the step length, thus α_k indicates how far to move along the search direction p_k. The search direction p_k is chosen such that $p_k^T \nabla f(x_k) < 0$ in which case p_k is a decent direction. The simplest way to guarantee $p_k^T \nabla f(x_k) < 0$ is to let $p_k = -\nabla f(x_k)$, which is called the steepest descent method.
In computation of an appropriate step length α_k, ideally one would choose

$$\alpha_k = \min_{\alpha > 0} \phi(\alpha) := \min_{\alpha > 0} f(x_k + \alpha p_k).$$

However, if the computation costs of computing this minimum are high, this would yield a slow iteration. Most line search algorithms, including backtracking line search, use other methods to compute an appropriate α, keeping more balance in the trade-off of reducing

f the most and doing this at minimal cost. Backtracking line search (Algorithm 2) uses the sufficient decrease Armijo condition to determine the appropriate α, with the Armijo condition given by

$$f(x_k + \alpha p_k) \leq f(x_k) + c\alpha \nabla f(x_k)^T p_k$$

for some $c \in (0, 1)$. α_k is now found by checking the Armijo condition for a scalar α^0 and repeating this for $\alpha^{n+1} = \rho\alpha^n$ for some $\rho \in (0, 1)$. The first value of α^n for which the Armijo condition holds is set to α_k, which we can then use to compute the next iterate x_{k+1} in the line search algorithm.

Result: approximation of optimizer x^* of f
Initialization: set $k = 1$, choose $\epsilon > 0$, x_0, $\rho \in (0, 1), c \in (0, 1)$;
while $k = 1$ *or* $||x_{k+1} - x_k|| > \epsilon$ **do**

(i) Determine the descent direction p_k

(ii) Determine a_k in the following way: choose $\hat{\alpha} > 0$ and set $\alpha = \hat{\alpha}$;
while $f(x_k + \alpha p_k) > f(x_k) + c\alpha \nabla f(x_k)^T p_k$ **do**

$$\alpha = \rho\alpha$$

end

(iii) Compute

$$x_{k+1} = x_k + \alpha p_k$$

(iv) $k = k + 1$

end

Algorithm 2: Backtracking line search.

The backtracking line search algorithm can not directly be used in optimizing the function f in an invasive species problem, since it does not handle constraints. One way to deal with this shortcoming is by using the same idea as constraint stochastic approximation, where, if the next iterate does not fulfill the constraints, the closest point (in terms of Euclidean norm) that does fulfill the constraints is chosen as next iterate. Thus, using this the iteration in backtracking line search is given by

$$x_{k+1} = \Pi_{\mathcal{X}}(x_k + \alpha_k p_k),$$

with $\Pi_{\mathcal{X}}(\cdot)$ the function that gives the closest point in Euclidean norm in the space $\mathcal{X}$. The backtracking line search algorithm is believed to optimize fast, but has as biggest concern that it does handle the constraints artificially: the constraints are not considered during the iteration steps but only after each iteration. An algorithm which does consider the constraints during the optimization steps is the NITRO algorithm, presented in the Section 5.3.

5.2. Stochastic approximation

To optimize the invasive species model under study, a version of constraint stochastic approximation can be used. One can view stochastic approximation as the stochastic version of line search algorithms: instead of the use of the exact derivative a stochastic estimator for the gradient is used to determine the direction of the iteration step. Our focus will be on Kiefer-Wolfowitz (KW) algorithms, characterized by their use of finite differences in the derivative estimation. It has been proven that under certain conditions stochastic approximation converges in probability to a local optimizer. One of the conditions in the proof is however that the function to optimize is convex, which is not guaranteed in our situation. In Asmussen and Glynn [3] it is proposed to solve this by using multiple starting points as inputs and decide on the values of the output which solution is a local optimizer and which solution is the global optimizer. Other stochastic algorithms capable of handling non-convex optimization were considered, but no adequate algorithm was found. Take for example the class of stochastic majorization-minimization algorithms. To perform these algorithms on a function f one needs a function g for which $g \geq f$ and $g(x) = f(x)$ for a certain x [26]. Since there is no closed-form expression available for our function f, such a function g can not be found explicitly, and the popular stochastic majorization-minimization algorithms can not be used. Thus the method for handling non-convexity as proposed by Asmussen and Glynn is used in our optimization.

Throughout this section we will assume x to be continuous and we will denote with $\mathcal{X} \subseteq \mathbb{R}^N$ the feasible region. The iterative Kiefer-Wolfowitz algorithms are now given by

$$x_{n+1} = \Pi_{\mathcal{X}}(x_n - a_n Y_{n+1}),$$

where $\{a_n\}_{n\geq 0}$ is a sequence of positive deterministic constants, Y_{n+1} a random variable denoting the gradient estimator as given by the method of finite differences and $\Pi_{\mathcal{X}}(x) : \mathbb{R}^N \to \mathcal{X}$ is a function giving the point in $\mathcal{X}$ closest to x. From this expression it can explicitly be noted that stochastic approximation is a stochastic variant of line search methods.

In the original paper of Kiefer and Wolfowitz [23] convergence in probability is proven in the case of a one-dimensional input process and one-dimensional output process, where certain continuity conditions need to hold for the function to optimize and some extra conditions need to hold for the sequence $\{a_n\}$ and the finite differences estimator. This proof was extended to the multidimensional case by Blum [7]. An outline of the proof as presented by Blum can be found in Appendix C. The conditions needed for $\{a_n\}$ and the finite differences estimator guarantee convergence in probability to the optimizer. However, these conditions also imply $a_n \to 0$ for $n \to \infty$, such that the convergence to the optimizer might happen arbitrary slow. For this reason we will adapt the KW-SA algorithm, defining a_n as in backtracking line search.

5.3. NITRO algorithm

NITRO stands for non-linear interior point trust region optimizer and the NITRO algorithm is designed to solve non-linearly constrained optimization problems of the form

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to} \quad & h(x) = 0 \\ & g(x) \leq 0 \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}, h : \mathbb{R}^n \to \mathbb{R}^t$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are smooth functions. The NITRO algorithm is designed by Byrd, Hribar and Nocedal [8] and uses the theory of trust region methods, sequential quadratic programming (SQP) and a barrier method. The NITRO algorithm is implemented in the standard Python library SciPy.

The barrier method is used to construct subproblems of the form

$$\begin{aligned} \min \quad & f(x) - \mu \sum_{i=1}^{m} \ln s_i \\ \text{subject to} \quad & h(x) = 0 \\ & g(x) + s = 0, \end{aligned}$$

with $\mu > 0$ a barrier parameter. The idea of the barrier method is that when μ is chosen to converge to 0, the sequence of (approximate) minimizers to the given subproblems converge to the actual minimizer.
Solving the subproblem is done by means of the Lagrangian, which is given by

$$\mathcal{L}(x, s, \lambda_h, \lambda_g) = f(x) - \mu \sum_{i=1}^{m} \ln s_i + \lambda_h^T h(x) + \lambda_g^T (g(x) + s),$$

with λ_h and λ_g the least square multiplier estimates for h and g respectively. The derivation of these least square estimates can be found in Appendix B. The barrier subproblem will be considered solved if an approximate minimizer $(\hat{x}, \hat{s})$ is found for which $E(\hat{x}, \hat{s}; \mu) \leq \epsilon_\mu$ for some predefined $\epsilon_\mu > 0$ and E defined by

$$E(x, s; \mu) = \max(||\nabla f(x) + A_h(x)\lambda_h + A_g(x)\lambda_g||_\infty, ||S\lambda_g - \mu e||_\infty, ||h(x)||_\infty, ||g(x) + s||_\infty).$$

Here $S = \text{diag}(s^1, \ldots, s^m), e = [1, \ldots, 1]^T$ and

$$A_h(x) = [\nabla h^1(x), \ldots, \nabla h^t(x)] \qquad A_g(x) = [\nabla g^1(x), \ldots, \nabla g^m(x)].$$

Now, we want to find an approximate solution to the original problem, thus the goal is to find x, s such that $E(x, s; 0) \leq \epsilon_0$. The barrier algorithm does this by decreasing μ and ϵ_μ, while finding an approximate solution $(\hat{x}, \hat{s})$ such that $E(\hat{x}, \hat{s}; \mu) \leq \epsilon_\mu$ for every pair (μ, ϵ_μ), until there is an approximate solution $(\hat{x}, \hat{s})$ such that $E(x, s; 0) \leq \epsilon_0$. Finding the approximate solution $(\hat{x}, \hat{s})$ is done by means of an SQP method with trust regions. Byrd, Hribar and Nocedal [8] motivate the use of an SQP approach in the optimization,

recognizing that SQP is known to be effective for solving problems with equality constraints.
SQP with trust regions first uses a quadratic model of the Lagrangian and a linear approximation of the constraints. Then a step $d := (d_x, d_x)^T$ is computed by optimization of this model subject to these linear constraints and a trust region bound on the step, where the trust region gives the region around the current iterate x_k for which we trust that the quadratic model is a good approximation of the Lagrangian. Then it has to be decided if the step d is accepted, where it will be accepted if it gives enough reduction in the optimization step.
The quadratic subproblem found by a Taylor approximation of the Lagrangian, noting that constants can be omitted in the optimization, is given by

$$\begin{aligned} \min_{d_x, d_s} \quad & \nabla f(x_k)^T d_x + \frac{1}{2} d_x^T \nabla^2_{xx} \mathcal{L}(x_k, s_k, \lambda_h, \lambda_g) d_x - \mu e^T S_k^{-1} d_s + \frac{1}{2} d_s^T \Sigma_k d_s \\ \text{subject to} \quad & A_h(x_k)^T d_x + h(x_k) = r_h \\ & A_g(x_k)^T d_x + d_s + g(x_k) + s_k = r_g \\ & (d_x, d_s)^T \in T_k. \end{aligned}$$

Σ_k represents (an approximation of) the Hessian of the Lagrangian with respect to s and will in the NITRO algorithm be chosen to be

$$\Sigma_k = S_k^{-1} \Lambda_g \qquad \Lambda_g = \text{diag}(\lambda_g^1, \ldots, \lambda_g^m).$$

In Appendix B a brief derivation is given for the used expresion of T_k, a closed and bounded set defining the trust region, given by

$$||(d_x, S_k^{-1} d_s)||_2 \leq \Delta_k, \qquad \text{and} \qquad d_s \geq -\Lambda s_k$$

for $\tau \in (0, 1)$ and Δ_k the trust region radius, which is updated every time step. $r = (r_h, r_g)$, called the residual vector, emerges from the linear approximation of the constraints and is chosen as the minimal vector (in terms of Euclidean norm) that makes the constraints in the quadratic model consistent.
As motivated, a step $d = (d_x, d_s)^T$ can be computed by optimization of this quadratic model subject to the constraints, after which it is decided if d is accepted. The step d will be accepted if it makes enough progress towards the minimum. In constrained optimization a scalar-valued function that indicates if the new iterate makes enough progress towards the minimum is called a merit function. The merit function for the acceptance of a step d used in the NITRO algorithm is, with $\nu > 0$ a penalty parameter:

$$\phi(x, s; \nu) = f(x) - \mu \sum_{i=1}^{m} \ln s_i + \nu || \begin{pmatrix} h(x) \\ g(x) + s \end{pmatrix} ||_2.$$

Now, the next step is the optimization of the quadratic model to compute $d = (d_x, d_s)$. The vector r will be determined during this optimization process. In the computation of d a composition is used. The idea is to write $d = v + w$, where v is called a vertical

step and w a horizontal step. The vertical step is performed to satisfy the constraints and the horizontal step is performed to achieve the optimal solution.
The vertical step is performed to both satisfy the constraints as good as possible in terms of least squares and make sure we end up in the trust region. Thus, $v = (v_x, v_s)^T$ is defined as the solution of the optimization problem

$$\begin{aligned} \min_{v} \quad & f_{con} := ||A_h(x_k)^T v_x + h||_2^2 + ||A_g(x_k)^T v_x + v_s + g(x_k) + s_k||_2^2 \\ \text{subject to} \quad & ||(v_x, S_k^{-1} v_s||_2 \leq \xi \Delta_k \\ & v_s \geq -\tau s_k \end{aligned}$$

with $\xi \in (0,1)$, to make sure the next iterate is in the trust region. Now, with c_1, c_2 being vectors independent of v:

$$\begin{aligned} ||A_h(x_k)^T v_x + h||_2^2 &= v_x^T A_h(x_k) A_h(x_k)^T v_x + 2h^T A_h(x_k)^T v_x + c_1 \\ ||A_g(x_k)^T v_x + v_s + g(x_k) + s_k||_2^2 &= ||A_g(x_k)^T v_x + v_s||_2^2 + 2(g(x_k) + s_k)^T (A_g(x_k)^T v_x + v_s) + c_2 \\ &= v_x^T A_g(x_k) A_g(x_k)^T v_x + 2 v_s^T A_g(x_k)^T v_x + v_s^T v_s \\ &\quad + 2(g(x_k) + s_k)^T (A_g(x_k)^T v_x + v_s) + c_2. \end{aligned}$$

Addition of these terms, omitting the constants c_1 and c_2 and defining

$$\hat{A}_k = \begin{pmatrix} A_h(x_k) & A_g(x_k) \\ 0 & S_k \end{pmatrix}$$

results in the equivalent optimization problem

$$\begin{aligned} \min_{v} \quad & \tilde{f}_{con} := 2 \begin{pmatrix} h^T & (g(x_k) + s_k)^T \end{pmatrix} \hat{A}^T \begin{pmatrix} v_x \\ S_k^{-1} v_s \end{pmatrix} + \begin{pmatrix} v_x^T & v_s^T S_k^{-1} \end{pmatrix} \hat{A} \hat{A}^T \begin{pmatrix} v_x \\ S_k^{-1} v_s \end{pmatrix} \\ \text{subject to} \quad & ||(v_x, S_k^{-1} v_s)||_2 \leq \xi \Delta_k \\ & v_s \geq -\tau s_k. \end{aligned}$$

Minimization of this quadratic to compute v is performed with a so-called dogleg method. In Appendix B the used dogleg method is described and the resulting expression for v is given.

With this vertical step the residual vector r is defined as

$$\begin{aligned} r_h &= A_h^T v_x + h \\ r_g &= A_g^T v_x + v_s + g + s. \end{aligned}$$

This implies the quadratic subproblem to equal

$$\begin{aligned} \min_{d} \quad & \nabla f(x_k)^T d_x + \frac{1}{2}(d_x^T \nabla_{xx}^2 \mathcal{L}(x_k, s_k, \lambda_h, \lambda_g) d_x + d_s^T \Sigma_k d_s) - \mu e^T S_k^{-1} d_s \\ \text{subject to} \quad & A_h(x_k)^T d_x = A_h(x_k)^T v_x \\ & A_g(x_k)^T d_x + d_s = A_g^T v_x + v_s \\ & ||(d_x, S_k^{-1} d_s)||_2 \leq \Delta_k \\ & d_x \geq -\tau s_k \end{aligned}$$

and for this problem feasible solutions exist, since $d = v$ satisfies the constraints. To find an approximate solution for the subproblem we let $d = v + w$ where v is the vertical step for which we know how to find an expression and w is the so-called horizontal step. With the notation

$$\tilde{d} = \begin{pmatrix} d_x \\ S_k^{-1} d_s \end{pmatrix} = \begin{pmatrix} v_x \\ \tilde{v}_s \end{pmatrix} + \begin{pmatrix} w_x \\ \tilde{w}_s \end{pmatrix} = \tilde{v} + \tilde{w}$$

and

$$G = \begin{pmatrix} \nabla^2_{xx}\mathcal{L} & 0 \\ 0 & S\Sigma S \end{pmatrix}$$

an easy computation shows that the objective of the subproblem can be rewritten as

$$q(\tilde{v} + \tilde{w}) = \begin{pmatrix} \nabla f(x_k)^T & -\mu e^T \end{pmatrix} (\tilde{v} + \tilde{w}) + \frac{1}{2}(\tilde{v} + \tilde{w})^T G(\tilde{v} + \tilde{w}).$$

Now, since we want to make sure the solution is feasible, we will require $\hat{A}^T \tilde{w} = 0$ which will imply that $A_h^T w_x = 0$ and $A_g^T w_x + S\tilde{w}_s = 0$, such that the first two constraints of the quadratic subproblem are satisfied. Furthermore, since $\tilde{v}$ lies in the range space of $\hat{A}_k$ (Appendix B), we have that $\tilde{w}^T \tilde{v}^T = 0$ and

$$||(d_x, S_k^{-1} d_s)||_2 \leq \Delta_k \implies ||\tilde{w}||_2^2 \leq \Delta^2 - ||\tilde{v}||_2^2$$

which altogether now yields the so-called horizontal subproblem

$$\begin{aligned} \min_{\tilde{w}} \quad & q(\tilde{v} + \tilde{w}) \\ \text{subject to} \quad & A_h^T(x_k) w_x = 0 \\ & A_g^T(x_k) w_x + S\tilde{w}_s = 0 \\ & ||\tilde{w}||_2^2 \leq \Delta^2 - ||\tilde{v}||_2^2 \\ & \tilde{w}_s \geq \tau e - \tilde{v}_s. \end{aligned}$$

The horizontal step $\tilde{w}$ will now be found by the use of the conjugate gradient method in combination with the stopping tests of Steihaug to take into account the trust region. Details of this procedure can be found in Appendix B.

Now that we are able to compute the approximate solution $d = (d_x, d_s)$ to the quadratic subproblem, we have to decide if a proposed step is accepted. As described above, this will be done by means of the merit function

$$\phi(x, s; \nu) = f(x) - \mu \sum_{i=1}^{m} \ln s_i + \nu || \begin{pmatrix} h(x) \\ g(x) + s \end{pmatrix} ||_2.$$

With ν_k the penalty parameter at the k-th iteration we define the predicted reduction, which is how much ϕ approximately changes with the step d, as

$$\text{pred}_k(d) = -q(\tilde{v} + \tilde{w}) + \nu_k \sqrt{-f_{cont}(\tilde{v})}.$$

The actual predicted reduction, which is how much ϕ actually changes with the step d, can be defined as

$$\text{ared}_k(d) = \phi(x_k, s_k; \nu_k) - \phi(x_k + d_x, s_k + d_s; \nu_k).$$

A step d is now accepted if $\text{ared}_k(d) \geq \eta\, \text{pred}_k(d)$ for a $\eta \in (0, 1)$. Details on how to determine the penalty parameter ν_k can by found in Byrd, Hribrar, Nocedal [8]. When a step is accepted the trust region is updated accordingly. In the NITRO algorithm the update is defined as

$$\Delta_{k+1} = \max\{\gamma||d_k||, \Delta_k\}$$

where γ is dependent on the ratio between $\text{ared}_k(d)$ and $\text{pred}_k(d)$. When a step is rejected the trust region also changes and is at most half and at least one tenth of the step length.

6. Performance of the functional form based optimization

The methodology as presented by Dieker et al. [12] and Patch et al. [30] is only relevant if the methodology outputs an effort allocation close to the optimal effort allocation and if it can outperform other relevant optimization methods. To analyze if this is indeed the case computational experiments were performed. In analyzing the performance of the functional form based methodology, results of performing the methodology on different example networks will be presented. In the first section small example networks will be considered, such that we have an intuitive understanding of what the optimal effort allocation should be, and whether the algorithm outputs this decision vector. In the second section a larger network will be considered and the results of using the algorithm on this network will be discussed. The third section will be used to discuss another use of the functional form based methodology: sensitivity analysis.

All simulations were carried out with Python (Python-code presented in Appendix D). When the initial effort allocation is not explicitly mentioned it is assumed to be chosen at random. Estimates of the parameters in the COTS model are needed in the implementation of the model. For the parameters the following choices were made, where the variables *salinity* and *temperature* represent the influence of the background process $(B(t),\ t \geq 0)$ as presented in Chapter 2:

- Survival probabilities: estimated by performing logistic regression with as factors *salinity*, *temperature* and *coral cover* (in case of COTS) or *adult COTS density* (in case of coral).
- Reproduction rates: estimated by performing linear regression with as factor *temperature*.
- Dispersal probabilities: chosen to be constant during the experiments for implementation purposes.

Since data-sets are necessary to perform (logistic) regression, two 'toy' .csv-files were created (Appendix D). The first file contains binary data to perform logistic regression for the survival probabilities with explanatory variables *temperature*, *salinity* and *coral cover* or *adult COTS density*. The second file contains data to perform linear regression for the reproduction rates with explanatory *temperature*. In all carried out simulations a time frame of 15 years was considered.

6.1. Specific small network examples

First the experiments on the small example networks will be presented together with the results. After these experiments are presented some concluding remarks will be given.

2-patch network
The first network under consideration will be the 2-patch network given in Figure 6.1 with budget constraint $C = 1$. This network contains two patches, with:

- Colonized patch: 1 (1 adult density)
- High-value patch: 2
- Dispersal: $p_{12} = 0.8$, $p_{21} = 0$.

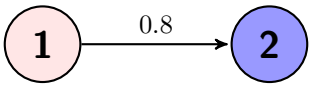

Figure 6.1.: 2-patch network, $I = \{2\}$, source: $\{1\}$.

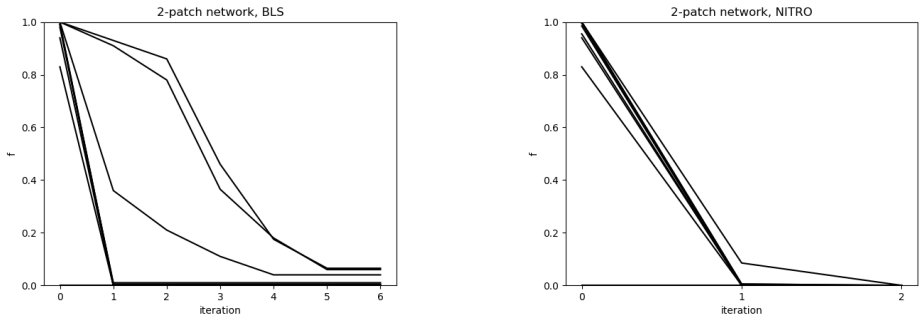

Figure 6.2.: Functional form based optimization algorithm paths in 2-patch network.

CPU time	Functional form based optimization		BLS-SA
	BLS	NITRO	
CMC:200	4.48	3.26	67.38
CMC:500	68.41	66.70	1059.26

Table 6.1.: CPU times computational experiments 2-patch network.

To analyze the performance of the functional form based optimization algorithm in the 2-patch network we run the algorithm with different initial effort allocations. Sample paths of the algorithm are shown in Figure 6.2 for both working with the BLS and NITRO

algorithm in the optimization step of the objective $\tilde{f}$. The sample paths are formed by estimated values of the objective function with as input the iterates x. In Table 6.1 the average CPU times for both methods are shown, as well as for performing the stochastic approximation algorithm presented in Chapter 5. We can make the following observations:

- Sample paths starting in the lower bound 0 remain constant, since in that case a stopping condition holds.
- Sample paths that do not start in 0 decrease and are after a few iterations around the value 0, which indicates the algorithm significantly reduces the objective function and in some cases even reaches the global optimum
- Using BLS the optimization needs more iterations before a stopping condition is met than using the NITRO algorithm.
- The CPU times in using the BLS and NITRO algorithm are of the same order.
- The CPU times for functional form based optimization are significantly lower than the CPU times for BLS-SA.

3-patch network
Consider the 3-patch network as presented in Figure 6.3 with $C = 0.5$ and:

- Colonized patch: 2 (1 COTS settler density)
- High-value patches: 1,3
- Dispersal: $p_{12} = p_{32} = 0$, $p_{21} = p_{23} = 0.5$

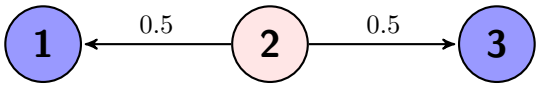

Figure 6.3.: 3-patch network, $I = \{1, 3\}$, source: $\{2\}$.

In this network there are two strategies to decrease the adult COTS density: the first strategy targets the high-value patches to make sure COTS settlers do not grow into adults on those patches, the second strategy targets the source patch to make sure there is no dispersal to the high-value patches. As can be expected the initial effort allocation vector has influence on which strategy the algorithm converges to and thus which optimum is reached. For example, running the BLS algorithms with initial allocation vector $[0.1, 0.1, 0.1]$ yields $[0.24, 0.03, 0.22]$, which is close to the local optimum $[0.25, 0.0, 0.25]$, while a run of the same algorithm with initial allocation vector $[0.0, 0.3, 0.0]$ yields $[0.01, 0.47, 0.00]$, which is close to the global optimum $[0.0, 0.5, 0.0]$. As mentioned in

the last chapter there is no indication that the objective is convex, which implies that the algorithm can indeed both converge to a local optimum instead of a global optimum.

Sample paths of random inputs with use of both the BLS and NITRO algorithms are shown in Figure 6.4. Average CPU times are shown in Table 6.2. We can make the following observations:

- When the sample path starts in the range [0.8,1.0] both methods significantly decrease this value.
- When the sample path starts in the range [0.0,0.25] it ends up in the same range, with most sample paths either converging the global optimum 0 or the local optimum around 0.23.
- Using BLS the optimization needs more iterations before a stopping condition is met than using the NITRO algorithm.
- The CPU times in using the BLS method are lower than in using the NITRO algorithm.
- The CPU times for functional form based optimization are significantly lower than the CPU times for BLS-SA.

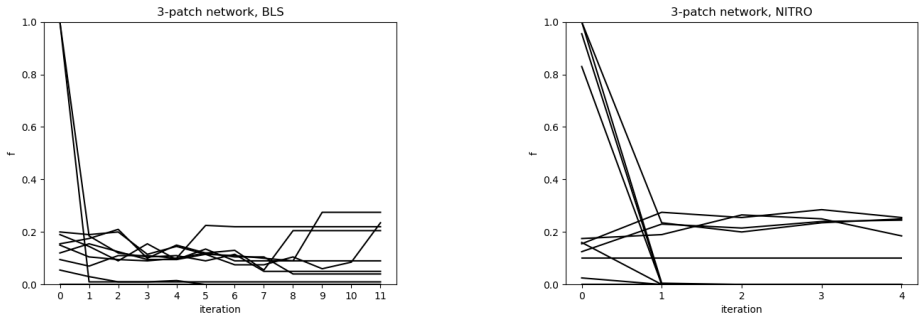

Figure 6.4.: Functional form based optimization algorithm paths in 3-patch network.

CPU time	Functional form based optimization		BLS-SA
	BLS	NITRO	
CMC:200	14.97	88.06	404.75
CMC:500	116.40	144.63	2503.45

Table 6.2.: CPU times computational experiments in the 3-patch network.

7-patch network

To confirm that the result in the 2- and 3-patch network can also be retrieved in a

slightly bigger network, let us consider the 7-patch network as shown in Figure 6.5 with $C = 1$ and

- Colonized patches: 3,4,5 (3 juvenile COTS densities per colonized patch)
- High-value patches: 1,6,7
- Dispersal: equally divided among connected patches.

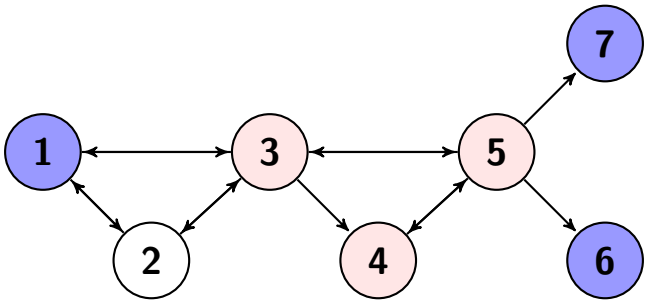

Figure 6.5.: 7-patch network, $I = \{1, 6, 7\}$, source: $\{3, 4, 5\}$.

As can easily be deduced from the figure, the control effort should be allocated to both patch 3 and patch 5 or both patch 1 and 5 to prevent the high-value patches from being colonized. The algorithm was run for multiple initial vectors and using both optimization strategies (BLS and NITRO) in optimizing $\tilde{f}$. In Figure 6.6 the sample paths of the optimizations are given and in Table 6.3 the average CPU times. We can make the following observations:

- The sample paths start in the range [0.9,1.0]. BLS decreases the values to the range [0.5,0.8]. Using the NITRO algorithm there are sample paths reaching the global optimum 0. Other sample paths either reach a value around 0.8 or around 0.4.
- Using BLS the optimization needs more iterations before a stopping condition is met than using the NITRO algorithm.
- The CPU times in using the BLS method are lower than in using the NITRO algorithm.
- The CPU times for functional form based optimization are significantly lower than the CPU times for BLS-SA.

4-patch network
Lastly, let us consider a 4-patch network in which the total allocated rate is too low to let the optimum be 0, a situation shown in Figure 6.7 with $C = 1$ and

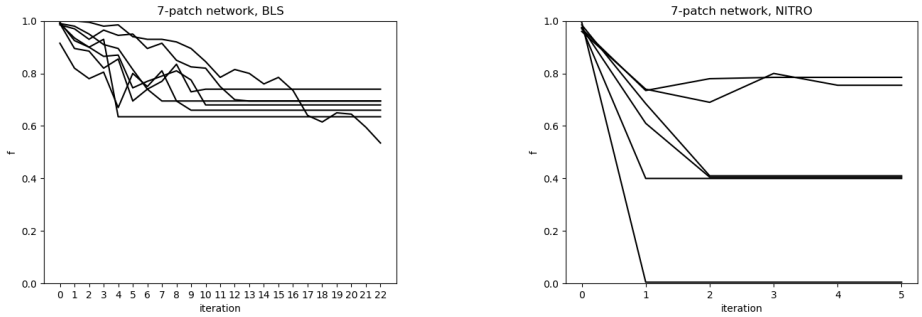

Figure 6.6.: Functional form based optimization algorithm paths 7-patch network.

CPU time	Functional form based optimization		BLS-SA
	BLS	NITRO	
CMC:200	164.95	634.8	2297.85

Table 6.3.: CPU times computational experiments in 7-patch network.

- Colonized patches: 1,2,3,4 (3 juvenile COTS densities per patch)
- High-value patches: 1,2,3
- Dispersal: $p_{ij} = 0.25$ for all $1 \leq i, j \leq 4$

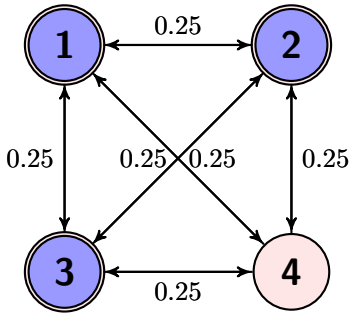

Figure 6.7.: 4-patch network, $I = \{1, 2, 3\}$, source: $\{1, 2, 3, 4\}$.

Since there is no evident optimal effort allocation resulting in an objective function with value 0, this is also an interesting case. Sample paths of the optimization are presented in Figure 6.8 and average CPU times in Table 6.4. Note that the true global optimal value lies around 0.57. We can make the following observations:

- The sample paths start in the range [0.6,0.85] and increase or decrease only slightly in every iteration. The global optimum is not reached, but the reached values are all 0.1 within the range of the global optimum.

- Using BLS the optimization needs more iterations before a stopping condition is met than using the NITRO algorithm.
- The CPU times of the BLS method and the NITRO algorithm are of the same order.
- The CPU times for functional form based optimization are significantly lower than the CPU times for BLS-SA.

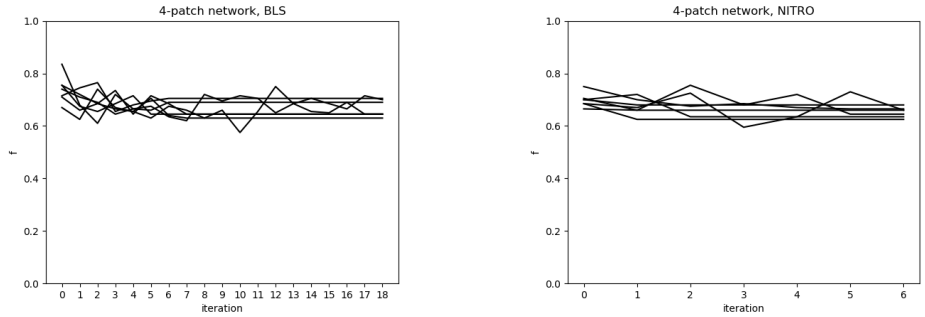

Figure 6.8.: Functional form based optimization algorithm paths in 4-patch network.

CPU time	Functional form based optimization		BLS-SA
	BLS	NITRO	
CMC:200	281.49	311.12	2360.15

Table 6.4.: CPU times computational experiments 4-patch network.

Remarks

The results of optimization with the functional form based approach in small networks look promising. Most sample paths show a significant decrease in the gap with the global optimum. The sample paths in the 3-patch example showing an increased estimate of the objective function have converged to the local optimum around 0.23. Optimization using BLS seems to need more iterations but still has an average CPU time lower than or of the same order as the average CPU time of optimization using the NITRO algorithm. The functional form based approach outperforms BLS-SA in terms of CPU time in these small networks, since in all presented networks the average CPU time of BLS-SA was significantly higher than that of functional form based optimization.

There are a few conclusions about the class of patches to target we can make based on the small examples just performed. There has been an ongoing discussion in invasive species management about the question to either target the source or the high-value patches [5]. Evident in our examples was that one can target the high-value patches in the case the allocated rate is high enough to significantly decrease the maximum

survival probability on the high-value patches. This was for instance true in the 2-patch example. If this is not the case the same holds for the source patches to make sure no dispersal takes place to the high-value patches. This happened in the 3-patch example with $C = 0.5$. However, there are situations in which it is optimal to target neither the high-value patches nor the source patches. An example of this was the 7-patch network.

6.2. Performance on a larger network

Having considered small networks in which it was easy to deduce which patches to target, we now move to a larger network. If the 20-patch network of figure 6.10 is considered, it can be seen that in that network it is not immediately clear which patches to target. The functional form based optimization algorithm was run multiple times. In Figure 6.9 sample paths can be seen and in Table 6.5 the average CPU times are shown. We observe that when using BLS in the optimization step, the first iterations all yield 1 as estimated objective function value. However, after a few iteration the paths all significantly decrease. When using the NITRO algorithm in the optimization step, fewer iterations are needed before the sample path decreases. The average CPU times of both algorithms are of the same order. The average CPU time of BLS-SA is significantly higher and thus does functional form based optimization also outperform BLS-SA in terms of CPU time in this larger network.

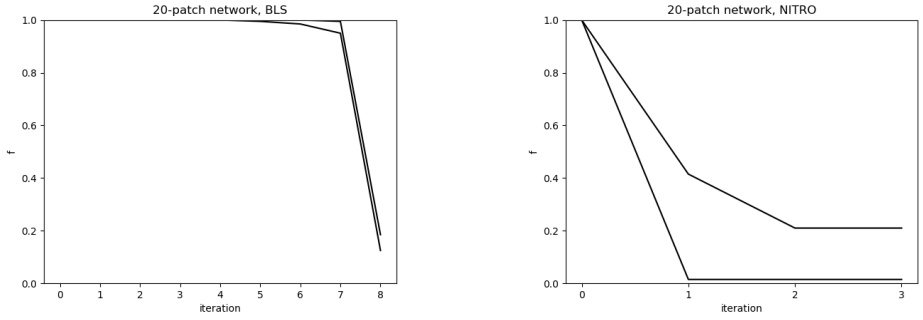

Figure 6.9.: Functional form based optimization algorithm paths 20-patch network.

CPU time	Functional form based optimization		BLS-SA
	BLS	NITRO	
20-patch	543.83	558.90	$> 10,000$

Table 6.5.: CPU times computational experiments 20-patch network.

The results from the 20-patch example look promising: the estimated objective function value is decreased significantly by the algorithm and the average CPU time needed to perform the optimization is significantly lower than the average CPU time needed to

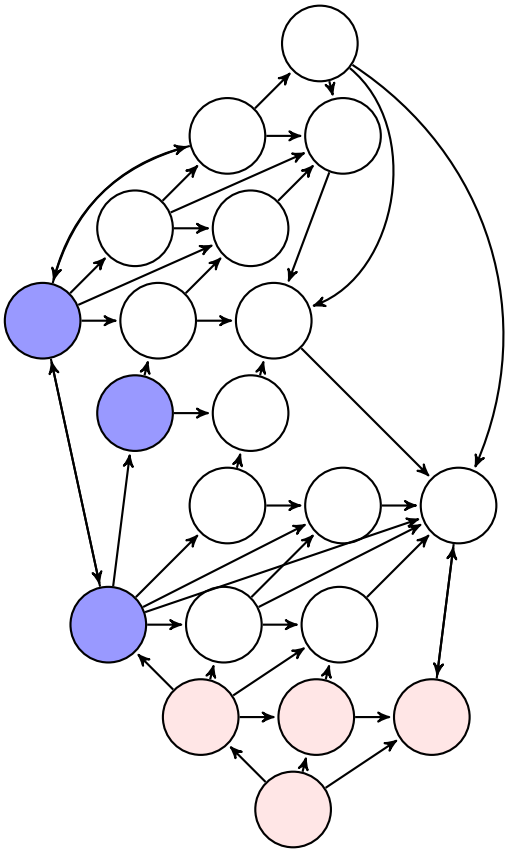

Figure 6.10.: 20-patch network, source patches red, high-value patches blue.

perform optimization using BLS-SA. The functional form based optimization algorithm should however be tested on more and larger networks before a real conclusion can be drawn.

6.3. Sensitivity analysis

The goal of the methodology we presented in Chapter 3 as used by Dieker et al. [12] and Patch et al. [30] is to speed up optimization in decision problems. One can however wonder if speeding up the optimization is necessary, when there is more time between two decisions than necessary for optimization. For example, if optimizing the real world model takes 3 months, but decisions only take place every year, the methodology seems otiose. There is however one important reason for the need to speed up the optimization, which is sensitivity analysis. Sensitivity analysis is performed to give insights on the effect of the parameters on the model. This gives outlines on where the model needs improvement; which parameters need to be estimated more accurately and which parameters might be left out of the model. Moreover, sensitivity analysis gives an idea on

the model's credibility which is important since only if the model is credible the model prediction can be trusted. In case the computational costs of running the simulation model are too high, sensitivity analysis can be performed by the methodology presented by Dieker et al. [12] and Patch et al. [30].

To study the effect of parameters on the model the idea is to run the model multiple times, while changing the parameter under study, and to observe the change in output [29]. More concretely, if one wants to study the effect of a certain parameter μ, one can for example run the model two times: once with parameter μ and once with parameter $\mu + \delta\mu$. Then one compares the outcomes y and $y + \delta y$. To solve the problem of y and μ possibly having different units, the so-called proportions are compared which results in the normalized sensitivity $\frac{\delta y}{y} / \frac{\delta \mu}{\mu} = \frac{\mu \delta y}{y \delta \mu}$. One problem is now how to choose $\delta\mu$. The theoretically best option is of course to take $\delta\mu$ infinitely small, but in simulations this does not always yield the best result, since not all programming software is capable of processing small numbers.

Since our model can be viewed as a black-box, sampling-based SA must be performed. If there is only one parameter in the model and this parameter can take 3 values, the methodology as described above can be used 3 times. However, most models, including the detailed COTS model presented in Chapter 2, have more parameters and exhaustive SA over all the values these parameters is not computationally feasible. One idea would be to sample the parameters at random, so to consider the whole credible range of the parameters and sample uniformly over this range. This does however not guarantee even spread in the samples. There are techniques to avoid this, two examples being stratified sampling and Latin hypercube sampling.

Stratification is a variance reduction method [3, 29] and the idea of stratified sampling is to divide the sample space into multiple regions, called strata, and take an equal number of samples in each region. The samples are uniformly distributed within a strata. The strata are defined by dividing the range using the most important random variables in the simulation by looking at their densities. One can also divide the strata by deciding on which parameter credible regions are important and assign more strata two these regions, or combine the two methods.

In Latin hypercube sampling the credible range is also split in different regions and the sampling method guarantees coverage of the whole credible range. Assume there are n parameters SA should be performed for and assume μ is the vector containing these n parameters. Then for each parameter its credible range is divided in M regions with equal probability and one sample is taken from each of those M regions. We are then able to construct a parameter vector μ_1 by taking for every parameter of the total of n parameters a random sample from the M samples taken for these parameters. With the unused samples a new parameter vector μ_2 can be constructed along the same lines and this process can be continued until M parameter vectors have been constructed. This process guarantees coverage of the whole credible range, since a sample is taken from each region of each parameter's credible range. The total number of sample points that needs to be generated equals nM, which for large m and N is a significant improvement from the original M^n.

Since the outputs of the objective function depend on parameter values which are in

our case estimated by data of our created csv-files, and there no guarantee that these estimates are correct (see Appendix D), performing sensitivity analysis on our model might give conclusions that are not to be trusted. Thus, with the present estimates sensitivity analysis will not be performed. It is however important to note that sensitivity analysis is one of the main reasons for the need for fast optimization procedures. The above can be a guidance on how to perform SA in the case representative parameter estimates are present.

7. Discussion and concluding remarks

The problem of optimizing an invasive species network by allocating intervention effort to patches in a network was considered. Given a budget constraint on the total available effort, the optimization becomes a constraint optimization problem where the objective has no closed-form expression, implying analytic optimization methods do not work. Moreover, the complexity of an invasive species model will make the computational costs of simulation based optimization too expensive. A functional form based optimization approach was proposed, which uses elements of both analytic and simulation based optimization. The functional form based optimization is described and the performance is illustrated by the use an example: a detailed model of the dispersal dynamics of COTS on the GBR was presented, for which an optimal effort allocation was found using a functional form based optimization method. The methodology was tested with computational experiments. From the experiments on small example networks it could be concluded that the methodology yields good results in terms of optimality: most sample paths showed significant decrease in the estimated objective function value. In terms of CPU time we have concluded that the proposed functional form based optimization significantly outperforms stochastic approximation in the small example networks. In the larger 20-patch network the same conclusion could be drawn. However, more experiments on large networks have to be performed to draw a real conclusion about the use of the functional form approach on large networks.

It should be noted that the model should be fitted with real-world data, to give the results meaning. Since there are no explicit expressions available for the parameters of the COTS model, the estimated parameters as used in the experiments in Chapter 6 are not representative. Further research should be conducted such that representative expressions or estimations for the parameters can be given. Only with proper expressions of the parameters the results of the functional form based methodology will have meaning. If the real-world data would for example suggest the dependence of coral and COTS to be large, this would influence the performance of the approximation model and coral has to be a part of the approximate model as well. Our proposal for further research would thus be to implement an even more complex model for COTS ecology which includes expressions for all the parameters and perform sensitivity analysis and functional form based optimization on this complex model to investigate the model itself and the performance of the proposed optimization algorithm

The approximation model was chosen to have as objective function the maximum number of expected adult COTS. As discussed in Chapter 4, the variance of the approximate model can also be computed explicitly to give an idea of the spread. Since we only used the location parameter in the objective function but the objective function of the detailed COTS model is a probability and thus dependent on both location

and spread, another choice of objective function based on both the location and spread might improve our results even more. Different distributions were already considered in constructing the approximate objective function, e.g. the normal, Pareto and binomial distribution. These distributions did however not fit the distribution of the adult COTS in our model well. Further research can be conducted on improving the approximate objective function, such that it uses information on both location and spread and is a good fit for f. This might reduce the number of iterations necessary to approximate the optimal diver allocation.

An optimal effort allocation was defined as an allocation which is feasible and minimizes the objective function f. In some situations multiple optimal effort allocations are present. Our algorithm tries to find one of these optimal effort allocations. In terms of costs it is however true that there are optimal effort allocations that are more preferred than others: when in a 3-patch network both [0.1,0.1,0.1] and [0.2,0.2,0.2] are optimal effort allocations, the former will be preferred over the latter since the costs will be lower. Future research can be conducted on how to adapt the algorithm such that not only an optimal effort allocation is found, but the optimal effort allocation with minimal costs.

For the optimization of the approximate objective function in the functional form based optimization loop we have looked at two different algorithms: backtracking line search and the NITRO algorithm. The NITRO algorithm was already developed in 1997, but is still used in Python to optimize inequality constraint multivariate optimization problems. The BLS algorithm is also relatively old. The performance of our methodology under these two optimization algorithms already resulted in fast optimization in terms of CPU time. There might however be more recent algorithms outperforming both algorithms in optimization CPU time. Besides the performance of the optimization algorithm implementation of both the detailed model and the approximate model is important in terms of CPU times. The Python code with the implemented models as used in the numerical experiments is given in Appendix D. It should be noted that there might be other implementations possible that yield better results in terms of CPU times.

The example of optimization of the objective of the COTS model was presented as guiding example for the optimization of minimizing colonization risk in invasive species networks. Invasive species models can be represented with the same structure as our presented COTS model: there is a network of habitat patches which can get colonized by introduction of an invasive species in the network which then disperses through the network. Applying the functional form based methodology on any invasive species management problem will consist of the same steps. Moreover, assuming the invasive species model does indeed have the same structure as the COTS model, finding an appropriate approximation model can be done along the same lines as for the COTS model.

Popular summary

We consider habitat networks where an invasive species has been introduced to the network at a certain patch and is able to disperse through the network, colonizing other patches and thereby harming the ecosystem. We apply this to an example of crown-of-thorns starfish (COTS) outbreaks on the Great Barrier Reef (GBR): COTS are an important cause of coral reef decline on the GBR and their spatial dynamics is similar to those observed in invasive species. It is assumed to be possible to eradicate COTS at certain patches in the network by applying control effort at specific locations. Having a budget constraint, eradication effort can only be conducted on a subset of patches and this requires deciding on which patches to intervene. The goal is to take actions to minimize the risk of adult COTS colonization on certain 'high-value' patches. Define $w_i = 1$ for high-value patches and $w_i = 0$ otherwise. Let $A_i(x, t)$ be the number of adult COTS that are present at patch i at the end of management interval t given a feasible diver-decision x. Then the objective can be written as:

$$\min_{x \in \mathcal{X}} \max_{i} \; w_i \mathbb{P}(A_i(x, t) > 0).$$

Firstly a detailed stochastic model is created, to simulate the spatial dynamics of COTS populations on the GBR, considering different age groups and dispersal of larvae through the network. Because of the complexity of the detailed model the objective function has no closed-form expression and current optimization techniques do not yield a fast approximation to the optimum. In this thesis functional form based optimization is described for the optimization of the COTS model under study. The approach connects the detailed model with a tractable approximate model that can be optimized by fast numerical optimization methods. This tractable model has an extra parameter Λ used to fit the model output locally to that of the output of the detailed model. The idea is that instead of optimizing the detailed model, the tractable model is optimized. This is done iteratively: every iteration of the algorithm Λ is found such that the outputs match, after which the tractable model is optimized, yielding a new iterate.

A set of numerical experiments is conducted to compare the results of our proposed method in terms of optimality gaps and computation times with a well-known method called stochastic approximation. These numerical experiments conducted on the COTS model example show great promise, since a value close to the optimal value can be reached with a computation time improving on that of stochastic approximation.

The idea is now that our framework can be used for invasive species problems, since these problems often have a non-closed form objective as well. Thus having an invasive species model, our tractable model can be used as a guide on how to construct a tractable model for the detailed model under study, after which the functional form approach can be used to find an optimal allocation of intervention resources.

Bibliography

[1] Australian Institute of Marine Science. `https://www.aims.gov.au/`. Accessed April 15 2019.

[2] P. Amarasekare. The role of density-dependent dispersal in source-sink dynamics. *Journal of Theoretical Biology*, 226:159-168, 2004

[3] S. Asmussen and P.W. Glynn. Stochastic simulation: algorithms and analysis. *Springer Science Business Media, LLC*, 2007

[4] R.C. Babcock and C.N. Mundy. Reproductive biology, spawning and field fertilisation rates of *Acanthaster planci*. *Australian Journal of Marine and Freshwater Research*, 43:525-534, 1992

[5] C.M. Baker. Target the source: optimal spatiotemporal resource allocation for invasive species control. *Conversation Letter*, 10(1): 41-48, 2017

[6] F. Bijma, M. Jonker, and A. Van der Vaart. Inleiding in de Statistiek. *Epsilon uitgaven*, Utrecht, 2013

[7] J.R. Blum. Multidimensional stochastic approximation methods. *Ann. Math. Stat.*, 25(4): 737-744, 1954

[8] H. Byrd, M.E. Hribar, and J. Nocedal. An interior point algorithm for large scale nonlinear programming. *SIAM Journal on Optimization*, 9(4): 682-706, 1997

[9] S.A. Condie and J.R. Andrewartha. Circulation and connectivity on the Australian Norh West Shelf. *Continental Shelf Research*, 28:1724-1739, 2008

[10] CSIRO. Connie 3. `http://www.csiro.au/connie/`. Accessed April 20, 2019

[11] G. De'ath and P.J. Moran. Factors affecting the behaviour of crown-of-thorns starfish (Acanthaster planci L.) on the Great Barrier Reef. *Journal of Experimental Marine Biology and Ecology*, 220(1):83-106, 1998

[12] A. Dieker, S. Ghosh, and M.S. Squillante. Optimal resource capacity management for stochastic networks. *Operations Research*, 65(1):221-241, 2017

[13] D. Eberly. Derivative approximation by finite differences. `http://www.geometrictools.com/Documentation/FiniteDifferences.pdf`. Accessed August 20, 2019

[14] P. Gaona, P. Ferreras, an M. Delibes. Dynamics and viability of a metapopulation of the Endangered Iberian Lynx (Lynx pardinus). *Ecological Monographs*, 68(3): 349-370, 1998

[15] GBRMPA. *Reef facts*. `http://www.gbrmpa.gov.au/the-reef/reef-facts`. Accessed May 25, 2019

[16] J. Gilmour, C.W. Speed, and R. Babcock. Coral reproduction in Western Australia. *PeerJ*, 4:e2010; DOI 10.7717/peerj.2010, 2016

[17] D.F. Gleason, and D.K. Hofmann. Coral larvae: from gametes to recruits. *Journal of Experimental Marine Biology and Ecology*, 408(1-2):42-57, 2011

[18] M. Herzfeld. An alternative coordinate system for solving finite difference ocean models. *Ocean modelling*, 14(3-4): 174-196, 2006

[19] R.C. Highsmith. Reproduction by fragmentation in corals. *Marine Ecology Progress Series*, 7:207-226, 1982

[20] K. Hock, N.H. Wolff, R. Beeden, J. Hoey, S.A. Condie, K.R.N. Anthony, H.P. Possingham, and P.J. Mumby. Controlling range expansion in habitat networks by adaptively targeting source populations. *Conservation Biology*, 30(4):856-866, 2016

[21] K. Hock, N.H. Wolff, S.A. Condie, K.R.N. Anthony, and P.J. Mumby. Connectivity networks reveal the risks of crown-of-thorns starfish outbreaks on the Great Barrier Reef. *Journal of Applied Ecology*, 51:1188-1196, 2014

[22] M.K. James, P.R. Armsworth, L.B. Mason, and L. Bode. The structure of reef fish metapopulations: modelling larval dispersal and retention patterns. *Proc. R. Soc. Lond. B*, 269:2079-2086, 2002

[23] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Ann. Math. Stat.*, 23: 462-466, 1952

[24] A.S. Kough, C.B. Paris. The influence of spawning periodicity on population connectiviy. *Coral Reefs*, 34(3):753-757, 2015

[25] M. Lamare, D. Pecorino, N. Hardy, M. Liddy, M. Byrne, and S. Uthicke. The thermal tolerance of crown-of-thorns (*Acanthaster planci*) embryos and bipinnaria lavae: implications for spatial and temporal variation in adult populations. *Coral Reefs*, 33:207-219, 2014

[26] Mairal. J. Stochastic majorization-minimization algorithms for large-scale optimization. *Adv. Neural Information Processing Systems (NIPS)*, 2013

[27] E.B. Morello, E.E. Plagányi, R.C. Babcock, H. Sweatman, R. Hillary, and A. Punt. Model to manage and reduce crown-of-thorns starfish outbreaks. *Marine Ecology Progress Series*, 512:167-183, 2014

[28] J. Nocedal, and S.J. Wright. Numerical optimization. *Springer Science Business Media, LLC*, 2006

[29] J. Norton. An introduction to sensitivity assessment of simulation models. *Environmental Modelling and Software*, 69: 166-174, 2015

[30] B. Patch, M.S. Squillante, and P.M. van de Ven. Optimisation of stochastic networks with blocking: a function-form approach. *arXiv: 1904.05040*, submitted April 10th 2019

[31] M.S. Pratchett, C.F. Caballes, J.C. Wilmes, S. Matthews, C. Melling, H.P.A. Sweatman, et al. Thirty years of research on crown-of-thorns starfish (1986-2016): scientific advances and emerging opportunities. *Diversity*, 9, 41, 2017

[32] A.G. Smith, R. MicVinish, and P.K. Pollett. A model for a spatially structured metapopulation accounting for within patch dynamics. *Mathematical Biosciences*, 247:69-79, 2014

[33] P.J.C. Spreij. Measure theoretic probability. Lecture notes, version: August 28, 2017

[34] C.D. Storlazzi, M. van Ormondt, Y. Cheng, and E.P.L. Elias. Modeling fine-scale coral larval dispersal and interisland connectivity to help designate mutually-supporting coral reef marine protected areas: insights from Maui Nui, Hawaii. *Frontiers in Marine Science*, 4:381, 2017

[35] X. Su, X. Yan, and C. Tsai. Linear regression. *WIREs Comput Stat* 2012, 4:275-294. doi: 10.1002/wics.1198

[36] E.A. Treml, P.N. Halpin, D.L. Urban, and L.F. Pratson. Modeling population connectivity by ocean currents, a graph-theoretic approach for marine conservation. *Landscape ecology*, 23(Suppl 1):19, 2008

[37] J. Vanhatalo, G.R. Hosack, and H.P.A. Sweatman. Spatiotemporal modelling of crown-of-thorns starfish outbreaks on the Great Barrier Reef to inform control strategies. *Journal of Applied Ecology*, 54:188-197, 2017

[38] J.R. Watson, B.E. Kendall, D.A. Siegel, and S. Mitarai. Changing seascapes, stochastic connectiviy, and marine metapopulation dynamics. *The American Naturalist*, 180(1):99-112, 2012

[39] P.D. Williams, and A. Hastings. Stochastic Dispersal and Population Persistence in Marine Organisms. *The American Naturalist*, 182(2):271-282, 2013

[40] J. Wilmes, S. Matthews, D. Schultz, V. Mesmer, A. Hoey, and M. Pratchett. Modelling growth of juvenile crown-of-thorns starfish on the Northern Great Barrier Reef. *Diversity*, 9(1), 1; doi:10.3390/d9010001, 2017

[41] S. Wood, C.B. Paris, A. Ridgwell, and E.J. Hendy. Modelling dispersal and connectivity of broadcast spawning corals at the global scale. *Global ecology and biogeography*, 23:1-11, 2014

A. Estimating the parameters in the model

The model parameters in our detailed model as presented in Chapter 2 need to be estimated. For the estimation we divide the parameters in three classes: survival probabilities, reproduction rates and dispersal probabilities. Below a description for the estimation of the parameters in all three classes is given as used in the implementation of the experiments. As mentioned in the text it should be noted that these estimations are not representative since in reality they will depend on a complex background process.

Survival probabilities
The available data on survival probabilities will be binary: every time step a unit of COTS population either has died (0) or survived (1). One of the most common methods to analyze binary data is logistic regression. We have assumed that the parameters of the COTS model presented in Chapter 2 depend on factors, both from the predator-prey relation between COTS and coral and a background process. The specific factors used in our implementation of logistic regression are presented in Table A.1, with literature references for the factors of the background processes. Factors identified by future research can be incorporated easily.

Logistic regression can be used in situations in which a probability has to be estimated which is dependent on factors. Let p be the probability to estimate and $x = (x_1, \ldots, x_p)$ the factors p depends on. Let X be a Bernoulli random variable having value 1 with probability p. The pdf of X can then be expressed as

$$f(y; p) = p^y (1-p)^{1-y}.$$

Let $X_1, \ldots, X_n$ be n observations of X. Then the GLM framework for the logistic regression with the canonical link function looks as follows:

(i) Random component: $X_i \sim f(y; p)$

(ii) Systematic component: $\eta_i = \beta x'$, $k = 1, \ldots, n$

(iii) Link function: $\eta_i = \log(\frac{p}{1-p})$, $k = 1, \ldots, n$.

From (ii) and (iii) now

$$\log \frac{p}{1-p} = \beta x'$$

Parameter function	Factors
p_i^a, p_i^j, p_i^l	Temperature, salinity [31], coral cover
p_i^c	Temperature [16], salinity [17], adult COTS density
b_i	Temperature [25]
b_i^c	Temperature [17]
$p_{ij,d}$	Temperature [25], salinity [31]
$p_{ij,d}^c$	Temperature [41], salinity [17]

Table A.1.: Factors of the parameter functions

which implies

$$p = \frac{e^{\beta x'}}{1 + e^{\beta x'}}.$$

Estimation of β now gives a value for p and estimation can be done using maximum likelihood estimation. The log-likelihood function can be written as

$$\ell(p) = \sum_{i=1}^{n} \log f(y_i; p) = \sum_{i=1}^{n} y_i \log(\frac{p}{1-p}) + \log(1-p) = \sum_{i=1}^{n} y_i \beta x' - \log(1 + e^{\beta x'}).$$

Differentiating with respect to β gives

$$\frac{\partial}{\partial \beta_k} \ell(\beta) = \sum_{i=1}^{n} y_i x_k - \frac{e^{\beta x'}}{1 + e^{\beta x'}} x_k$$
$$\frac{\partial^2}{\partial \beta_k^2} \ell(\beta) = \sum_{i=1}^{n} -x_k^2 \frac{e^{\beta x'}}{(1 + e^{\beta x'})^2} < 0$$

from which can be concluded that for β maximizing ℓ it holds that for every k

$$\sum_{i=1}^{n} y_i x_k = \sum_{i=1}^{n} \frac{e^{\beta x'}}{1 + e^{\beta x'}} x_k.$$

Since these equations can't be solved exact, the MLE β will be computed numerically. This can be done in various ways, among which the Fisher scoring method and the Newton-Raphson method.

Reproduction rates
Logistic regression falls under the umbrella of generalized linear models (GLM). As becomes apparent from the name, a generalized linear model is a generalization of an ordinary linear model in which the error distribution is assumed to be normal. We will use this type of model to model the reproduction rates b_i and b_i^c and the parameter function f_i^c representing the growth of coral cover by asexual reproduction. Realistic birth or fertilization rates are hard to infer, since studies have shown that a female COTS

is able to produce over 100 million eggs per year [31] and it has furthermore been shown that a factor as temperature influences the fertilization rates [17, 31]. A natural way to model the reproduction rates is then with an ordinary linear model. The method to estimate the rates by using ordinary linear models is described in Appendix A. Factors identified by the literature and incorporated in our model are presented in Table A.1. Again, factors identified by future research can be incorporated easily.

Let the objective be to estimate y and let $x_i = (x_{i1}, \ldots, x_{ip})$, with x_{ij} a factor for y and p the total number of factors. Let there be available data $(y_1, x_1), \ldots, (y_n, x_n)$, y_i being the i-th response. Then the ordinary linear model with parameter vector $\beta = (\beta_0, \ldots, \beta_p)'$ is

$$y = X\beta + \epsilon$$

with $y = (y_1, \ldots, y_n)'$, $X = (x_{ij})_{n\times(p+1)}$ having $x_{i0} = 1$ and $\epsilon = (\epsilon_1, \ldots, \epsilon_n)$, with the assumption that $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$ for some $\sigma^2 \in \mathbb{R}_{\geq 0}$. Just as with logistic regression it now holds that a value for y can be found once the parameter vector β and the variance σ^2 are estimated. One method to do this, given that X is full rank, is by maximum likelihood estimation, using that

$$y|X \sim \mathcal{N}(X\beta, \sigma^2 I).$$

This gives the log-likelihood function

$$\ell(\beta, \sigma^2) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{(y - X\beta)'(y - X\beta)}{2\sigma^2}.$$

Differentiating with respect to β and σ^2 gives

$$\frac{\partial}{\partial\beta}\ell(\beta, \sigma^2) = -\frac{1}{\sigma^2}X'(y - X\beta)$$
$$\frac{\partial}{\partial\sigma^2}\ell(\beta, \sigma^2) = -\frac{n}{2\sigma^2} + \frac{||y - X\beta||^2}{2\sigma^4}$$

from which it is easily deducted that

$$\beta = (X'X)^{-1}X'y$$
$$\sigma^2 = \frac{||y - X\beta||^2}{n}$$

are the maximum likelihood estimators. $(X'X)^{-1}$ only exists if X is of full rank. However X can always be chosen to be of full rank. This is because when X is not full rank this indicates collinearity. By deleting or merging columns X can in that case be made full rank [6].

Dispersal probabilities

There has been quite a lot of research conducted on the topic of connectivity in marine environments (e.g. [21, 20, 18, 22, 36]). It has however turned out that the realized

patterns of connectivity in the GBR are very difficult to infer. Ocean currents play a significant role in larval dispersal and thus the structure of the connectivity network. There are however other factors that play a role in larval dispersal, e.g. tides, winds [21] and temperature [41], which also have to be incorporated in the model. The research on connectivity networks in marine environments has generally simulated dispersal with the help of oceanographic models, which are used to produce system-wide connectivity matrices. Here we will focus on the SHOC oceanographic model, which has been developed to simulate GBR hydrodynamics and has an online interface for visualizing simple dispersal simulations named Connie 3.

The hydrodynamic SHOC (Sparse Hydrodynamic Ocean Code) model can be used to estimate the dispersal probabilities, but was originally designed for more general purposes and models hydrodynamic forces as a three-dimensional model of velocity fields. With the right inputs the model can be used as a tool to obtain the COTS connectivity network of the GBR. The SHOC model has been developed as part of the CSIRO Environmental Modelling Suite resulting in the online tool Connie 3 [10]. Connie 3 uses data of realized ocean currents to estimate dispersal probabilities. Particle tracking is done by planting 100 particles per grid cell per day over the release period specified by the user. To find the horizontal velocity a fourth-order Runge-Kutta ODE solver is used per particle. The user is also able to add horizontal velocity by specifying behaviour of the particles [10]. An example of particle behaviour is the depth, which can be used to model the fact that coral larvae are located in the ocean surface, since they are attracted to the light. Thus, to find dispersal probabilities, the user can specify the patch from which the larvae will disperse, the dispersal time in terms of number of days the larvae will spend in the water and certain behaviour of the larvae and Connie 3 will output a .csv-file with the dispersal probabilities of all locations with non-zero dispersal probability and the coördinates of these locations. For certain types of marine species, among which COTS and coral, the behaviour and dispersal time is included using values from the literature, and the user is able to select these species without specifying behaviour and dispersal time. Models like these can be used to obtain more realistic dispersal probabilities to which algorithms such as those proposed in this study can then be applied. However, considerations to obtain such models are beyond the scope of the current study.

B. Parameters of the NITRO algorithm

In section 5.3 the NITRO algorithm is described. In the algorithm as designed by Byrd, Hribar and Nocedal there are multiple procedures in the algorithm and multiple parameters for which an expression is needed. The derivations of the expressions for these parameters will be given here.

Least squares multiplier estimates
The vector $\lambda = (\lambda_g, \lambda_h)$ is the vector of multiplier estimates which will be obtained using a least squares method the perturbed KKT conditions on the Lagrangian of the barrier problem. The Karush-Kuhn-Tucker (KKT) conditions of the Lagrangian are given by

$$\begin{aligned} \nabla f(x^*) + A_h(x^*)\lambda_h + A_g(x^*)\lambda_g &= 0 \\ -\mu(S^*)^{-1}e + \lambda_g &= 0 \\ h(x) &= 0 \\ g(x) + s &= 0 \end{aligned}$$

with (x^*, s^*) the optimal solution. The perturbed KKT conditions are now given by multiplication of the second identity with S^*, which yields

$$\begin{aligned} \nabla f(x^*) + A_h(x^*)\lambda_h + A_g(x^*)\lambda_g &= 0 \\ -\mu e + S^*\lambda_g &= 0 \\ h(x) &= 0 \\ g(x) + s &= 0. \end{aligned}$$

Now, finding the vector λ that minimizes the Euclidean norm implies solving

$$\min_{\lambda} ||\nabla f(x^*) + A_h(x^*)\lambda_h + A_g(x^*)\lambda_g||_2^2 + || - \mu e + S^*\lambda_g||_2^2.$$

Expanding the norms and setting the derivatives with respect to λ equal to 0 yields the set of equations

$$\begin{aligned} A_h^T(x^*)A_h(x^*)\lambda_h + A_h^T(x^*)A_g(x^*)\lambda_g &= -A_h^T(x^*)\nabla f(x^*) \\ A_g^T(x^*)A_h(x^*)\lambda_h + A_g^T(x^*)A_g(x^*)\lambda_g + S^2\lambda_g &= -A_g^T(x^*)\nabla f(x^*) + \mu Se \end{aligned}$$

from which it can easily be derived that

$$\lambda = (\hat{A}^T\hat{A})^{-1}\hat{A}^T \begin{pmatrix} -\nabla f(x^*) \\ \mu e \end{pmatrix}$$

with

$$\hat{A} = \begin{pmatrix} A_h(x^*) & A_g(x^*) \\ 0 & S^* \end{pmatrix}.$$

The problem that this solution faces is that (x^*, s^*) is unknown. In SQP methods this is solved by using the current iterate in the computation of the estimates. Thus, for every iterate (x_k, s_k), the Lagrangian multiplier estimates are given by

$$\lambda_k = (\hat{A}_k^T \hat{A}_k)^{-1} \hat{A}_k^T \begin{pmatrix} -\nabla f(x_k) \\ \mu e \end{pmatrix}$$

with

$$\hat{A}_k = \begin{pmatrix} A_h(x_k) & A_g(x_k) \\ 0 & S_k \end{pmatrix}.$$

Trust region
A trust region is defined with two objectives. The first objective was already mentioned in Section 5.3, namely that the quadratic model is believed to be a good approximation within the trust region, and that moreover the linear approximations of the constraints are also believed to be good approximations within this trust region. Thus, to make sure the results can be trusted a Δ_k is chosen such that $||d|| \leq \Delta_k$. A scaling is added, such that the slack variables do not approach 0 too early, which changes the shape of the trust region. The scaled trust region will be given by

$$||(d_x, S_k^{-1} d_s||_2 \leq \Delta_k.$$

The second objective concerns the slack variables. A condition on the slack variables is that they should remain positive. The NITRO algorithm makes sure of this by using the fraction to the boundary rule given by

$$\frac{(1-\tau)s_k}{s_k + d_s} \geq 0$$

with $\tau \in (0, 1)$. This can be rewritten as $d_s \geq -\tau s_k$. In conclusion, the trust region should have the form

$$||(d_x, S_k^{-1} d_s||_2 \leq \Delta_k, \qquad d_s \geq -\tau s_k.$$

Computation of the vertical step via the dogleg method
In the computation of the vertical step v the following minimization problem arises (see Section 5.3 for details):

$$\begin{aligned} \min_v \quad & \tilde{f}_{con} := 2 \begin{pmatrix} h^T & (g(x_k) + s_k)^T \end{pmatrix} \hat{A}^T \begin{pmatrix} v_x \\ S_k^{-1} v_s \end{pmatrix} + \begin{pmatrix} v_x^T & v_s^T S_k^{-1} \end{pmatrix} \hat{A}\hat{A}^T \begin{pmatrix} v_x \\ S_k^{-1} v_s \end{pmatrix} \\ \text{subject to} \quad & ||(v_x, S_k^{-1} v_s)||_2 \leq \xi \Delta_k \\ & v_s \geq -\tau s_k \end{aligned}$$

for which an approximate solution has to be computed resulting in v. The approximate solution is computed with a dogleg method. The first step in the dogleg method is computation of the Cauchy point. An algorithm for the computation of the Cauchy point is presented in [28]. For our minimization problem this algorithm gives the Cauchy point

$$\tilde{v}^{cp} = -\alpha \tilde{A}_k \begin{pmatrix} h \\ g+s \end{pmatrix}$$

with

$$\alpha = \frac{||\hat{A}_k \begin{pmatrix} h \\ g+s \end{pmatrix}||_2^2}{\begin{pmatrix} h^T & (g+s)^T \end{pmatrix} (\hat{A}_k^T \hat{A}_k)^2 \begin{pmatrix} h \\ g+s \end{pmatrix}}.$$

The second step of the dogleg method computes the Newton step, denoted by $\tilde{v}^N$, which is the norm minimizer of $\tilde{f}_{con}$. An easy computations shows that

$$\tilde{v}^N = -\hat{A}(\hat{A}^T\hat{A})^{-1} \begin{pmatrix} h \\ g+s \end{pmatrix}.$$

The dogleg path is now the path from $\tilde{v} = 0$ to $\tilde{v}^{cp}$ and from $\tilde{v}^{cp}$ to $\tilde{v}^N$. The dogleg step is the point on this path which mimizes $\tilde{f}_{con}$ most and satisfies the constraints

$$||(v_x, S_k^{-1} v_s)||_2 \leq \xi \Delta_k, \qquad v_s \geq -\tau s_k.$$

Because of the constraint $v_s \geq -\tau s_k$, this dogleg step is possibly a short step. When this is the case, one can compute the largest $\theta \in (0, 1]$ for which $\theta \tilde{v}^N$ is feasible and take this truncation as the dogleg step, if this gives a lower value for $\tilde{f}_{con}$.

Computation of the horizontal step via CG iteration
Recall the horizontal subproblem

$$\begin{aligned} \min_{\tilde{w}} \quad & q(\tilde{v} + \tilde{w}) \\ \text{subject to} \quad & A_h^T(x_k) w_x = 0 \\ & A_g^T(x_k) w_x + S\tilde{w}_s = 0 \\ & ||\tilde{w}||_2^2 \leq \Delta^2 - ||\tilde{v}||_2^2 \\ & \tilde{w}_s \geq \tau e - \tilde{v}_s \end{aligned}$$

and the fact that we required $\hat{A}_k^T \tilde{w} = 0$. Thus $\tilde{w}$ is by definition an element of the null-space of $\hat{A}_k^T$ and can be written as $\tilde{w} = Zu := \begin{pmatrix} Z_x & Z_s \end{pmatrix}^T u$ for a vector $u \in \mathbb{R}^{n-t}$ and Z a basis for the null-space of $\hat{A}^T$. Thus the subproblem becomes

$$\begin{aligned} \min_{u} \quad & q(\tilde{v} + Zu) \\ \text{subject to} \quad & ||Zu||_2^2 \leq \Delta^2 - ||\tilde{v}||_2^2 \\ & Z_s u \geq \tau e - \tilde{v}_s. \end{aligned}$$

The conjugate gradient iteration now takes the form

$$u^+ = u + \alpha\tilde{p},$$

which we recognize from the line search algorithm, since α will be the minimized step size along the direction $\tilde{p}$. For the direction $\tilde{p}$ the iteration

$$\tilde{p}^+ = -(Z^TZ)^{-1}Z^T\nabla q(\tilde{v} + Zu) + \beta\tilde{p}$$

is used. Here $Z^T\nabla q(\tilde{v} + Zu)$ is the derivative of the objective q with respect to u and is $-(Z^TZ)^{-1}$ added as preconditioning, β to maintain conjugacy. Details for this specific choice of $\tilde{p}^+$ are outside the scope of this thesis and can be found in the article by Byrd, Hribral and Nocedal. A methodology for speeding up these matrix multiplications can also be found there.
Since the constraint

$$||\tilde{w}||_2^2 \leq \Delta^2 - ||\tilde{v}||_2^2$$

still needs to be fulfilled, a Steihaug's stopping test if performed in the CG-iteration. This stopping test stops the iteration if the iterates do not fulfill this constraint any longer. It moreover stops the iteration if the gradient of q becomes too small, making the step unneccessary, or if the direction is of negative curvature.

C. Proof of multidimensional SA

Let Y_x be a random variable whose distribution depends on the vector x. Let $M(x)$ be the expectation of Y_x given the vector x. The goal of stochastic approximation is to find x^* such that M is maximized. M has in this case no closed-form, but observations of Y_x can be sampled. There are two sets of assumptions under which approximation of the SA algorithm to the maximizer x^* of M can be proven. In both proofs the following corollary is used, which is a consequence of Doob's martingale convergence theorem [33]:

Corollary C.0.1. *Let X_n be a sequence of integrable random variables which satisfy*

$$\sum_{n=1}^{\infty} \mathbb{E}[\mathbb{E}[X_{n+1} - X_n | X_1, \ldots, X_n]^+] < \infty$$

($X^+ = \frac{1}{2}X + \frac{1}{2}|X|$) and are bounded below uniformly in n. Then X_n converges a.s. to a random variable.

This corollary will first be used to prove the theorem presented below. Let f be a real-valued function. Define $D(x)$ and $A(x)$ as respectively the matrix with first partial derivatives and second partial derivatives of f. Denote $U(x) = \langle D(x), Y_x \rangle$ and

$$V_a(x) = \mathbb{E}[\langle Y_x, A(x + \theta a Y_x) Y_x \rangle].$$

Define Y_n as the distribution of Y_x when $x = X_n$ and define U_n and V_n in the same way. Moreover, let $Z_n = f(X_n)$.

Theorem C.0.2. *If the sequence a_n satisfies*

$$\sum_{n=1}^{\infty} a_n = \infty, \quad \sum_{n=1}^{\infty} a_n^2 < \infty \tag{C.1}$$

and if there exists a real-valued function $f(x)$ with continuous first and second partial derivatives satisfying

$$Z_x \geq 0, \tag{C.2}$$

$$\sup_{||x|| \geq \epsilon} U(x) < 0, \tag{C.3}$$

$$\inf_{||x|| \geq \epsilon} |Z_x - Z_0| > 0, \tag{C.4}$$

$$V_a(x) \leq V < \infty, \tag{C.5}$$

for all a and all $\epsilon > 0$. Then the sequence $\{X_n\}$ converges to zero a.s.

Proof. Let $a \in \mathbb{R}$, then the Taylor approximation of f around x for $a \in \mathbb{R}$ is given by

$$f(x + aY_x) = f(x) + a\langle D(x), Y_x\rangle + \frac{1}{2}a^2\langle Y_x, A(x + \theta aY_x)Y_x\rangle$$

with $\theta \in [0, 1]$. This implies

$$\begin{aligned}\mathbb{E}f(x + aY_x) &= f(x) + a\langle D(x), M(x)\rangle + \frac{1}{2}a^2\mathbb{E}[\langle Y_x, A(x + \theta aY_x)Y_x\rangle] \\ &= f(x) + aU(x) + \frac{1}{2}a^2V_a(x).\end{aligned}$$

Filling in X_n for x and a_n for a results in

$$\mathbb{E}f(X_n + a_nY_{X_n}) = f(X_n) + a_nU(X_n) + \frac{1}{2}a_n^2V_{a_n}(X_n).$$

Then, considering the iteration, the following equation holds true:

$$\mathbb{E}[Z_{n+1}|Z_1, \ldots, Z_n] = Z_n + a_n\mathbb{E}[U_n|Z_1, \ldots, Z_n] + \frac{1}{2}a_n^2\mathbb{E}[V_n|Z_1, \ldots, Z_n] \quad a.s. \tag{C.6}$$

By the assumption $M(0) = 0$ and (C.3) we know that

$$\mathbb{E}[U_n|Z_1, \ldots, Z_n] \leq 0 \quad a.s. \tag{C.7}$$

Furthermore, by (C.5) it also holds that for all n

$$\mathbb{E}[V_n|Z_1, \ldots, Z_n] \leq V \quad a.s. \tag{C.8}$$

This implies

$$\mathbb{E}[Z_{n+1} - Z_n|Z_1, \ldots, Z_n] \leq \frac{1}{2}a_n^2V \quad a.s.$$

Now, since V can be assumed to be non-negative, it holds that

$$\sum_{n=1}^{\infty}\mathbb{E}[\mathbb{E}[Z_{n+1} - Z_n|Z_1, \ldots, Z_n]^+] \leq \sum_{n=1}^{\infty}\frac{1}{2}a_n^2V = \frac{1}{2}V\sum_{n=1}^{\infty}a_n^2 < \infty,$$

where the conclusion follows from (C.1). Now, by (C.2) and Corollary C.0.1 we know that Z_n converges a.s. to a random variable. The next step will be to prove that Z_n converges to Z_0 a.s., since then by the continuity of f and (C.4) X_n converges to 0 a.s., which is exactly what needs to be proven.

Iteration over (C.6) and again taking expectations yields

$$\mathbb{E}Z_{n+1} = Z_1 + \sum_{i=1}^{n}a_i\mathbb{E}U_i + \frac{1}{2}\sum_{i=1}^{n}a_i^2\mathbb{E}V_i. \tag{C.9}$$

Taking expectation over (C.7), (C.8) and (C.2) it yields furthermore that

$$\mathbb{E}Z_n \geq 0, \qquad \mathbb{E}U_n \leq 0, \qquad \mathbb{E}V_n \leq V$$

for all n. Since the left-hand side of (C.9) is non-negative and V is also non-negative, it can be concluded that $\sum_{i=1}^{\infty} a_i \mathbb{E}U_i$ must converge. Moreover, since $\sum_{i=1}^{\infty} a_i = \infty$ by assumption, this implies that

$$\limsup_{n\to\infty} \mathbb{E}U_n = 0 \qquad \liminf_{n\to\infty} \mathbb{E}|U_n| = 0.$$

Let $\{n_k\}$ be a sequence of integers such that $\lim_{k\to\infty} \mathbb{E}[|U_{n_k}|] = 0$, thus $U_{n_k} \to 0$ in $\mathcal{L}^1$. This implies that $U_{n_k} \to 0$ in probability, which is equivalent (Proposition 7.5 in [33]) with the fact that the subsequence $\{n_k\}$ contains a further subsequence $\{U_{m_k}\}$ that is a.s. convergent to 0. By assumption (C.3) this implies $\{X_{m_k}\}$ that is a.s. convergent to 0. By the continuity of f now $Z_n \to Z_0$. □

Now we again look at the problem of maximizing the function $M(x) = \mathbb{E}F(x)$. We assume without loss of generality that the maximum of M is at $x = 0$ and that $M(0) = 0$, implying $M(x) \leq 0$ for all x. Thus, a sequence $\{X_n\}$ has to be found for which $X_n \to 0$ a.s. Just as in the original one-dimensional proof of Kiefer and Wolfowitz two sequences $\{a_n\}$ and $\{c_n\}$ will be defined for which

$$\lim_{n\to\infty} c_n = 0, \quad \sum_{n=1}^{\infty} a_n = \infty, \quad \sum_{n=1}^{\infty} a_n c_n < \infty, \quad \sum_{n=1}^{\infty} a_n^2 c_n^{-2} < \infty$$

(for example $a_n = n^{-1}, c_n = n^{-1/3}$), which define the iteration as

$$X_{n+1} = X_n + a_n Y_{n,c_n},$$

where Y_{n,c_n} is the finite differences estimator given X_n and c_n, thus with i-th element given by

$$Y_{n,c_n}^{(i)} = \frac{Y_{X_n + c_n e_i} - Y_{X_n}}{c_n} =: c_n^{-1} Y_n.$$

Let us denote with $D(x)$ and $A(x)$ respectively the first and second partial derivatives of $M(x)$ (note: not of f as assumed previously). We again denote $D_n := D(X_n)$ and $A_n = A(X_n)$, where $\bar{A}_n$ is used as notation for the diagonal of A_n. Lastly we write $\Delta_n := \mathbb{E}[Y_n | X_n]$ and σ_x^2 for the variance of Y_x for a given input vector x. With this notation we are now able to present the other main theorem:

Theorem C.0.3. *Suppose the sequences $\{a_n\}$ and $\{c_n\}$ satisfy the above conditions and that furthermore*

(*i*) *$M(x)$ is itself continuous and has both continuous first and second derivatives, where the second partial derivatives are moreover bounded*

(ii) $\sigma_x^2 \leq \sigma^2 < \infty$

(iii) *for every* $\epsilon > 0$ *there exists a* $\rho > 0$ *such that*

$$||x|| \geq \epsilon \implies M(x) \leq -\rho \text{ and } ||D(x)|| \geq \rho.$$

Then the sequence $\{X_n\}$ *converges a.s. to 0.*

Proof. The Taylor expansion

$$M(X_n + a_n Y_{n,c_n}) = M(X_n) + a_n c_n^{-1}\langle D_n, Y_n\rangle + \frac{1}{2}a_n^2 c_n^{-2}\langle Y_n, A(X_n + \theta a_n c_n^{-1} Y_n)Y_n\rangle,$$

yields the relation (by taking conditional expectations)

$$\mathbb{E}[-M(X_{n+1}) = -M(X_n) - a_n c_n^{-1}\langle D_n, \Delta_n\rangle - \frac{1}{2}a_n^2 c_n^{-2}\mathbb{E}[\langle Y_n, A(X_n + \theta a_n c_n^{-1} Y_n)Y_n\rangle | X_n] \quad a.s.$$

By (i) and (ii) the matrix $A(x)$ and σ_x^2 are bounded, thus

$$\begin{aligned}|\mathbb{E}[\langle Y_n, A(X_n + \theta a_n c_n^{-1} Y_n)Y_n\rangle | X_n]| &\leq K|\mathbb{E}[\langle Y_n, Y_n\rangle | X_n] \\ &\leq K||\Delta_n||^2 + K_1\sigma_x^2 \leq K||\Delta_n||^2 + K_2,\end{aligned}$$

with K_1, K_2 positive constants. Again using a Taylor expansion the i-th component of Δ_n can be written as

$$\begin{aligned}\Delta_n^{(i)} &= \mathbb{E}[Y_{X_n + c_n e_i} - Y_{X_n}] \\ &= M(X_n + c_n e_i) - M(X_n) \\ &= c_n\langle D_n, e_i\rangle + \frac{1}{2}c_n^2\langle e_i, A(X_n + \theta^{(i)} c_n e_i)e_i\rangle\end{aligned}$$

where $\theta^{(i)} \in [0,1]$. This implies

$$\begin{aligned}\langle D_n, \Delta_n\rangle &= c_n||D_n||^2 + \frac{1}{2}c_n^2\langle D_n, \bar{A}_n\rangle \\ ||\Delta_n||^2 &= c_n^2||D_n||^2 + c_n^3\langle D_n, \bar{A}_n\rangle + \frac{1}{4}c_n^4||\bar{A}_n||^2.\end{aligned}$$

By (i) A is bounded and thus $\bar{A}_n$ is bounded. This means there exists a K_3 such that $||\bar{A}_n||^2 \leq K_3$ and this furthermore implies $||\langle D_n, \bar{A}_n\rangle||^2 \leq K_3||D_n||^2$. An easy computation now shows that with the derived bounds it holds that

$$\begin{aligned}\mathbb{E}[-M(X_{n+1})|X_n] \leq -M(X_n) - (||D_n||^2(a_n - \frac{1}{2}K_1 a_n^2) - ||D_n||K_3^{1/2}(\frac{1}{2}a_n c_n - \frac{1}{2}K_1 a_n^2 c_n)) \\ + \frac{1}{8}K_1 K_3 a_n^2 c_n^2 + \frac{1}{2}K_2 a_n^2 c_n^{-2} \quad a.s.\end{aligned}$$

Note that for n sufficiently large $a_n - \frac{1}{2}K_1 a_n^2$ and $\frac{1}{2}a_n c_n - \frac{1}{2}K_1 a_n^2 c_n$ are non-negative. Define a sequence of random variables μ_n as $\mu_n = 1$ for $||D_n|| \geq 1$ and $\mu_n = 0$ otherwise. Then by this definition for n sufficiently large it holds that

$$||D_n||^2(a_n - \frac{1}{2}K_1 a_n^2) - \mu_n ||D_n|| K_3^{1/2}(\frac{1}{2}a_n c_n - \frac{1}{2}K_1 a_n^2 c_n) \geq 0.$$

which yields the bound

$$\begin{aligned}\mathbb{E}[-M(X_{n+1})|X_n] \leq -M(X_n) - (1-\mu_n)(||D_n|| K_3^{1/2}(\frac{1}{2}a_n c_n - \frac{1}{2}K_1 a_n^2 c_n)) \\ + \frac{1}{8}K_1 K_3 a_n^2 c_n^2 + \frac{1}{2}K_2 a_n^2 c_n^{-2} \quad a.s.\end{aligned}$$

Taking conditional expectations on both sides gives

$$\begin{aligned}\mathbb{E}[-M(X_{n+1})|M(X_n)] \leq -M(X_n) - \mathbb{E}[(1-\mu_n)(||D_n|| K_3^{1/2}(\frac{1}{2}a_n c_n - \frac{1}{2}K_1 a_n^2 c_n))|M(X_n)] \\ + \frac{1}{8}K_1 K_3 a_n^2 c_n^2 + \frac{1}{2}K_2 a_n^2 c_n^{-2} \quad a.s.\end{aligned}$$

We have that

$$\begin{aligned}\sum_{n=1}^{\infty} \frac{1}{8}K_1 K_3 a_n^2 c_n^2 = \frac{1}{8}K_1 K_3 \sum_{n=1}^{\infty}(a_n c_n)^2 < \infty \\ \sum_{n=1}^{\infty} \frac{1}{2}K_2 a_n^2 c_n^{-2} = \frac{1}{2}K_2 \sum_{n=1}^{\infty} a_n^2 c_n^{-2} < \infty.\end{aligned}$$

Both conclusions follow from the conditions on the sequences $\{a_n\}$ and $\{c_n\}$. It also holds that

$$\begin{aligned}\sum_{n=1}^{\infty} \mathbb{E}[(1-\mu_n)(||D_n|| K_3^{1/2}(\frac{1}{2}a_n c_n - \frac{1}{2}K_1 a_n^2 c_n))|M(X_n)] \\ = \frac{1}{2}K_3^{1/2} \sum_{n=1}^{\infty}((a_n c_n - K_1 a_n^2 c_n))\mathbb{E}[(1-\mu_n)||D_n|||M(X_n)] \\ \leq \frac{1}{2}K_3^{1/2} \sum_{n=1}^{\infty}((a_n c_n - K_1 a_n^2 c_n)) < \infty \quad a.s.\end{aligned}$$

Thus, with these three bounds we can conclude that

$$\sum_{n=1}^{\infty} \mathbb{E}[\mathbb{E}[-M(X_{n+1}) - -M(X_n)|M(X_n)]^+] < \infty,$$

which means the sequence $-M(X_n)$ satisfies the first condition of Corollary C.0.1. This sequence is moreover bounded below uniformly in n, since by assumption $M(X_n) \leq 0$. Corollary C.0.1 now gives us that $-M(X_n)$ converges a.s. to a random variable,

which implies $M(X_n)$ also converges to a random variable. Now, since we know $M(X_n)$ converges a.s., $M(X_n) \leq 0$ and $\sum_{j=1}^{n} a_j$ diverges, the series

$$\sum_{j=1}^{n} a_j \mathbb{E}[||D_j||^2(1 - \frac{1}{2}K_1 a_j) - \mu_j ||D_j|| K_3^{1/2}(\frac{1}{2}c_j - \frac{1}{2}K_1 a_j c_j$$

converges. Since we also proved that every element of this sum is non-negative, there exists a subsequence $\{j_k\}$ such that $D_{j_k} \to 0$ a.s. By the assumed continuity of $D(x)$, this implies again that $X_{j_k} \to 0$ a.s., which by the continuity of $M(x)$ and the fact that we assumed $M(0) = 0$ implies that $M(X_n) \to 0$ a.s.

□

D. Data and Python-code

The computational experiments were all carried out in Python. Two .csv-files were created containing data on the parameters needed to perform regression. The first file contains binary data to perform logistic regression for the survival probabilities with explanatory variables *temperature*, *salinity* and *coral density* or *number of adult COTS*. The second file contains data to perform linear regression for the birth rates with explanatory variable *temperature*.

One of the main sources for marine data on the GBR is AIMS, the Australian Institute of Marine Science. AIMS collects data on e.g. coral, COTS, salinity and temperature of the water in the GBR. The data suggests that *temperature* should have a value between 22 and 34°C and *salinity* should have a value between 32 and 36 PSU [1]. These ranges are also used in creating the data on *temperature* and *salinity* for prediction of the survival probability or the number of born larvae. No specific relation between the factors of the background process and the explanatory variable was implemented.

In our network the capacities were always chosen as 100 maximum number of COTS population units and 50 for the coral cover units. This choice was based on two issues. Firstly, there were no studies (to the knowledge of the author) on the capacities of the reefs conducted so far, meaning no estimates from the literature can be used. Secondly it was a pragmatic choice: in the Python code truncated Poisson is performed which uses the faculty operation and Python is unable to handle the faculty operation for large numbers. When the functional form based algorithm is used on real-world data, more representative data can be used. The capacity of the reefs will then depend on factors as e.g. the area of the reef.

Now the Python-code will be presented used to perform the simulations.
Estimates of the objective of the real world model:

```
# Function that estimates the objective of the COTS model
def simulationcont(s,x,I):

    # intialize lambda
    Lambda = np.zeros((n_patches, n_patches))
    counter = np.zeros((n_patches, n_patches))

    # perform simulation multiple times to estimate expectation
    for z in range(2000):
        numbers = 'input initial vector'

        for t in range(s):
```

```
larvae_cots = np.zeros(n_patches)
larvae_coral = np.zeros(n_patches)
total_adults = np.zeros(n_patches)
total_cots = np.zeros(n_patches)
total_coral = np.zeros(n_patches)

# loop through all patches for death and births
for j in range(n_patches):
    n = numbers[j]
    total_cots[j] = n[0] + n[1] + n[2]
    total_adults[j] = n[0]
    total_coral[j] = n[3]

    # let spawning larvae COTS be born
    larvae_cots[j] = fb(b_cots[j],T,t)
    # let spawning larvae coral be born
    larvae_coral[j] = fb(b_coral[j],T,t)
    # compute new number of adults
    n[0] = np.random.binomial(n[0],
        f_a1(p_a1[j],S,T,total_coral[j],t,x[j]))
        +np.random.binomial(n[1],
        f_a1(p_j1[j],S,T,total_coral[j],t,x[j]))
    # compute new number of juveniles
    n[1] = np.random.binomial(n[2],
        f1(p_l1[j],S,T,total_coral[j],t))
    # set number of larvae to zero
    n[2] = 0
    # compute new density coral by coral born asexual
    n[3] = np.random.binomial(max(round(fb(b_asexual[j],T,t)),0),
        f1(p_c[j],S,T,total_adults[j],t))

    numbers[j] = n

# loop through all patches for larval dispersal
for j in range(n_patches):
    n = numbers[j]
    l_coral = 0
    lamda = [0]*n_patches

    for i in range(n_patches):
        # compute mean cots dispersing to j
        lamda[i] = total_adults[i]*larvae_cots[i]
            *f(p_disp[i*n_patches+j],S,T,t)*p[i][j]
        # coral to j
```

```
            l_coral += total_coral[i]*larvae_coral[i]
                *f(p_disp_c[i*n_patches+j],S,T,t)*p[i][j]

        # compute new number of larvae in patch j
        l_cots = sum(lamda)
        if(l_cots != 0):
            probs_cots = [((l_cots**i)/math.factorial(i)) for i
                in range(K[j] - int(total_cots[j]) + 1)]
            sum_probs = sum(probs_cots)
            probs_cots_trunc = [k/sum_probs for k in probs_cots]
            n[2] = trunc_poisson(range(K[j]
                - int(total_cots[j]) + 1),
                probs_cots_trunc)

        # update lambda
        for i in range(n_patches):
            if(total_adults[i] != 0):
                counter[i][j] += 1
                if(l_cots == 0):
                    Lambda[i][j] += 0
                else:
                    Lambda[i][j] += (n[2]*lamda[i])
                        /(l_cots*total_adults[i])

        # compute sexually born coral
        if(K_c[j] - int(total_coral[j]) + 1 > 0):
            probs_coral = [((l_coral**i)/math.factorial(i))
                for i in range(K_c[j]
                - int(total_coral[j]) + 1)]
            sum_probs_c = sum(probs_coral)
            probs_coral_trunc = [k/sum_probs_c
                for k in probs_coral]
            n[3] += trunc_poisson(range(K_c[j]
                - int(total_coral[j]) + 1),
                probs_coral_trunc)

        numbers[j] = n

# decide on presence of adult COTS
present = [0] * n_patches
for w in I:
    if(numbers[w-1][0] > 0):
        present[w-1] = 1
if(z == 0):
```

```
            probability = np.array(present)
        else:
            probability += np.array(present)

    # compute lambda
    for i in range(n_patches):
        for j in range(n_patches):
            if(counter[i][j] != 0):
                Lambda[i][j] = Lambda[i][j]/counter[i][j]

    return(max(probability/2000), Lambda)
```

Computing the objective of the approximate model:

```
# Function which computes the MGF
def g_t(k,h,x,numbers,t,lam):

# Computation first 3 time steps
    tau_a = [0]*n_patches
    tau_l = [0]*n_patches
    tau_j = [0]*n_patches

    tau_l[k-1] = np.log(1 - f(p_l[k-1],S,T,t-2) + f(p_l[k-1],S,T,t-2)
        *(1 - f_a(p_j[k-1],S,T,t-1,x[k-1]) + f_a(p_j[k-1],S,T,t-1,x[k-1])
        *(1 - f_a(p_a[k-1],S,T,t,x[k-1])
        + f_a(p_a[k-1],S,T,t,x[k-1])*np.exp(h))))
    tau_j[k-1] = np.log(1 - f_a(p_j[k-1],S,T,t-2,x[k-1])
        + f_a(p_j[k-1],S,T,t-2,x[k-1])*(1 - f_a(p_a[k-1],S,T,t-1,x[k-1])
        + f_a(p_a[k-1],S,T,t-1,x[k-1])*(1 - f_a(p_a[k-1],S,T,t,x[k-1])
        + f_a(p_a[k-1],S,T,t,x[k-1])*np.exp(h))))
    h_l = - f(p_l[k-1],S,T,t-1) + f(p_l[k-1],S,T,t-1)
        *(1 - f_a(p_j[k-1],S,T,t,x[k-1])
        + f_a(p_j[k-1],S,T,t,x[k-1])* np.exp(h))
    for i in range(n_patches):
        tau_a[i] = lam[i][k-1]*h_l
    tau_a[k-1] += np.log(1 - f_a(p_a[k-1],S,T,t-2,x[k-1])
        + f_a(p_a[k-1],S,T,t-2,x[k-1])
        *(1 - f_a(p_a[k-1],S,T,t-1,x[k-1]) + f_a(p_a[k-1],S,T,t-1,x[k-1])
        *(1 - f_a(p_a[k-1],S,T,t,x[k-1]) + f_a(p_a[k-1],S,T,t,x[k-1])
        *np.exp(logsumexp(h)))))

# recursion with time steps
    for s in range(t-3,0,-1):
        sum_a = [0]*n_patches
```

```
        for i in range(n_patches):
            for j in range(n_patches):
                sum_a[i] += lam[i][j]*(np.exp(logsumexp(tau_l[j])) - 1)

        for i in range(n_patches):
            tau_l[i] = np.log(1 - f(p_l[i],S,T,s)
                + f(p_l[i],S,T,s)*np.exp(logsumexp(tau_j[i])))
            tau_j[i] = np.log(1 - f_a(p_j[i],S,T,s,x[i])
                + f_a(p_j[i],S,T,s,x[i])*np.exp(logsumexp(tau_a[i])))
            tau_a[i] = np.log(1 - f_a(p_a[i],S,T,s,x[i])
                + f_a(p_a[i],S,T,s,x[i])*np.exp(logsumexp(tau_a[i])))
                + sum_a[i]

    adult_sum, juvenile_sum, larvae_sum = 0,0,0
    for i in range(n_patches):
        adult_sum += numbers[i][0]*tau_a[i]
        juvenile_sum += numbers[i][1]*tau_j[i]
        larvae_sum += numbers[i][2]*tau_l[i]

    return(np.exp(logsumexp(adult_sum) + logsumexp(juvenile_sum)
        + logsumexp(larvae_sum)))

# Function that computes mean and variance given MGF
def meancov(x,lam):
    numbers = 'input initial vector'
    t = 'input time horizon'

    h = 0.0001

    mean = [0] * n_patches

    # compute mean for i in I
    for i in I:
        e_h = [0] * n_patches
        e_h[i-1] = h
        f_h = g_t(t,n_patches,e_h,[0]*n_patches,[0]*n_patches,numbers,x,lam)
        mean[i-1] = (f_h - 1)/h

    return(max(mean))
```

Functional form based optimization using BLS:

```
# Function that computes optimum of f using BLS
def line_search_armijo_ap(f, lam, xk, rho, c1=1e-4):
```

```
n = 0
epsilon = 1

while((n < 50) and (epsilon > 0.025)):

    x1 = [xk[j] for j in range(n_patches)]

    phi_0 = f(xk,lam)
    if(phi_0 == 0):
        return(xk)

    # find pk
    grad_0 = [0]*n_patches
    for i in range(n_patches):
        xh = [xk[j] for j in range(n_patches)]
        xh[i] += 0.001
        linda = f(xh,lam)
        grad_0[i] = (linda - phi_0)/0.001
    pk = -np.array(grad_0)

    # find alpha
    alpha0 = 0.001
    phi_a0 = f(xk + alpha0*pk,lam)

    m = 0
    while(phi_a0 > phi_0 + c1*alpha0
        *np.dot(np.array(grad_0),np.array(pk))):
        alpha0 = rho*alpha0
        phi_a0 = f(xk + alpha0*pk,lam)
        m += 1
        if(m > 1000):
            return(xk)

    xk += alpha0*pk
    if(sum(xk) > 0.5):
        def distance(x):
            a = np.array(x) - np.array(xk)
            b = np.dot(np.transpose(a),a)
            return(b)
        lin_con = LinearConstraint([[1,1,1]],[0],[0.5])
        bounds = Bounds([0.0,0.0,0.0],[1.0,1.0,1.0])
        xk = scipy.optimize.minimize(distance, xk,
            method = 'SLSQP', constraints = [lin_con], bounds = bounds)
```

```
            xk = xk.x

        # compute increase and update
        epsilon = np.sqrt(np.sum(np.square(xk - x1)))
        n += 1
    return(xk)

# Functional form based optimization algorithm with BLS
def optimizesystem(n_patches,x_0,epsilon_1,time,I):

    x_1 = n_patches * [0]
    k = 0

    while((max(map(operator.sub, x_0, x_1)) > epsilon_1) or (k == 0)):
        x_1 = [x_0[i] for i in range(n_patches)]

        # decide on lambda by simulation
        sim = simulationcont(time,x_1,I)
        if(k == 0):
            path = [sim[0]]
            lam = sim[1]
        else:
            path.append(sim[0])
            lam = (lam + sim[1])/2
        if(sim[0] == 0):
            return(path,k)

        # optimize approx model for lambda
        x_0 = line_search_armijo_ap(meancov, lam, x_1, 0.5, c1=1e-4)

        k += 1
    return(path,k)
```

Functional form based optimization using NITRO:

```
# Functional form based optimization algorithm with NITRO
def optimizesystem2(n_patches,x_0,epsilon_1,time,I):

    x_1 = n_patches * [0]
    k = 0

    while((max(map(operator.sub, x_0, x_1)) > epsilon_1) or (k == 0)):
        x_1 = [x_0[i] for i in range(n_patches)]
```

```
        # decide on lambda by simulation
        sim = simulationcont(time,x_1,I)
        if(k == 0):
            path = [sim[0]]
            lam = sim[1]
        else:
            path.append(sim[0])
            lam = (lam + sim[1])/2
        if(sim[0] == 0):
            return(path,k)

        # optimize approx model for lambda
        bounds = Bounds([0.0,0.0,0.0],[1.0,1.0,1.0])
        lin_con = LinearConstraint([[1,1,1]],[0.0],[0.5])
        x_0 = minimize(meancov,x_1,args=(lam),method='trust-constr'
            ,constraints=[lin_con],bounds=bounds)
        x_0 = x_0.x

        k += 1
    return(path,k)
```

The stochastic approximation algorithm used:

```
def stoch_appr1(f,x0,epsilon,N):

# initialize
beta,rho_1,rho_2,D = 0.5,0.5,0.5,100

n = 1
g = epsilon + 1

while((g > epsilon) and (n <= N)):

    beta_n = beta*n**(-1/3)
    delta_n = n**(-1/6)
    x_n = [x0[j] for j in range(n_patches)]
    print(x_n)

    f0 = f(x0)
    if(n == 1):
        path = [f0]
    else:
        path.append(f0)
    if(f0 == 0):
```

```
    return(path)

# find gradient
grad_0 = [0]*n_patches
for i in range(n_patches):
    xh = [x0[j] for j in range(n_patches)]
    xh[i] += delta_n
    grad_0[i] = (f(xh) - f(x0))/delta_n
p_n = -np.array(grad_0)

# choose step size alpha
alpha_n = beta_n
d = 1
x_a = x0 + alpha_n*p_n
if(sum(x0 + alpha_n*p_n) > 0.5):
    def distance(x):
        a = np.array(x) - np.array(x0 + alpha_n*p_n)
        b = np.dot(np.transpose(a),a)
        return(b)
    lin_con = LinearConstraint([[1,1,1]],[0],[0.5])
    bounds = Bounds([0.0,0.0,0.0],[1.0,1.0,1.0])
    x_a = scipy.optimize.minimize(distance, x0, method = 'SLSQP',
        constraints = [lin_con], bounds = bounds)
    x_a = x_a.x
F = f(x_a)
f_a = f0 + rho_2*alpha_n*np.dot(np.array(grad_0),np.array(p_n))

while((F < f_a) and (d < D)):

    alpha_n = rho_1*alpha_n
    d += 1

    x_a = x0 + alpha_n*p_n
    if(sum(x0 + alpha_n*p_n) > 0.5):
        def distance(x):
            a = np.array(x) - np.array(x0 + alpha_n*p_n)
            b = np.dot(np.transpose(a),a)
            return(b)
        lin_con = LinearConstraint([[1,1,1]],[0],[0.5])
        bounds = Bounds([0.0,0.0,0.0],[1.0,1.0,1.0])
        x_a = scipy.optimize.minimize(distance, x0,
            method = 'SLSQP', constraints = [lin_con],
            bounds = bounds)
        x_a = x_a.x
```

```
            F = f(x_a)
            f_a = f0 + rho_2*alpha_n*np.dot(np.array(grad_0),
                np.array(p_n))

        x0 = x_a

        # compute new g and n
        g = np.sqrt(np.sum(np.square(x_n - x0)))
        n += 1

    return(path)
```