THE UNIVERSITY
of EDINBURGH

Cell-ebrating Efficiency: Optimisation of Biological Systems

Honours Project for the Degree of Bachelor of Science in Biological Sciences (Biotechnology)

B223626

May 2025 — 5992 words

Abstract

Cell-free protein synthesis allows us to harness the power of transcription and translation without the constraints associated with living cells. Recent advances have made this technology commercially viable for biomanufacturing a range of complex proteins, but the industry standard remains protein overexpression in bacteria. Here we show how optimisation algorithms can be applied to a simple cell-free expression system to improve performance across a range of applications. By changing the DNA concentration, energy concentration and dilution interval of the model system, we achieve a theoretical 3.5-fold increase in total GFP production and a 3-fold peak GFP concentration improvement compared to previous experiments. We also reduce the time taken for the system to reach steady state by 15-fold and increase the GFP concentration at steady state by 2.6-fold. We evaluate the suitability of different optimisation algorithms for improving cell-free technologies, finding that gradient-based approaches fail to accurately navigate discrete dilution cycles and that evolutionary algorithms converge best at optimum solutions across a variety of problems. This work improves our understanding of system dynamics and the optimal conditions for cell-free protein production, allowing us to increase resource efficiency and potentially reduce the costs associated with this emerging technology.

Acknowledgements

I would like to thank Andrea Weiße for giving me the opportunity to work on this project and for her guidance and support.

Thank you to Christoph Wagner for your insightful opinions, consistent patience, and for always being willing to help me debug my code and demystify my results.

To the rest of the Biomedical Computation Group, thank you for your feedback, help, and interest in this project during our weekly meetings, and for inspiring me with your own research and ideas.

Lastly, to my family, thank you for your unwavering belief and support.

Contents

Abbreviations

CFPS Cell-Free Protein Synthesis

TX Transcription

TL Translation

PURE Protein Synthesis Using Recombinant Elements

GFP Green Fluorescent Protein

NTP Nucleotide Triphosphate

ODE Ordinary Differential Equation

IP Interior Point

BBO Black-Box Optimisation

BFGS Broyden–Fletcher–Goldfarb–Shanno

LBFGS Limited-memory Broyden–Fletcher–Goldfarb–Shanno

EA Evolutionary Algorithm

Introduction

1.1 Cell-free protein synthesis

Cell-free protein synthesis (CFPS) involves biochemical reactions that occur independently of living cells and is increasingly used in engineering biology and biomanufacturing to produce complex proteins. Given that one of the fundamental goals of synthetic biology is the creation of a fully synthetic cell, the ability to build systems that can perform *in vitro* transcription and translation (TX-TL) is vital [1].

A variety of catalytic components are required for TX-TL: mRNA polymerase, ribosomes, translation factors, amino acyl-tRNA synthetases, tRNAs, amino acids, an energy regeneration system, and nucleotides [2]. CFPS systems can use extract-based sources of these components, commonly from *E. coli* cells, although unintended contaminants native to the cell can reduce yields and make modelling less accurate [3]. Degradation by native proteases and nucleases can also deplete DNA content and the protein product. Reconstituting protein synthesis from purified components, such as the PURE system developed by Shimizu and colleagues, conveys better parameter control, albeit at higher costs [4].

Cell-free approaches reduce complexity, facilitate modular part insulation [5], and do not require cell viability considerations [6]. This makes them attractive alternatives to whole-cell protein production approaches. For example, Kanter et al. used an *E. coli* CFPS system to synthesize a tumor-specific single-chain antibody fragment of immunoglobulin for lymphoma vaccines at levels that would be toxic to living cells [7]. Goshima et al. demonstrated the potential of CFPS for generating recombinant protein libraries, using a wheat-germ cell-free expression system to manufacture 13,364 human proteins [8]. Thoare et al. found that CFPS of monoclonal antibodies results

in more than twice the number of MAbs produced using Chinese Hamster Ovary cells, with additional advantages when generating 'difficult-to-express' therapeutic targets like membrane proteins [9].

1.2 Modelling CFPS

CFPS allows for defined reaction setups, making it possible for us to simulate and model cell-free systems [10] and allowing us to improve our ability to predict and engineer system behaviour [11]. These models also allow us to characterize kinetic parameters and understand molecular interactions. Using this information, the efficiency of CFPS can ultimately be improved, leading to lower costs, increased resource efficiency, and higher protein yields [12].

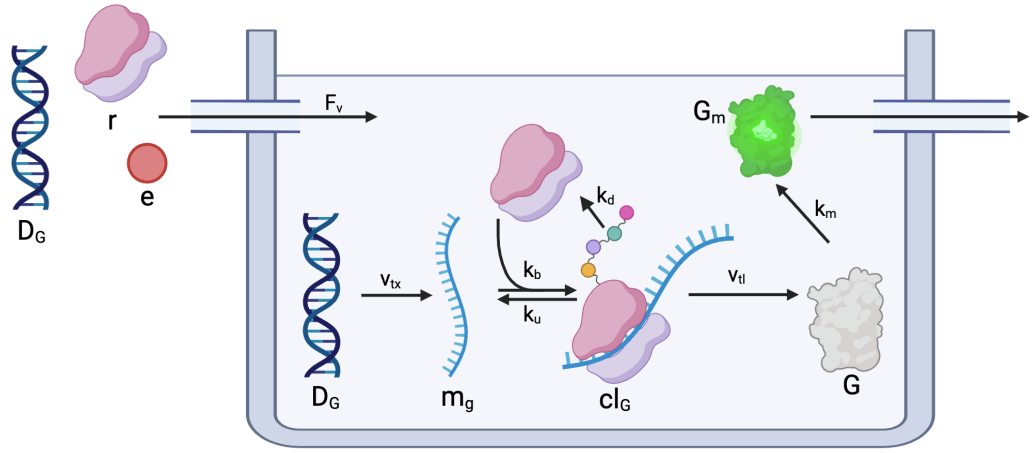


Figure 1.1: Schematic showing the core TX-TL translational machinery involved in a GFP-producing CFPS system, with key species labelled in Table 1.1. Rate of transcription shown as v_{tx} and rate of translation v_{tl}, with dilution rate (F_v), GFP maturation rate (k_m), also shown. Binding and unbinding affinities of the TL elongation complex represented by k_b and k_u respectively, with k_d the degradation rate of the mRNA. Created by the author in BioRender using information gathered from Carlson [2] and Gregorio [3].

Commonly these models involve ordinary differential equations (ODEs) that describe TX-TL and protein degradation [2]. Smaller models focus predominantly on macromolecular components like mRNA, DNA and proteins, often using Green Fluorescent Protein (GFP) as an experimental readout (see Figure 1.1) [10]. For example, Stogbauer et al. studied a cell-free expression system consisting of reconstituted components, developing a coarse-grained model of four equations that describe mRNA transcription and GFP production [13]. In contrast, fine-grained models

investigate a particular process in more detail to more accurately represent the system's substrate limitations *in silico*. For example, a hybrid model created by Arnold et al. consisted of more than 500 equations, providing a comprehensive framework for sequence-related effects in mRNA synthesis, degradation and ribosomal translation [14].

1.3 A simple cell-free model

The rate of change of key TX-TL species (see Table 1.1) can be mathematically represented by differential equations that show the factors affecting their concentrations over time [15]. The following simple ODE system of seven equations has been formulated in previous research by the group, with the key parameters explained in Table 1.2:

$$\frac{de}{dt} = -(v_{\text{tx}} \cdot q_x \cdot n_g + v_{\text{tl}} \cdot q_l \cdot n_G) + e_{\text{in}} - \mu \cdot e, \tag{1.1}$$

$$\frac{dD}{dt} = D_{\text{in}} - \mu \cdot D, \tag{1.2}$$

$$\frac{dm}{dt} = v_{\text{tx}} - k_b \cdot R \cdot m + k_u \cdot c_l + v_{\text{tl}} - k_d \cdot m - \mu \cdot m, \tag{1.3}$$

$$\frac{dcl}{dt} = k_b \cdot R \cdot m - k_u \cdot c_l - v_{\text{tl}} - k_d \cdot c_l - \mu \cdot c_l, \tag{1.4}$$

$$\frac{dG}{dt} = v_{\text{tl}} - k_m \cdot G - \mu \cdot G, \tag{1.5}$$

$$\frac{dG_m}{dt} = k_m \cdot G - \mu \cdot G_m, \tag{1.6}$$

$$\frac{dR}{dt} = -k_b \cdot R \cdot m + k_u \cdot c_l + v_{\text{tl}} + k_d \cdot c_l + R_{\text{in}} - \mu \cdot R. \tag{1.7}$$

Equation 1.1, for example, represents the rate of change of energy in the system. It considers the energy consumption by transcription for all the nucleotides in GFP's mRNA ($v_{\text{tx}} \cdot q_x \cdot n_g$) and the energy consumption by translation for all the amino acids in GFP ($v_{\text{tl}} \cdot q_l \cdot n_G$). It also considers the energy diluted in and out of the system (e_{in} and $\mu \cdot e$).

The rate of transcription (v_{tx}) is given by the DNA concentration multiplied by the transcription rate per nucleotide divided by the number of nucleotides in GFP, multiplied by a Michaelis-Menten term that models the non-linear, saturable effect of energy availability on transcription:

$$v_{tx} = D\frac{k_{tx}}{n_g} \cdot \frac{e}{K_{tx} + e}. \tag{1.8}$$

Similarly, the rate of translation v_{tl} can be represented by

$$v_{tl} = cl\frac{k_{tl}}{n_G} \cdot \frac{e}{K_{tl} + e}. \tag{1.9}$$

Species	Meaning	Default concentration (µM)
e	Energy supply in NTP	33600.0
D	DNA coding for GFP's mRNA transcript	0.005
m	mRNA for GFP	-
cl	TL elongation complex (ribosome bound to mRNA)	-
G	GFP	-
G_m	Fluorescent GFP	-
R	Ribosome	1.51

Table 1.1: The key species involved in a simple cell-free system, with default concentrations used in both the microchemostat experiment and simulations included.

Parameter	Meaning	Value	Unit	Reference
n_g	Number of nucleotides in m	833	nt	§
n_G	Number of amino acids in G	236	aa	§
q_x	Cost of transcribing 1 µM of nucleotides	2	µM	[16]
q_l	Cost of translating 1 µM of amino acids	4	µM	[17]
k_{tx}	Transcription rate	5820	nt/min	[18]
k_b	mRNA-Ribosome binding rate	1000	1/µMmin	[16]
k_u	mRNA-Ribosome unbinding rate	1	1/min	Arbitrary
k_{tl}	Translation rate per ribosome	105	aa/min	*
k_d	mRNA degradation rate	0.038	1/min	*
k_m	Maturation rate of deGFP	0.084	1/min	[19]
K_{tx}	Michaelis-Menten transcription NTP binding coefficient	54.75	µM	[14]
K_{tl}	Michaelis-Menten translation NTP binding coefficient	100	µM	[14]
F_v	Fraction of the solution displaced per dilution cycle	20	%	Microchemostat
τ	Time between dilution cycles	20	min	Microchemostat
μ	Dilution rate	f_v/τ	1/min	Microchemostat
e_{in}	Energy influx	$e_0 * \mu$	µM/min	Microchemostat
D_{in}	DNA influx	$D_0 * \mu$	µM/min	Microchemostat
R_{in}	Ribosome influx	$R_0 * \mu$	µM/min	Microchemostat

Table 1.2: Values for various parameters in the cell-free system, calculated by Christoph Wagner. References marked * are averages over cell-free literature values. References marked § can be found in the Introduction folder of the Supplementary Code (see Supplementary Figure A.1).

where k_{tl} is the translation rate per ribosome-mRNA elongation complex (cl), n_G is the number of amino acids per GFP molecule, and K_{tl} is a half saturation constant for translation, representing the energy level at which translation proceeds at half its maximum rate [20].

These equations were derived from a microchemostat system, created prior to the author's involvement and based on the microfluidic reactor used by Lavickova et al. [21], itself inspired by the nanoliter-scale reactors introduced by Niederholtmeyer et al. [22]. By inputting the DNA that codes for GFP (D), ribosomes (R) and energy (e), along with amino acids and tRNA in excess, a quantitative GFP output was generated from an *E. coli* cell extract. The readout from this experiment is shown in Figure 1.2.

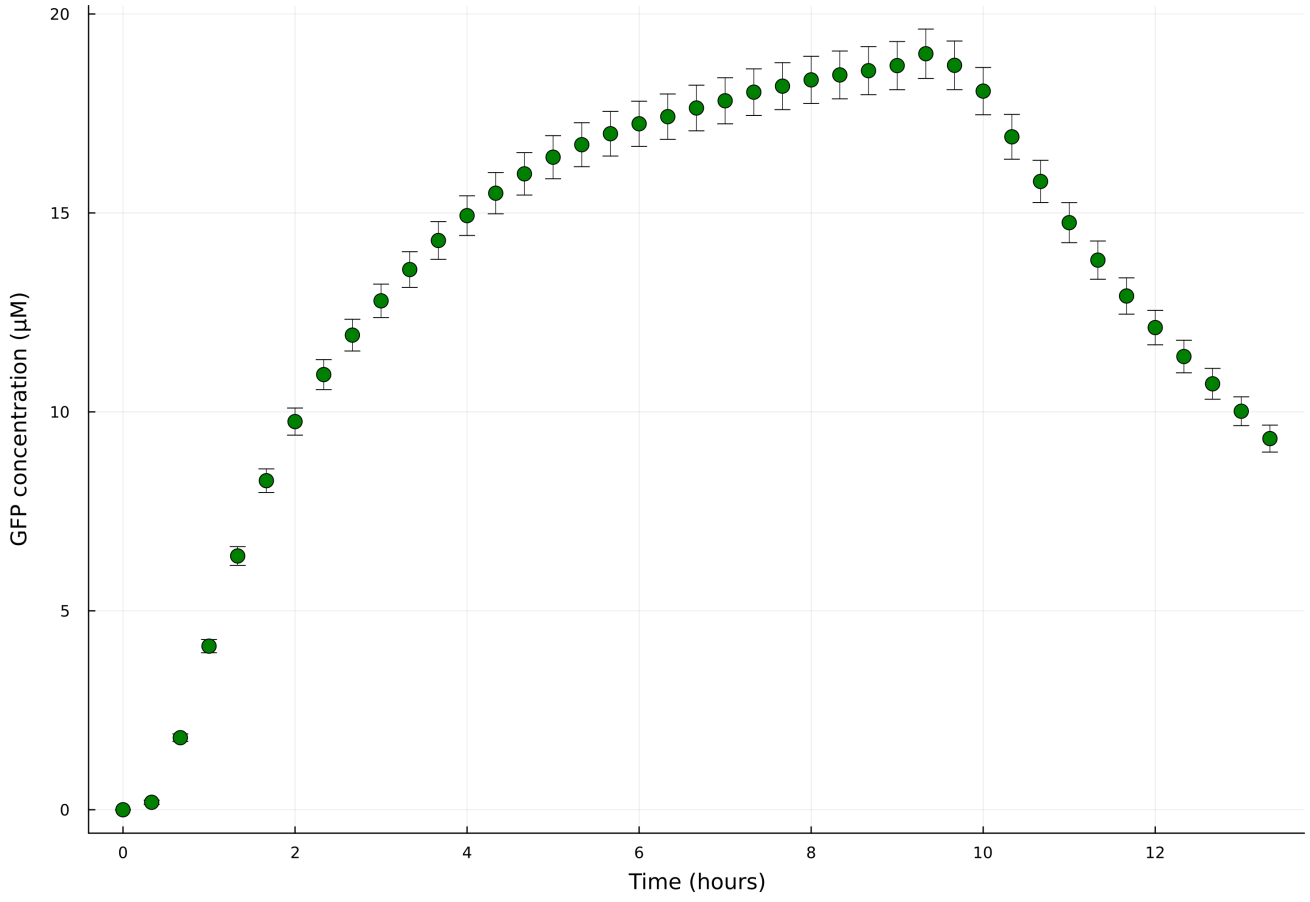


Figure 1.2: Graph showing the GFP concentration over time, produced through TX-TL in a cell-free system. Performed in a microchemostat, with data obtained by Christoph Wagner. Every 20 minutes, 20% of the microchemostat solution was diluted out and replaced with an equal volume of *E. coli* extract made up of energy, DNA and ribosomes. A washout phase commenced at 9 hours, after which no more DNA was added to the system during each dilution.

1.4 Solving the ODE system

For simple ODEs, an algebraic approach would be suitable to solve the equations and calculate the species concentrations over time. However, for a system as complex and non-linear as the cell-free ODE model, it becomes impractical to derive a closed-form, algebraic solution. For practical purposes, a numeric approximation of the solution can be computed using algorithms [23]. For this project, the majority of code was written in Julia, a dynamic programming language for computational science [24], using primarily Runge-Kutta methods found in Julia's `OrdinaryDiffEqFIRK` package [25]. Code for all of the simulations run in this report can be found in a GitHub repository (see Supp. Figure A.1).

Runge-Kutta methods evaluate the derivative at several intermediate points within each step to estimate the next value. Euler's method is the simplest form of a Runge-Kutta method and estimates the solution by using the derivative at the current point and a linear prediction for the next point (see Figure 1.3). Such numeric approximation methods can produce errors, which refer to the difference between the numeric and exact analytical solution.

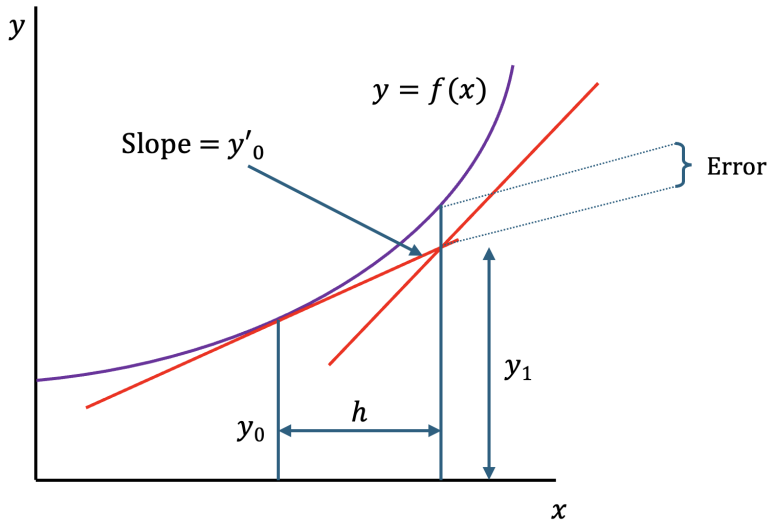


Figure 1.3: Diagram showing how Euler's forward method (also called explicit Euler's method) can be used to create approximate solutions (red) for an equation $f(x)$ (purple). At each time step, the algorithm computes the slope of $f(x)$ and then moves forward by step h to find the next value of y. Each step uses a linear approximation, assuming the slope doesn't change within h, resulting in the accumulation of small errors, especially over large step sizes and rapidly changing or stiff systems.

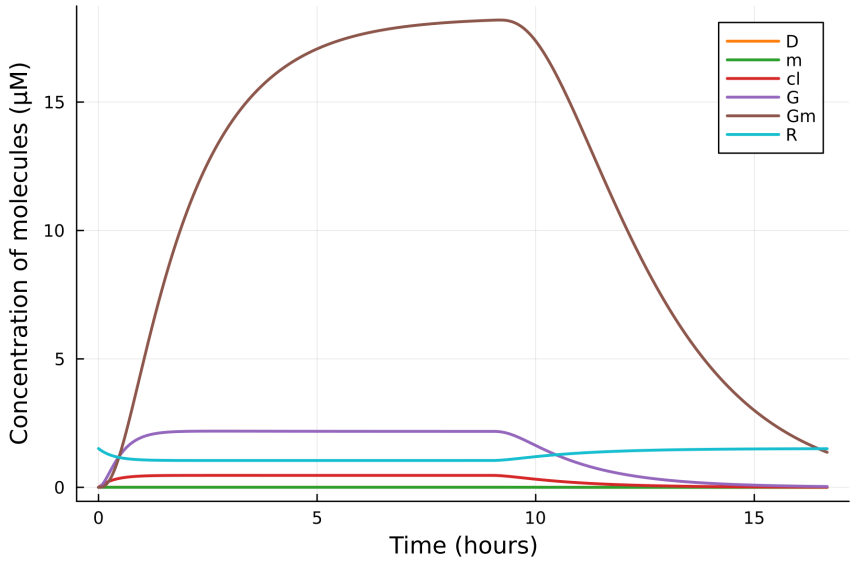
ODE solvers can also be explicit or implicit, depending on whether the solution depends only

on known values or also on the value of the unknown point, respectively. For stiff ODE systems where there are widely varying timescales, explicit solvers (such as Runge-Kutta 4th order) must use extremely small time steps to maintain stability, making them computationally inefficient. Instead, implicit solvers (like `RadauIIA5`, a high-order stiff solver in Julia) can allow for larger time steps while maintaining stability, automatically adapting step sizes to efficiently handle varying dynamics [26]. For the cell-free ODE system, due to a wide disparity between energy concentration and mRNA concentration (which reaches values as small as 0.0002 µM), solvers capable of handling stiff systems are preferred [27]. During the course of the project, an update to the open-source `OrdinaryDiffEq` package introduced numerical issues when using the `RadauIIA5()` algorithm. To address this, `RadauIIA9()` was used for the majority of simulations, offering improved stability and accuracy for the highly stiff nature of the system and underscoring the importance of solver selection and numerical robustness in ODE modelling. The ODE system can be seen in Figure 1.4a, solved using this algorithm.

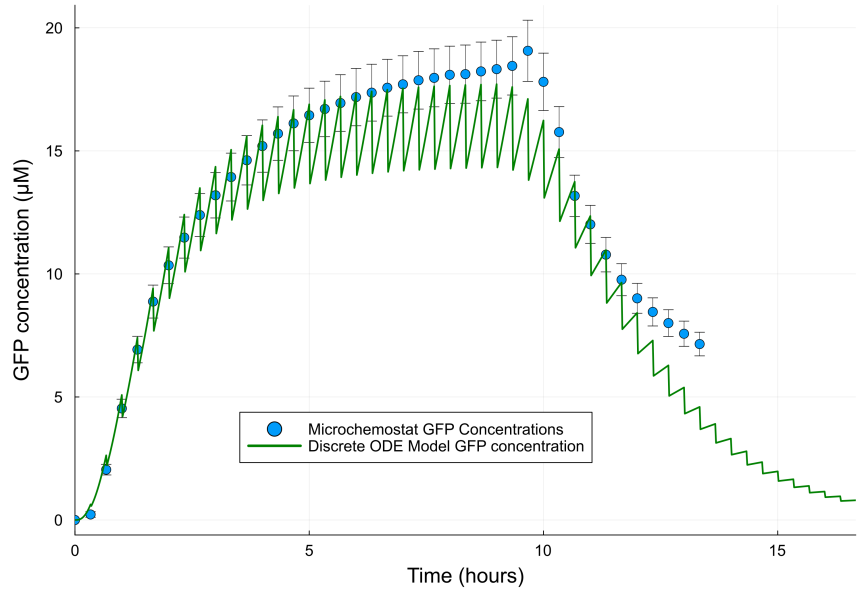
Whereas the model shown in Figure 1.4a is continuous and uses a compound value µ that represents both the dilution fraction and the interval between dilutions, in reality new extract is manually diluted into the microchemostat at discrete intervals of τ minutes. To correctly represent this in the simulation, we use Julia's Callback functions to modify the concentrations of various species as the simulation progresses, to create a discrete version of the model [28]. An overlay of this discrete model over the microchemostat data can be seen in Figure 1.4b. For future reference, it is important to note that because 20% of the microchemostat solution is replaced each dilution cycle, the energy and DNA supplied for each dilution are equal to 20% of their starting concentrations. Hence, when we write that optimisation algorithms converge on solutions by changing e_0 and D_0, this affects the starting concentrations of DNA and energy as well as the concentrations of these species that is supplied in each dilution.

1.5 Aims and objectives

This project aims to apply the optimisation algorithms described below to this simple cell-free ODE system. It will explore the most common optimisation algorithms and how they can be used to optimise various aspects of biological processes, evaluating their relative strengths and weaknesses, as well as their applicability to different systems. It will begin by briefly applying



(a) Continuous ODE plot.



(b) Discrete ODE plot, overlaid with the microchemostat data.

Figure 1.4: Figure showing how ODE models can be constructed to accurately represent experimental data. Both models are solved using Julia's `RadauIIA9()` algorithm, with a dilution interval of $\tau = 20$ minutes. **(a)** shows the cell-free model described in the Equations 1.1-1.7, with parameters designed to closely represent the microchemostat experimental system shown in Figure 1.2. Energy (not shown) is used to transcribe DNA into mRNA (at extremely low concentrations and hence difficult to see on the graph) using free nucleotides. This mRNA binds to ribosomes (blue), forming a translation complex (red) which translates free amino acids into GFP (purple). This GFP matures over time to become fluorescent GFP (brown). **(b)** shows the experimental microchemostat data recorded by Christoph Wagner, overlaid with the discrete ODE model that uses `PeriodicCallback()` to implement dilutions of 20% of the solution volume every 20 minutes.

these approaches to a simple parametric equation, before extending them on to a more complex biological system of two ODEs, and finally the seven-equation cell-free system itself.

By changing the dilution interval, DNA and energy concentrations of the cell-free model, we aim to optimise the theoretical total GFP concentration that can be produced over time, as well as the peak GFP concentration that can be achieved. We also aim to calculate the optimal conditions required to reach steady-state as fast as possible, and to maximise the GFP concentration in a steady state period. The optimisation results included here will fit into a larger, iterative design-learn-build-test pipeline; further progress beyond this project will involve testing the findings for accuracy by comparison with microchemostat experiments. Ultimately, an improved understanding of how CFPS can be optimised will contribute to making this emerging technology more resource-efficient and inexpensive.

Methods

2.1 Optimisation in a cell-free context

2.1.1 Optimisation problems

Mathematical optimisation methods can be applied to continuous or discrete ODE systems (see Table 2.1), to optimise an 'objective function' $f(x)$ by changing the decision variables [29]. Most optimisation algorithms aim to find the minimum value of $f(x)$, as all maximisation problems can be easily converted to minimisation problems because

$$\max_{x \in \mathbb{R}^n} f(x) \quad \text{is equivalent to} \quad \min_{x \in \mathbb{R}^n} -f(x). \tag{2.1}$$

When calculating the solution of an optimisation problem, it is important to consider whether the solution is a global or local minimum, as certain algorithms struggle to locate global solutions [30]. Furthermore, some optimisation problems require constraints to ensure the solutions remain biologically or physically feasible.

Discrete Optimisation	**Continuous Optimisation**
Solutions are chosen from a *countable* set	Solutions are chosen from a *continuous* set
e.g. Selecting genetic constructs (e.g. promoters or plasmids) and identifying which combination gives the greatest expression.	e.g. Tuning DNA concentration in a cell-free system to maximise protein yield.

Table 2.1: Examples of discrete and continuous optimisation problems [31].

2.1.2 Optimising the cell-free system

Various optimisation problems are of interest to cell-free systems. Every dilution cycle, the amount of energy and DNA supplied to the system can be varied along with the interval between dilutions. It should be noted that although these values are parameters in a biotechnology context, when being optimised, they are the decision variables that can be changed to optimise a solution.

Optimising total GFP yield

Given that an important goal of CFPS research is to improve yield and reduce costs, we aim to maximise the total GFP production over time:

$$\max_{e_0, D_0, \tau} \quad \int_0^T G_m(t)\, dt, \tag{2.2}$$

where e_0 and D_0 are respectively the initial energy and DNA concentrations, a fraction of which is input into the chemostat at every τ dilution interval, and T is the total runtime of the simulation (an arbitrary threshold of 1000 minutes, unless otherwise specified). Consistent yields over a production cycle are key to bioreactor optimisation and resource efficiency, important for applications involving sustained, long-term protein synthesis (e.g. therapeutic proteins).

Solving this problem involves integrating the area under the mature GFP concentration curve (see Figure 2.1). Integration functions like Julia's `quadgk()` function compute the definite integral of continuous functions and do not consider the time points at which the ODE solver computes each solution [32]. We therefore calculate the total amount of GFP produced by a Riemann sum approximation of the integral [33]:

$$\text{Total GFP Production} \approx \sum_{i=1}^{N} v_{\text{tl}}(t_i) \cdot \Delta t_i, \tag{2.3}$$

where

$$\Delta t_i = t_i - t_{i-1} \tag{2.4}$$

is the time interval between two successive solver outputs, and $v_{tl}(t_i)$ is the GFP production rate at a discrete solver time, given by the rate of translation from Equation 1.9 [34].

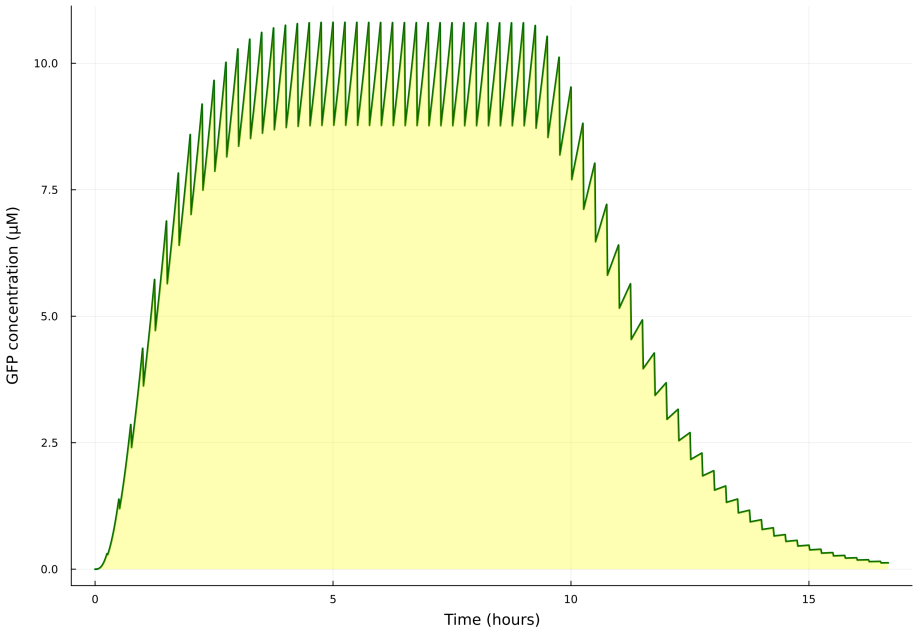


Figure 2.1: Figure showing the total GFP produced over time by the cell-free ODE system. Total GFP output can be calculated by integrating the area (yellow) under the mature GFP concentration curve (green).

Optimising peak GFP output

Optimising the variables to produce the highest peak GFP concentration is another optimisation problem we consider:

$$\max_{e_0, D_0, \tau} \quad G_m(t), \tag{2.5}$$

as seen in Figure 2.2. Producing a high output of protein quickly is important for diagnostic tools where speed and signal-to-noise ratio are essential (such as detecting biomarkers or pathogens), facilitating faster and more accurate signal detection [35]. Additionally, for high-throughput screening applications, CFPS systems that have a high protein peak can generate measurable signals faster, accelerating screening efficiency [36]. This is particularly valuable when testing large libraries of conditions, as strong signals reduce assay time and increase throughput by enabling earlier and more reliable readouts.

Optimising steady-state characteristics

Reducing the time taken for the system to reach a steady state, defined ideally as the time when the rate of change of protein concentration approaches zero:

$$\frac{dG_m}{dt} \approx 0, \tag{2.6}$$

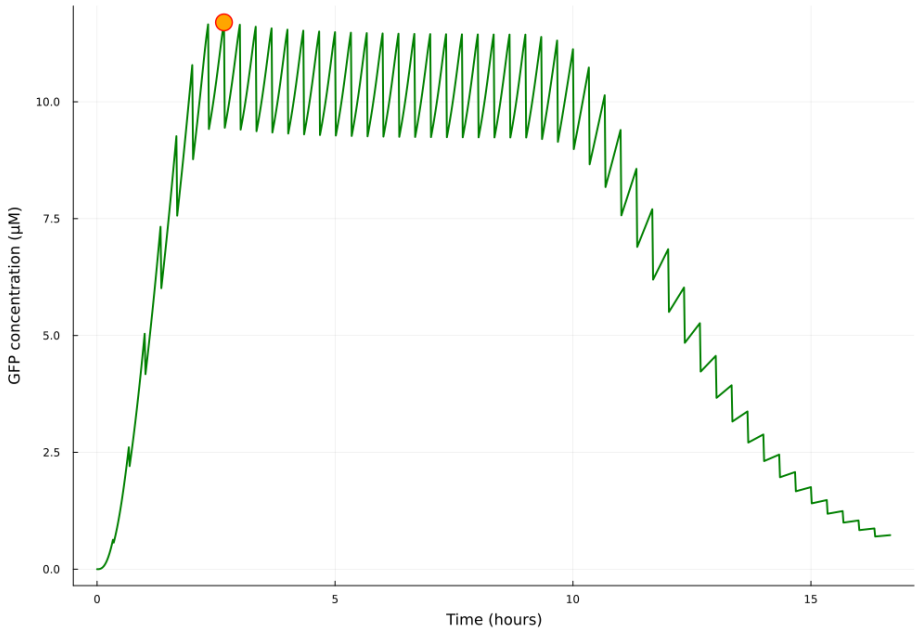


Figure 2.2: Graph showing the mature GFP concentration over time (green) for the cell-free ODE system, with the peak GFP concentration labelled in orange. Peak GFP output can be found using Optim.jl's `maximum()` function to return the highest G_m concentration recorded during the simulation [37].

is important for minimising resource consumption and downtime between bioreactor cycles [22]. Faster system stabilization also improves control and predictability, allowing the system to remain within an optimal range for a longer period of time. Thus, we also aim to explore the steady state period of the system, minimising the time taken to reach steady state:

$$\min_{e_0, D_0, \tau} \quad t_{\text{ss}} \tag{2.7}$$

followed by separately maximising the GFP concentration at steady state:

$$\max_{e_0, D_0, \tau} \quad G_m(t_{\text{ss}}), \tag{2.8}$$

both of which are subject to:

$$\left| \frac{dG_m(t)}{dt} \right| < \varepsilon \quad \forall t \geq t_{\text{ss}} \quad , \tag{2.9}$$

where t_{ss} is the time taken to reach steady-state. Here, ε represents a symbolic threshold used to define ideal steady-state conditions; in practice, steady-state is assessed using species-specific tolerances (ϵ^j) via a scaled finite-difference approach [38]. Since the model includes discrete dilution cycles, the rate of change for each species is approximated by comparing their concentrations (u^j) at pairs of consecutive times (t_i and t_{i+1}) just before a dilution event occurs:

$$\Delta u^j(t_i) = |u^j(t_{i+1}) - u^j(t_i)|. \tag{2.10}$$

To ensure changes are evaluated meaningfully across species with varying magnitudes, the absolute difference of each species is scaled by a tolerance vector ϵ, defined by

$$\epsilon^j(t_i) = \epsilon_{abs} + \epsilon_{rel} \cdot |u^j(t_{i+1})|, \tag{2.11}$$

where $\epsilon_{abs} = 1e^{-3}$ and $\epsilon_{rel} = 1e^{-2}$ [39]. This means that species with high concentrations (e.g. energy) are scaled with the relative constant ϵ_{rel}, and species with low concentrations (e.g. mRNA) are prevented from having a negligible effect with the absolute constant ϵ_{abs}. Once the scaled relative change for each species has been computed,

$$r(t_i) = \max_j \left(\frac{\Delta u^j(t_i)}{\epsilon^j(t_i)} \right), \tag{2.12}$$

if the species with the greatest scaled relative change $r(t_i)$ is less than 1 for a required number of consecutive cycles (C_{req}), the system is deemed as having reached periodic steady state (see Figure 2.3). Unless otherwise specified, we have considered steady state to have been reached if $r(t_i) < 1$ for a C_{req} of 2.

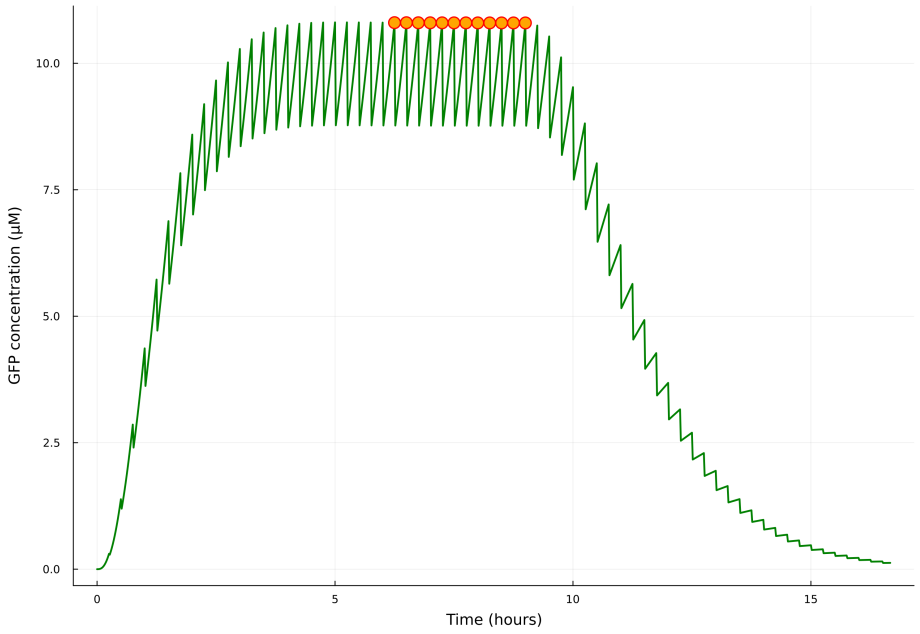


Figure 2.3: GFP concentration over time (G_m, green line) with dilution cycles that have reached steady state shown in orange. Steady state is determined using the scaled finite-difference method described in Equations 2.7–2.12, where the maximum scaled relative change $r(t_i)$ across all species falls below 1 for $C_{\text{req}} = 2$ consecutive cycles, with $\epsilon_{abs} = 1e^{-3}$ and $\epsilon_{rel} = 1e^{-2}$.

2.2 Optimisation algorithms

To solve these problems, we use a variety of optimisation algorithms that begin with an initial guess for the decision variables, and are distinguished from each other by the strategy used to move from one iteration to the next (see Figure 2.4).

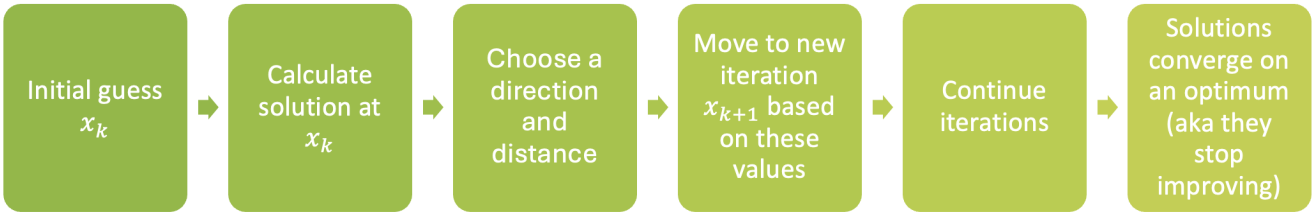


Figure 2.4: General methodology for how optimisation algorithms begin at an initial guess in the problem space and iteratively improve decision variables, producing new objective function solutions that eventually converge as close as possible to a global optimum.

2.2.1 Gradient-based algorithms

Gradient-based optimisation algorithms use information gathered about the derivatives of the objective function to choose an appropriate direction vector p_k and scalar step length α [40].

Gradient descent

The gradient descent approach aims to minimise the objective function by choosing a descent direction in the opposite direction of the gradient ($-\nabla f_k$):

$$x_{k+1} = x_k - \alpha \nabla f(x_k) \tag{2.13}$$

[37]. By following the slope of the surface created by the objective function downhill, an optimum can be reached. Choosing a correct α value for simple algorithms such as this can be challenging; small step lengths lead to painfully slow convergence, while large values can overshoot minima [41].

Newton's method

To help inform the algorithm of suitable step lengths and directions, second derivatives can also be considered [31]. Newton's Method uses a second order Taylor expansion of a smooth function

$f(x)$ around the point x_k:

$$f(x) \approx f(x_k) + f'(x_k)(x - x_k) + \frac{1}{2}f''(x_k)(x - x_k)^2. \tag{2.14}$$

This equation considers a linear approximation $f'(x_k)(x - x_k)$ that describes the slope of the function, and a quadratic term $\frac{1}{2}f''(x_k)(x - x_k)^2$ that describes the curvature [42]. Minimising this quadratic:

$$f'(x_k) + f''(x_k)(x - x_k) = 0, \tag{2.15}$$

and solving for x points towards a stationary point, and is the one-dimensional version of Newton's step:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}. \tag{2.16}$$

In higher dimensions the second-order partial derivatives are contained in the Hessian matrix $H(x_k)$ and partial derivatives of $f(x)$ in the gradient vector $\nabla f(x_k)$:

$$x_{k+1} = x_k - H^{-1}\nabla f(x_k). \tag{2.17}$$

Algorithms such as Optim's `Newton()` have fast rates of local convergence that are typically quadratic [37]. However, sometimes explicit computation of the Hessian matrix can be cumbersome and computationally expensive. Quasi-Newton methods such as Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (the function `LBFGS()` in Optim.jl) construct a model of the objective function and approximate its Hessians, resulting in superlinear convergence [43]. Automatic differentiation (AD) techniques (e.g. Zygote.jl's `AutoZygote()` [44] and SciML.jl's `AutoForwardDiff()` [45]) are often used to automatically calculate first or second derivatives.

Interior Point

Interior point (IP) methods like `IPNewton()` have polynomial runtimes, and analyse the gradient and Hessian to navigate the interior boundary of the feasible region [46]. Each interior-point iteration is expensive to compute, but can make significant progress towards the solution [31].

These algorithms convert constrained optimisation problems into a form where a derivative test can still be applied. They use Lagrangian multipliers (λ) to reformulate the original problem into a Lagrangian function:

$$\mathcal{L}(x, \lambda) \equiv f(x) + \langle \lambda, g(x) \rangle, \tag{2.18}$$

where the notation $\langle \lambda, g(x) \rangle$ denotes the inner product of the Lagrangian multiplier λ and the constraint functions $g(x)$. For the common case of linear constraints, this simplifies to $\lambda^T g(x)$.

IP methods add a barrier function, defined in the interior of the feasible region such that as the solution approaches the boundary of the feasible region, the barrier term tends to infinity and prevents the iterates from violating the constraints [47]. The barrier parameter μ controls the distance from the boundary of the feasible region, and results in a modified Lagrangian that typically includes logarithmic barrier terms:

$$\mathcal{L}(x, s, \lambda; \mu) = f(x) + \lambda^T (g(x) + s) - \mu \sum_i \log(s_i), \tag{2.19}$$

where s represents slack variables that facilitate the conversion of the constrained optimisation problem [48]. As the algorithm progresses, μ gradually decreases toward zero, allowing the solution to approach the boundary of the feasible region while following a trajectory called the central path.

2.2.2 Gradient-free algorithms

For functions where derivatives are difficult to compute, algorithms have been developed that do not attempt to approximate the gradient [31].

Nelder-Mead

The Nelder-Mead method is a direct search approach that uses the concept of a simplex (a polytope of $n + 1$ vertices in n dimensions) to solve unconstrained optimisation problems [49]. It begins with a set of points $x_0, ..., x_n \in \mathbb{R}^n$ that are considered the vertices of the simplex S [50]. Each iteration of the Nelder-Mead algorithm attempts to remove the vertex with the worst function value and replace it with another point by reflecting, contracting, or expanding the simplex. It performs this transformation along the line joining the centroid of the remaining vertices with the excluded vertex. If no better point exists along this line, only the vertex with the best function value is retained [31].

Evolutionary optimisers

Evolutionary algorithms (EAs) are inspired by Darwinian evolutionary systems, and solve optimisation problems by iteratively evolving a population of candidate solutions over several generations [51]. Following the creation of an initial candidate population, these algorithms follow a basic cycle of: GPM (genotype-phenotype mapping), evaluation, fitness assignment, selection, and reproduction.

A population of n individuals is randomly created, each of which represents a point in the search space [52]. During GPM, these points are then translated into phenotypes which represent the actual solution in the problem space. The values of the objective function are then evaluated, and fitness values are assigned to each individual; high-fitness candidates enter the mating pool with a higher probability. During the reproduction phase, offspring are derived from the genotypes of the selected points through either mutation (modification of a single genotype) or crossover (combination of two genotypes to form a new one). The rate at which offspring take over the population is determined by the population handling strategy being used.

Differential evolution (DE) approaches use stochastic direct searches that can start at isolated, stochastically chosen locations [25]. Their mutation strategy is unique to other algorithms: they

create a mutant vector by combining two randomly selected population vectors to a third vector. This mutant vector v_i is then combined with the current solution x_i to form a trial vector that is measured for fitness, replacing x_i in the next generation if fitness is improved. BlackBoxOptim.jl's `BBO()` algorithm is used extensively here and internally implements adaptive variants of DE strategies, adapting mutation parameters during the search and using a binomial crossover strategy [53], making it well-suited for noisy, non-differentiable, and multimodal objective functions [54].

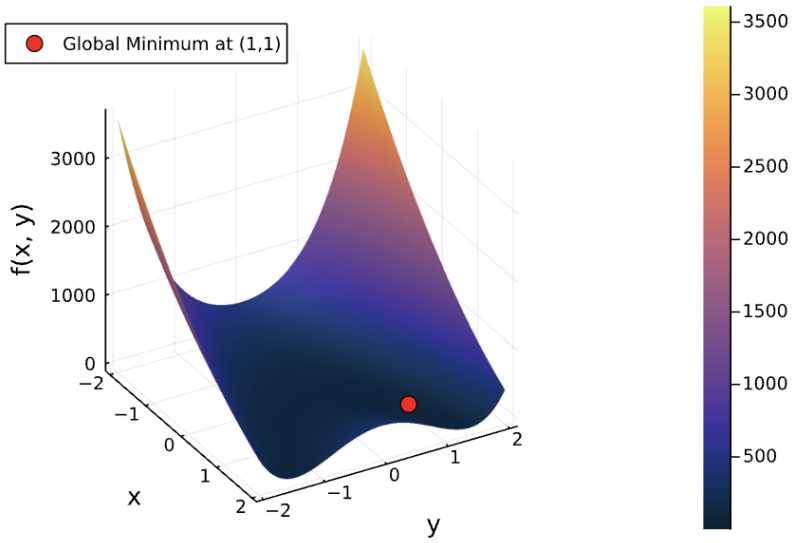
2.2.3 Performance test on the Rosenbrock function

A common non-convex function used as a performance test problem for optimisation algorithms is the Rosenbrock Equation, defined by:

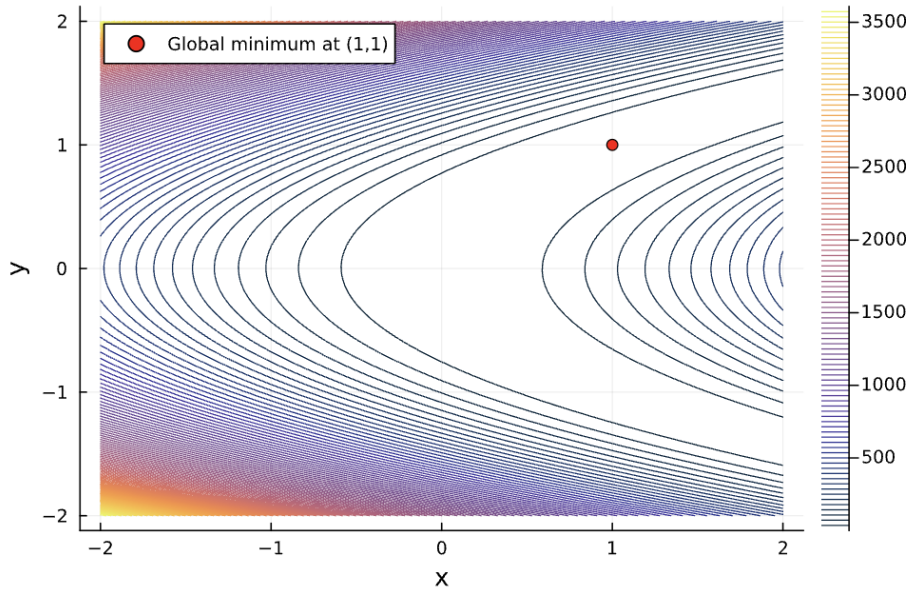
$$f(x, y) = (a - x)^2 + b(y - x^2)^2 \quad . \tag{2.20}$$

It has a global minimum at $(1, 1)$ where $(x, y) = (a, a^2)$ and $f(x, y) = 0$. Due to a long, narrow curved valley and steep, orthogonal walls (see Figure 2.5), it is a challenge for optimisation algorithms to locate the minimum without adaptive step sizes or second-order methods. Because of this geometry, standard methods like gradient descent can converge extremely slowly, especially if the initial guess is too far from the valley or if step sizes are too great. In preparation for optimising the cell-free system of ODEs, several optimisation algorithms are applied to the Rosenbrock function (see Table 2.2).

Of the gradient-based methods, L-BFGS converges the fastest due to its consideration of curvature information via Hessian approximations. Newton's method also performs well due to exact second-order updates. Gradient Descent requires an extensive run-time due to its first order nature, while the IP approach incurs the highest computational cost due to its difficulty handling internal constraints. The Differential Evolution optimiser (`BBO()`), good at exploring less-characterised landscapes but slower at fine-tuning in narrow valleys where directional information is important, reaches the maximum number of iterations specified to the algorithm before being able to converge at the global optimum. The results for the Nelder-Mead algorithm illustrate its effectiveness in low-dimensional problems, offering surprisingly fast convergence.



(a) Surface plot of the Rosenbrock function.



(b) Contour plot of the Rosenbrock function.

Figure 2.5: Plots showing the Rosenbrock function, a classic test problem for optimisation algorithms, characterised by a narrow, curved valley leading to a global minimum at (1, 1), where $f(x, y) = 0$. **(a)** shows the equation graphed in three-dimensional space, illustrating steep walls and a flat valley floor that create challenges for gradient-based methods, requiring navigation of a long, curved trajectory to reach the minimum. **(b)** shows a contour plot that emphasises the shape and orientation of the function in two-dimensional space.

Algorithm	**Iterations**	**Time (sec)**	**Comments**
Gradient Descent	54	1.25	Converged reliably; slow due to basic first-order updates.
Newton	29	0.353	Very efficient; uses second-order derivatives for fast convergence.
L-BFGS	27	0.0365	Fast and efficient; uses Hessian approximation for curvature info.
IPNewton	59	3.117852	Slower due to constraint handling overhead.
BBO	10001	0.241967	Hit max iterations; gradient-free global optimiser.
Nelder-Mead	79	0.020059	Reasonable performance; simplex method (no derivatives).

Table 2.2: Performance comparison of several optimisation algorithms applied to the Rosenbrock function, with the number of iterations and time taken to reach the global optimum shown. Initial guesses $x_0, y_0 = 5.0, 5.0$ and parameters $a = 1$, $b = 100$.

Results

Having completed an initial performance test on the Rosenbrock equation, we apply the optimisation algorithms to a simple two-equation predator-prey relationship and progress to optimising each of the cell-free optimisation problems.

3.1 Optimisation of a Lotka-Volterra ODE system

The Lotka-Volterra equations (see Figure 3.1) are a pair of first-order non-linear differential equations used to describe species interactions in biological systems:

$$\frac{dx}{dt} = \alpha x - \beta xy, \tag{3.1}$$

$$\frac{dy}{dt} = -\gamma y + \delta xy, \tag{3.2}$$

where x and y are the prey and predator populations respectively. The parameters indicate the growth rate and decline of each population: α is the birth rate of the prey, β is the predation rate coefficient, δ is the reproduction rate of predators per prey consumed and γ is the death rate of the predators.

3.1.1 Optimisation of initial populations

We begin by maximising peak prey population by simply changing the initial prey population:

$$\max_{x_0} \quad x(t) \quad \text{subject to} \begin{cases} 0.1 \leq x_0 \leq 50.0, \\ y_0 = 1.0, \\ \alpha = 0.5, \beta = 1.0, \gamma = 3.0, \delta = 1.0. \end{cases} \tag{3.3}$$

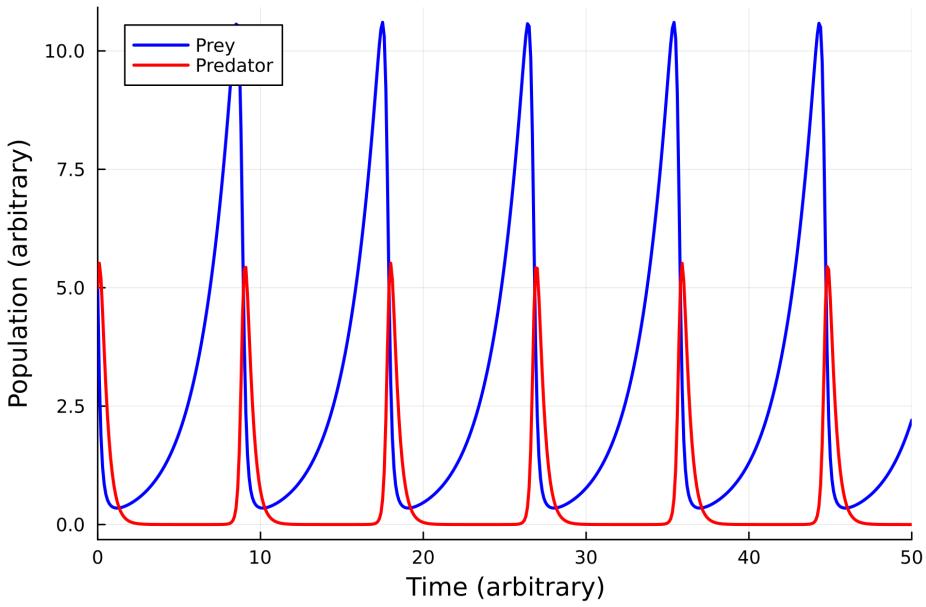


Figure 3.1: Graph showing the changing populations of a prey and predator species over time, as defined by the Lotka-Volterra equations. Initially, the populations of both species begin at 5 ($x_0 = 5.0, y_0 = 5.0$). The prey population immediately declines due to the predation rate ($\beta = 1.0$). The predator population also declines shortly after this, due to the predator death rate ($\gamma = 3.0$) and the fact that fewer prey are contributing to the reproduction rate of predators per prey consumed ($\delta = 1.0$). Once the predator population becomes sufficiently small, predation is minimal and the prey birth rate ($\alpha = 0.5$) allows prey populations to increase again. This cycle repeats in an oscillating pattern.

or maximising peak predator population by changing the initial predator population:

$$\max_{y_0} \quad y(t) \quad x(t) \quad \text{subject to} \begin{cases} x_0 = 1.0, \\ 0.1 \leq y_0 \leq 50.0, \\ \alpha = 0.5, \beta = 1.0, \gamma = 3.0, \delta = 1.0. \end{cases} \tag{3.4}$$

The non-linear dynamics of the Lotka-Volterra system can cause the objective functions to be non-smooth or discontinuous at certain points. Small variations in either x_0 or y_0 result in chaotic, non-linear dynamics, illustrating the system's sensitive dependence on its initial conditions (see interactive simulation in supplementary code). The integration methods used to solve the ODE system can also introduce numerical noise [55]. When x_0 and y_0 increase too much, the ODE solver calculates exponential increases in x_0, with y_0 reaching negative population values, necessitating a domain check function to prevent y from reaching negative values.

Intuitively, optimisation algorithms would be expected to converge on initial population values

as close to the upper bound as possible, with a highest peak at $t = 0$ where $\max_{x_0} = x_0$ and $\max_{y_0} = y_0$ (see Supplementary Figure A.2). These results can be mostly achieved, with initial guess alterations required for gradient-based algorithms (see Table 3.1).

Optimisation Algorithm	**Optimal x_0**	**Peak Prey Population**	**Optimal y_0**	**Peak Predator Population**
BBO	50.0	50.0	50.0	50.0
Nelder-Mead	7.525	7.8402	7.525	9.4298
LBFGS	49.9999	49.9999	4.9856	6.4950
IPNewton	50.0	50.0	4.9898	6.4769

Table 3.1: Optimisation of x_0 and y_0 to maximise peak x and y populations. Initial guesses at x_0=1.0 and y_0=1.0 respectively. Nelder-Mead requires a multidimensional problem space to generate enough vertices for its simplex, and hence struggles to converge on correct solutions. BBO, L-BFGS and interior-point algorithms successfully converge on optimum x_0 values when optimising prey population. Only the BBO algorithm achieves an optimum y_0 for the predator population optimisation, because a rugged gradient landscape below $y_0 = 16$ means gradient-based approaches failed to escape local minima (see Supp. Figure A.3). Beyond an initial guess for y_0 of 20.0, gradient-based algorithms can navigate the flat landscape and correctly converge on y_0 values.

3.1.2 Optimisation of birth rate

Next, we optimise the parameters α and δ to maximise the peak prey and predator populations respectively, starting with prey birth rate α:

$$\max_{\alpha} \quad x(t), \begin{cases} 0.0 \leq \alpha \leq 3.0, \\ x_0 = 1.0, y_0 = 1.0, \\ \beta = 1.0, \gamma = 3.0, \delta = 1.0. \end{cases} \tag{3.5}$$

and then predator reproduction rate per prey consumed δ:

$$\max_{\delta} \quad y(t). \begin{cases} 0.0 \leq \delta \leq 3.0, \\ x_0 = 1.0, y_0 = 1.0, \\ \alpha = 0.5, \beta = 1.0, \gamma = 3.0. \end{cases} \tag{3.6}$$

The general trend for the effects of increasing α and δ on peak x and y values can be seen in Supp. Figure A.4. A higher prey birth rate (α) correlates to high peak population values, while lower δ values correspond to higher peak predator populations (with lower predation, prey population is allowed to reach higher peaks which can then temporarily support a large predator population).

Optimisation Algorithm	Optimal α (Prey)	Peak Prey Population	Optimal δ (Predator)	Peak Predator Population
BBO	2.9986	2.1997	0.0512	15.3704
NelderMead	1.525	1.3086	0.0513	14.2948
LBFGS	2.9999	2.1935	1.0338	2.7291
IPNewton	1.0003	1.0876	1.0000	2.7237

Table 3.2: Optimisation by different algorithms to maximise peak x and y populations by varying α and δ, with initial guesses $\alpha = 1.0$ and $\delta = 1.0$. Once more, Nelder-Mead fails due to a one-dimensional problem space. Interior point fails for both problems due to difficulty calculating second-order derivatives. The BBO algorithm successfully converges on optimum values for both α and δ, calculating slightly more accurate values than those found by feeding the vector from the optimisation landscape in Supp. Figure A.4 into Julia's `maximum()` function (due to the resolution of the landscape). LBFGS remains stuck in local minima near the initial guess for δ due to the rugged gradient landscape (see Supp. Figure A.5), but correctly converges on an optimum solution for x_0 due to the flatter landscape.

3.1.3 Optimisation of oscillation period

We attempt to optimise α, β, γ and δ to achieve an oscillation period as close to a target value as possible:

$$\min_{\alpha,\beta,\gamma,\delta} |T_{\text{computed}}(\alpha,\beta,\gamma,\delta) - T_{\text{target}}| \quad \text{subject to} \begin{cases} 0.01 \leq \alpha, \beta, \gamma, \delta \leq 5.0 \\ x_0 = 5.0, \quad y_0 = 5.0, \\ T_{\text{target}} = 18. \end{cases} \tag{3.7}$$

where $T_{\text{computed}}(\alpha,\beta,\gamma,\delta)$ is the estimated period of oscillations in the prey population and T_{target} is the desired oscillation period. The period is estimated by detecting the time points T_{peaks} at which the prey population $x(t)$ reaches local maxima, a similar concept to calculating steady state at discrete dilution intervals in the cell-free ODE model (see Equation 2.7). The oscillations are chosen by computing the x values at which

$$\frac{d}{dt}x(t) \approx \frac{x(t_i) - x(t_{i-1})}{t_i - t_{i-1}} \tag{3.8}$$

changes sign, and removing outliers beyond the interquartile range. The results for the optimisation algorithms applied to this problem can be seen in Table 3.3. Even in multidimensional problems, the Nelder-Mead and BBO algorithms accurately converge to optimum decision variables.

Optimisation Method	α	β	γ	δ	**Optimised Period**	**Target Period Difference**
Nelder-Mead	0.4253	1.4802	2.3455	1.4705	18.0	0.0
BBO	2.2153	4.5190	0.7214	1.1623	18.0	0.0
LBFGS	0.5	1.0	3.0	1.0	27.0	9.0
IP Newton	0.5	1.0	3.0	1.0	27.0	9.0

Table 3.3: Table showing the results of different optimisation algorithms changing all four parameter values simultaneously to converge on an oscillation period of 18.0. Initial guesses at 0.5, 1.0, 3.0 and 1.0 for α, β, γ and δ respectively. Given the multidimensional problem space, the Nelder-Mead algorithm performs well, along with the BBO approach. Gradient-based methods fail to calculate derivatives due to the chaotic nature of the system, and hence fail to progress beyond the initial guesses.

3.2 Optimising cell-free protein expression

Having gained an appreciation for the importance of problem dimensionality, gradient calculation accuracy, and the ODE solvers employed, we seek to optimise the cell-free problems described above.

3.2.1 Optimising total GFP production

To maximise the total GFP production over time (see Equation 2.2), we begin by considering the effect of each parameter on the cell-free model individually, before optimising the total GFP by changing all three decision variables simultaneously.

Energy

Considering the effect of energy on total GFP output:

$$\max_{e_0} \quad \int_0^T G_m(t)\,dt \quad \text{subject to} \begin{cases} 30000\ \mu M \leq e_0 \leq 60000\ \mu M, \\ D_0 = 0.005\ \mu\text{M}, \\ \tau = 20\ \text{min}, \end{cases} \tag{3.9}$$

we reason that a higher energy concentration facilitates more TX-TL and increases GFP production. Applying several optimisation algorithms to the problem in Equation 3.9 (see Table 3.4) reveals, as expected, values of e_0 close to the upper limit of the constrained problem. For lower-dimensional problems, graphical representations can help validate the findings of the optimisation algorithms (see Figure 3.2).

DNA

We then consider the effect of DNA concentration:

$$\max_{D_0} \quad \int_0^T G_m(t) \quad \text{subject to} \begin{cases} e_0 = 336000\ \mu M, \\ 0.001\ \mu M \leq D_0 \leq 0.1\ \mu M, \\ \tau = 20\ \text{min}, \end{cases} \tag{3.10}$$

plotting the effect of changing D_0 on total GFP output in Figure 3.3. We reason that the decrease in GFP production beyond $D_0 = 0.025$ µM is due to a higher concentration of DNA redirecting

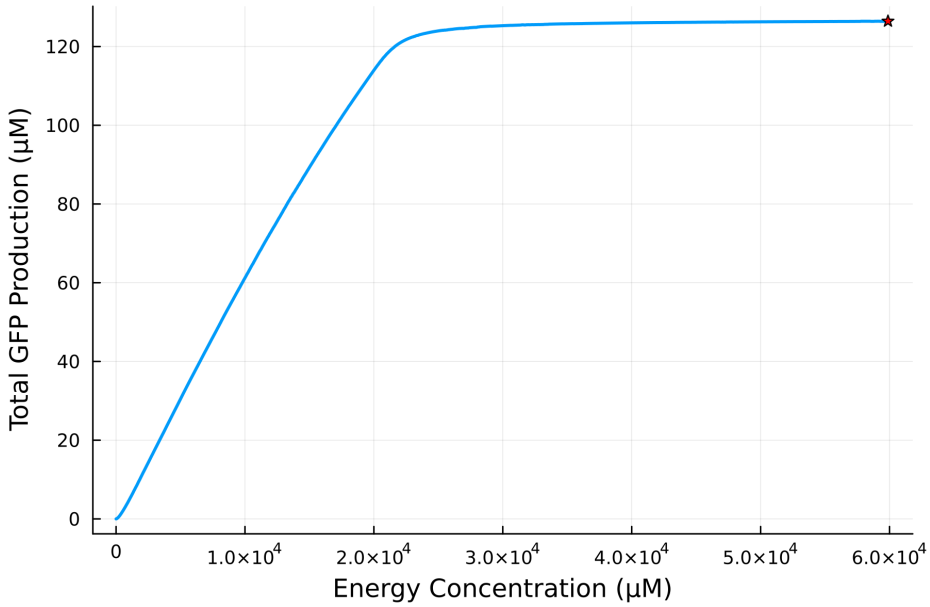


Figure 3.2: Relationship between the energy concentration in NTP supplied to the system and the total GFP produced. Total GFP output initially increases near-linearly with energy, as more TX-TL can occur, before plateauing beyond around 2.5×10^4 µM NTP as other factors (e.g. DNA and ribosomes) become limiting. Star shows maximum GFP output of 126.3579 µM at an optimum e_0 value of 60000 µM, found by supplying the vector of solutions to Julia's `maximum()` function.

Algorithm	**Termination Status**	**Optimum e_0 (µM)**	**Total G_m (µM)**
Nelder-Mead	Success	50400.0250	126.2234
BBO	MaxIters	59999.9999	126.3578
L-BFGS	Success	59999.9999	126.3578
IPNewton	Success	59999.9999	126.3578

Table 3.4: Results for several optimisation algorithms changing the energy concentration between 30000-60000 µM to maximise total GFP output, starting from an initial guess of $e_0 = 33600$ µM. Although Nelder-Mead achieves a high total G_m output, it remains poorly equipped to deal with one-dimensional problems and does not converge on the same global optimum found by the BBO and gradient-based algorithms.

energy away from translation and into transcription. To confirm this, we plot the energy consumed by transcription and translation at various concentrations of DNA (see Figure 3.4), finding that at higher DNA concentrations, more energy is consumed by transcription than by translation, resulting in decreased protein production.

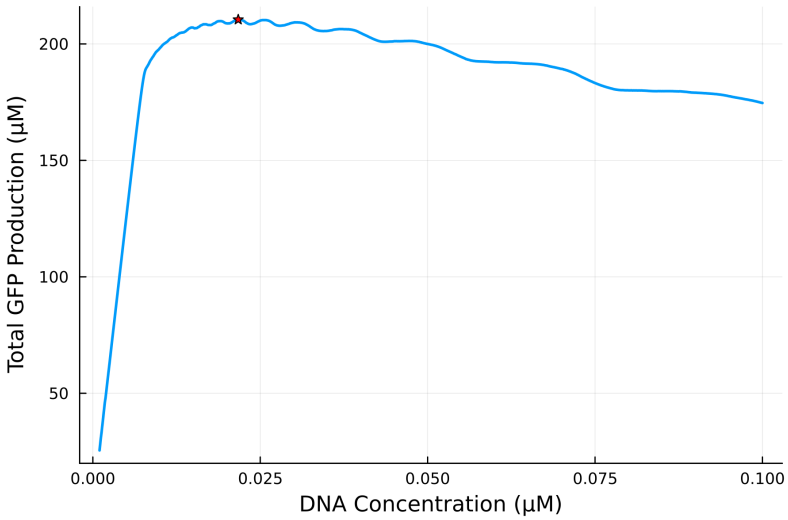


Figure 3.3: Graph showing the effect of increasing DNA concentration on the total GFP produced. Star shows the maximum total GFP output of 210.4484 μM at an optimum D_0 value of 0.02171 μM, found by supplying the vector of solutions to Julia's `maximum()` function. Applying the black-box differential evolution algorithm to Equation 3.10 at a fixed e_0 of 33600 μM calculates a more accurate optimum DNA concentration of 0.02172 μM, resulting in a total GFP output of 210.4578 μM. No explanation has yet been found for the somewhat regular oscillations in the relationship, but this will be pursued in further research.

Prior to a fully combined optimisation simulation, we consider the effect of both DNA and energy on the total GFP output, plotting this as a heat map in Figure 3.5. This reveals that the optimum energy concentration (regardless of DNA) is close to the upper limit of 60000 μM, and the optimum DNA concentration at $e_0 = 60000$ μM is around 0.025 μM. The results of several optimisation algorithms confirm this (see Table 3.5).

Dilution interval

The next parameter that we optimise is the interval between dilution cycles in the microchemostat:

$$\max_{\tau} \quad \int_0^T G_m(t) \quad \text{subject to} \begin{cases} e_0 = 33600\ \mu M, \\ D_0 = 0.005\ \mu M, \\ 1\text{ min} \leq \tau \leq 120\text{ min}. \end{cases} \tag{3.11}$$

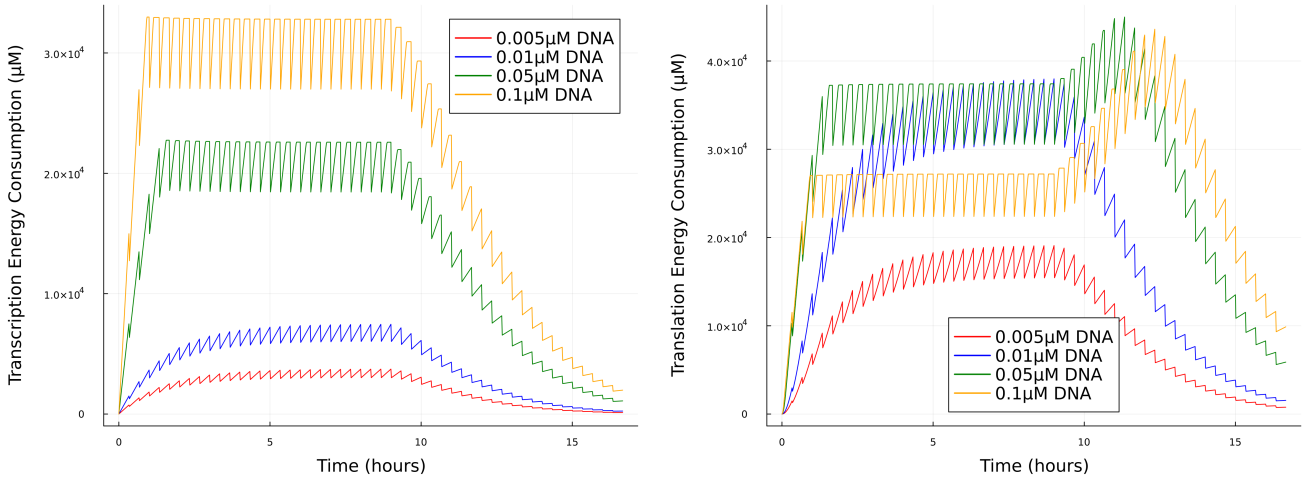


Figure 3.4: Graph showing the energy consumed by transcription and translation at different DNA concentrations. At low DNA concentrations, little energy is used in either transcription or translation as minimal DNA is present to be transcribed into mRNA. High concentrations of DNA (D_0=0.1 μM, shown in yellow), lead to high amounts of energy being consumed by transcription, while reducing the amount of energy contributing to translation. Thus, a high DNA concentration is not necessarily beneficial for total GFP production. The delayed peaks in translation energy consumption are due to the washout phase removing DNA, meaning all energy is briefly directed through translation, before the remaining mRNA gets diluted out of the system too.

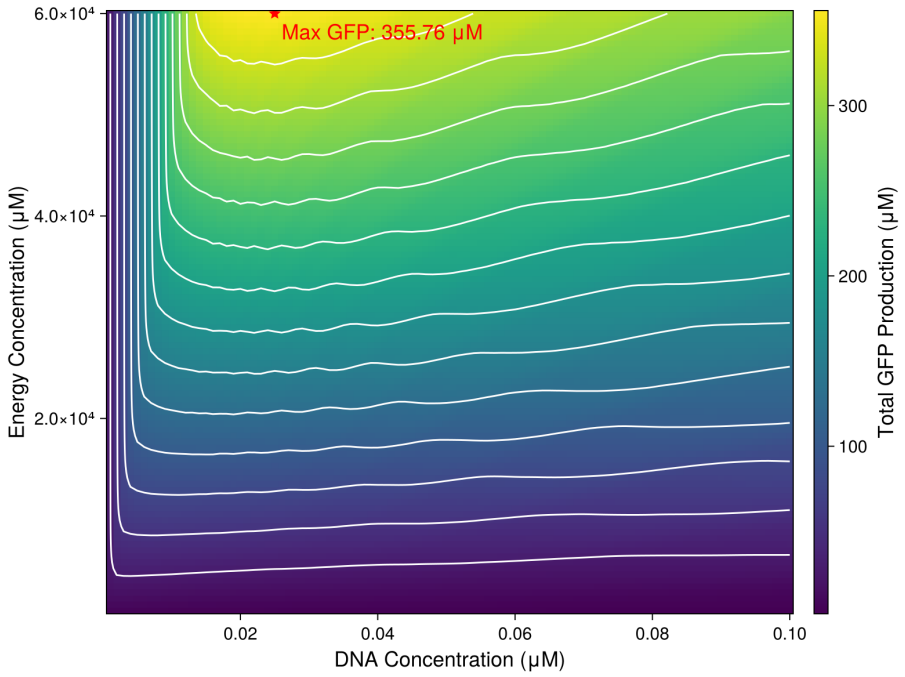


Figure 3.5: Heat map showing the relationship between initial energy and DNA concentrations on total GFP at a dilution interval of 20 minutes. Maximum shown in red at e_0 = 60000 μM and D_0 = 0.02538 μM. Nelder-Mead found a higher value of 418 μM (see Table 3.5), albeit by extending e_0 beyond the bounds specified for this heat map.

Algorithm	Termination Status	Optimal e_0 (µM)	Optimal D_0(µM)	Total GFP (µM)
BBO	MaxIters	59999.995	0.02533	355.912
Nelder-Mead	Success	75000.803	0.02	418.379
LBFGS	Success	59999.640	0.02118	355.637
IPNewton	Success	59999.446	0.02534	355.907

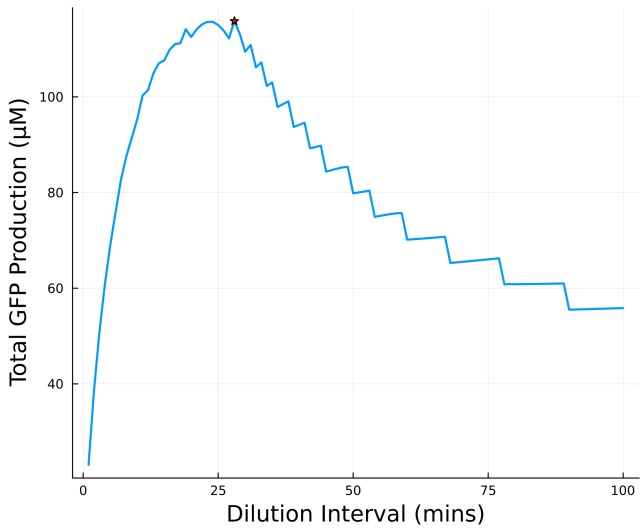
Table 3.5: Results of several optimisation algorithms attempting to maximise total GFP concentration by changing energy and DNA concentration. Initial guesses at e_0=50000 µM and D_0 = 0.02 µM. Nelder-Mead converges on a solution using an energy concentration outside of the specified bounds of the problem (33600 µM $\leq e \leq$ 60000 µM), illustrating its difficulty at handling constraints [50]. BBO, LBFGS and IPNewtons algorithms successfully converge at total GFP values of 355 µM, confirmed by Figure 3.5.

We plot the effect of τ on the total GFP output, and GFP concentration over time at several τ values in Figure 3.6. Applying optimisation algorithms to the problem (see Table 3.6) confirms an optimum dilution interval of around 30 minutes.

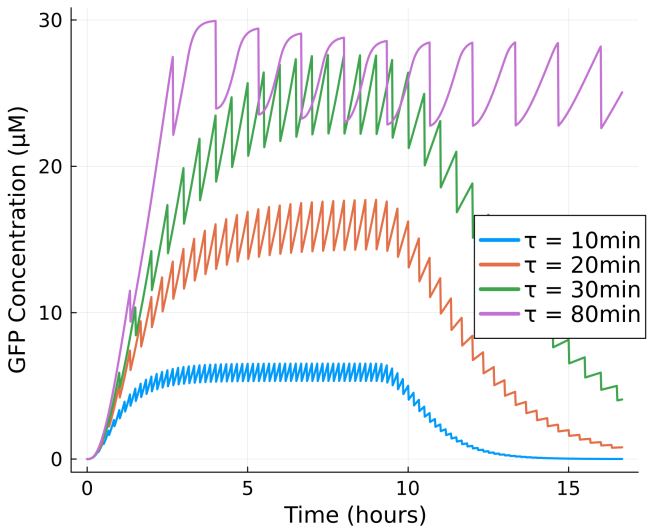
Method	Termination Status	Optimum Dilution Interval τ (min)	Total GFP (µM)
BBO	MaxIters	29.9286	144.3591
Nelder-Mead	Success	30.025	138.9395
LBFGS	Success	20.0	125.6828
IPNewton	Fail	-	-

Table 3.6: Results from various algorithms changing the dilution interval to maximise total GFP output. Initial guess of τ = 20.0 minutes. LBFGS and IPNewton algorithms struggle to correctly calculate derivatives of the model due to the discrete nature of the callback function implementation. Nelder-Mead fails to correctly converge due to the one-dimensional optimisation problem. The BBO algorithm achieves an accurate τ value, validated by Figure 3.6a.

To better visualise how the dilution interval affects other species in the cell-free system, we then plot energy concentration at different τ values (see Figure 3.7). We observe that a τ value of 30 minutes (close to the optimum) is sufficient to fully use the energy in the system, without the system spending prolonged periods without any energy and waiting for it to be added back in at the next dilution cycle.



(a) Effect of changing τ on total GFP output.



(b) GFP concentration over time at several τ values.

Figure 3.6: Figure showing how τ affects GFP production. **(a)** shows the relationship between dilution interval and the total GFP output. Peak τ value found at around 30 minutes. Fluctuations beyond this value result from the number of DNA-supplying dilution cycles (dictated by τ) that can occur before the washout phase removes DNA from the system (see the supplementary code for an interactive simulation of dilution number at various values of τ). **(b)** shows the concentration of mature GFP over time at several τ values. Although GFP concentration is high throughout the simulation when $\tau = 80$min, this does not necessarily correlate to high total GFP concentration, as GFP is allowed to remain and build up in the system for longer, rather than being quickly diluted out and being regenerated rapidly at a lower τ value of 30 minutes.

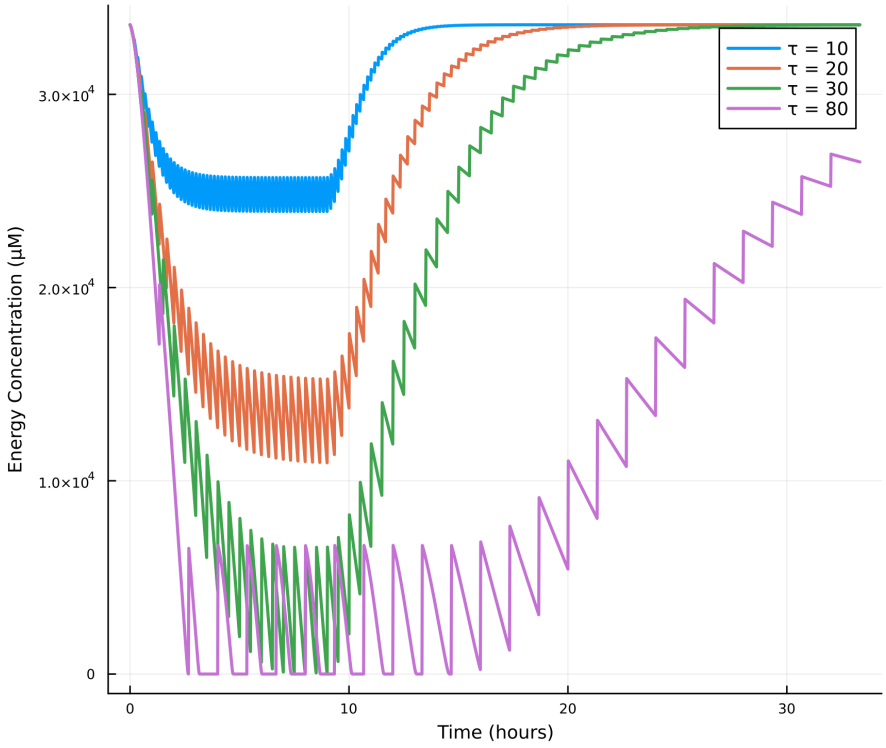


Figure 3.7: Graph showing the concentration of energy over time at several τ values. At short dilution intervals, the energy in the system is not fully used between each dilution cycle. At higher dilution intervals however, the system is able to fully use the available energy each dilution cycle. Therefore a τ value of around 30 minutes allows sufficient energy to be used each cycle, but not so much that the system spends extended periods of time without the energy to fuel TX-TL.

Combined optimisation

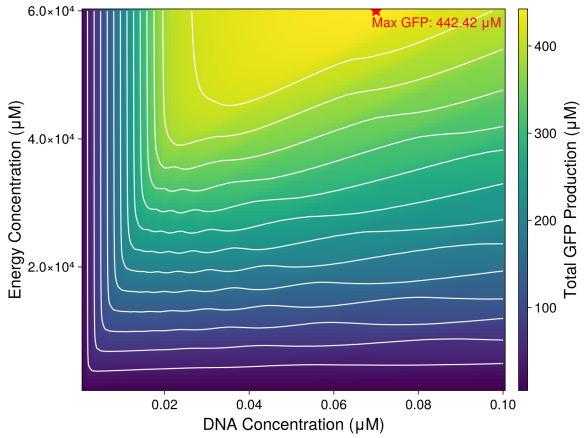
We then aim to optimise total GFP produced in the system by changing energy, DNA and dilution interval simultaneously:

$$\max_{e_0, D_0, \tau} \quad \int_0^T G_m(t) \quad \text{subject to} \begin{cases} 30000\ \mu M \leq e_0 \leq 60000\ \mu M, \\ 0.001\ \mu M \leq D_0 \leq 0.1\ \mu M, \\ 5\ \text{min} \leq \tau \leq 120\ \text{min}. \end{cases} \tag{3.12}$$

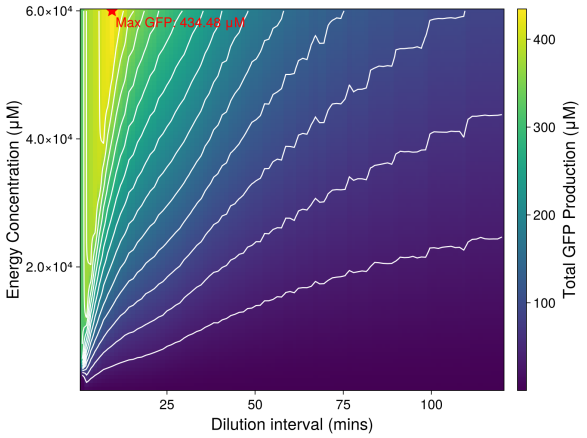
Plotting the total GFP output as a function of the three variables (see Figure 3.8) reveals that high total GFP values correlate with maximum energy levels of 60000 µM, low dilution intervals around 10 minutes, and DNA concentrations between 0.07-0.09 µM. The optimisation algorithms confirm the optimal values for e_0, D_0, and τ with greater precision, as can be seen in Table 3.7.

Algorithm	**Termination status**	**Optimal** e_0 (µM)	**Optimal** D_0 (µM)	**Optimal** τ (µM)	**Total GFP** (µM)
BBO	Max Iters	59999.9999	0.07077	10.9996	442.4672
LBFGS	Success	59999.9997	0.02502	20.1398	362.5462
Nelder-Mead	Success	59999.9992	0.05287	11.7380	441.8262
IPNewton	Fail	-	-	-	-

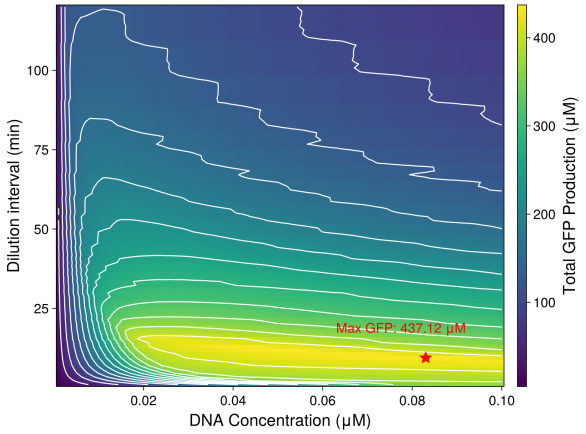
Table 3.7: Results from several algorithms for the optimisation of total GFP production by changing the energy concentration, DNA concentration and dilution interval of the system. Initial guesses at $e_0 = 50000$ µM, $D_0 = 0.02$ µM and $\tau = 20.0$. Although LBFGS successfully optimises both e_0 and D_0, it struggles to compute the derivative with τ considered (again due to the discrete effect of this variable on the optimisation landscape). Interior point methods result in complete solution failures due to an inability to calculate the Hessians of the discrete function. Nelder-Mead performs well, although the black-box DE algorithm converges better on the global optimum, achieving a marginally higher total GFP output.



(a) $\tau = 10.9996$ minutes.



(b) D_0=0.07077 µM.



(c) e_0=60000 µM.

Figure 3.8: Relationship between energy and DNA (3.8a), energy and tau (3.8b), and DNA and tau (3.8c) on the total GFP produced by the system (shown in colour). Each heat map is plotted with the optimal third variable calculated by the BBO algorithm from Table 3.7. Optimal DNA concentrations are significantly higher (around 0.07-0.085 µM) than those calculated in Figure 3.5 because a shorter dilution interval of 10min reduces mRNA accumulation and minimises periods of energy over-consumption by transcription. Maximum total GFP values shown on the figures are sampled manually from the vectors of solutions using `maximum()` and are lower than the BBO results from Table 3.7 due to the sampling resolution used in the heatmaps (each decision variable was only varied over a length of 100 - see supplementary code for full plotting details).

3.2.2 Optimisation of peak GFP

Optimising the concentration of peak GFP in the system is another problem we aim to solve:

$$\max_{e_0, D_0, \tau} \quad G_m(t) \quad \text{subject to} \begin{cases} 30000\ \mu M \leq e_0 \leq 60000\ \mu M, \\ 0.001\ \mu M \leq D_0 \leq 0.02\ \mu M, \\ 5\ \text{min} \leq \tau \leq 120\ \text{min}. \end{cases} \tag{3.13}$$

Once more, we begin by presenting the optimisation results for each variable individually.

Energy

Viewing the relationship between the energy supplied to the system and the resulting peak GFP concentration with a fixed DNA concentration and dilution interval:

$$\max_{e_0} \quad G_m(t) \quad \text{subject to} \begin{cases} 30000\ \mu M \leq e_0 \leq 60000\ \mu M, \\ D_0 = 0.005\ \mu M, \\ \tau = 20\ \text{min}, \end{cases} \tag{3.14}$$

reveals that once more, higher energy concentration leads to a higher peak GFP concentration (see Figure 3.9a). Lower energy concentrations reach their peak GFP concentration earlier, while higher energy concentrations reach a higher peak later in the experiment (see Supp. Figure A.6). We reason that this is because at higher e_0 concentrations it takes longer to fully consume all of the energy supplied to the system and reach peak GFP concentration. All four algorithms successfully converge at optimum energy values of 60000 µM (see Supp. Table A.1), confirming that the more energy that is supplied to the system, the greater the peak GFP it can reach is.

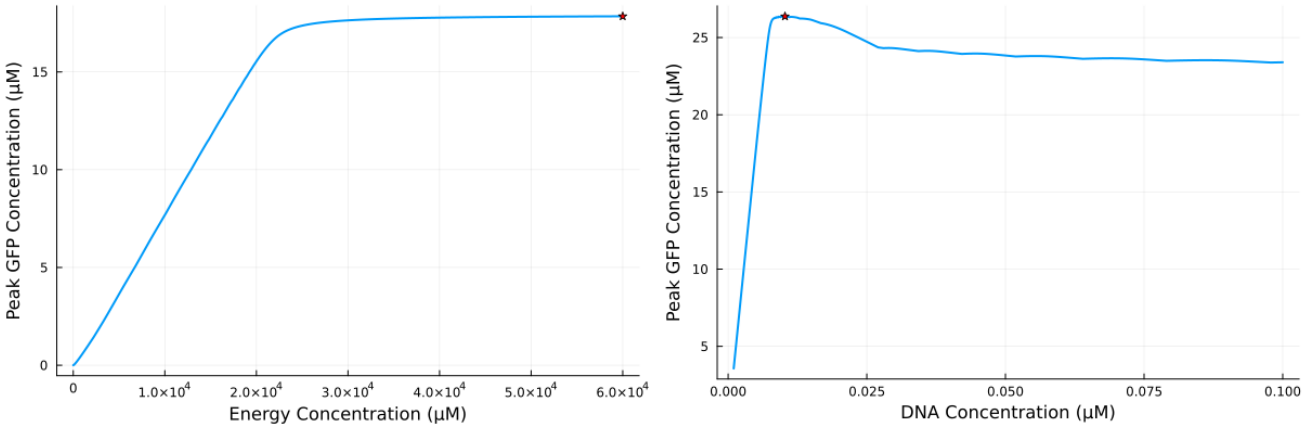
DNA

The relationship between DNA and peak GFP concentration:

$$\max_{D_0} \quad G_m(t) \quad \text{subject to} \begin{cases} e_0 = 33600\ \mu M, \\ 0.001\ \mu M \leq D_0 \leq 0.1\ \mu M, \\ \tau = 20\ \text{min}, \end{cases} \tag{3.15}$$

can be seen in Figure 3.9b. Similarly to the relationship between D_0 and the total GFP output, we reason that the decrease in peak GFP concentration beyond the optimal DNA concentration

is due to transcription dominating energy consumption beyond this point (see Supp. Figure A.7). Calculating the optimum DNA concentration using the optimisation algorithms reveals a DNA value lower than in the total GFP optimisation (from Table 3.5.), with DE optimisation converging to a value of 0.01561 µM (for the full results of peak GFP optimisation by changing e_0 and D_0, see Sup. Table A.2).



(a) Effect of energy concentration on peak GFP.

(b) Effect of DNA concentration on peak GFP.

Figure 3.9: Graphs showing the relationship between both energy concentration and DNA concentration on the peak GFP concentration reached by the cell-free system. **(a)** shows the effect of increasing the energy concentration supplied to the system on the peak GFP output. The initial near-linear relationship reaches a plateau at around 22000 µM, after which increases in energy concentration provide diminishing peak GFP returns. Star shows the maximum peak GFP concentration at an e_0 value of 60000 µM. **(b)** shows that increasing the DNA concentration results in increasing peak GFP concentrations, up until the optimal DNA concentration (shown with red star at D_0 =0.0102 µM and resulting in a GFP peak of 26.371 µM). Beyond this point, increased transcriptional resource usage reduces the GFP concentration that can be produced (see Figure A.7).

Dilution interval

We explore how the dilution interval τ affects the peak GFP concentration

$$\max_{e_0, D_0, \tau} \quad G_m(t) \quad \text{subject to} \begin{cases} e_0 = 336000\ \mu M, \\ D_0 = 0.005\ \mu M, \\ 5\ \text{min} \leq \tau \leq 120\ \text{min}, \end{cases} \tag{3.16}$$

revealing a different trend to the effect of τ on total GFP production (see Figure 3.10). The black-box DE algorithm is the only one capable of converging on the correct optimum dilution

interval of 120 minutes (see Table 3.8). Both LBFGS and the IPNewton algorithms fail to compute derivatives, and the Nelder-Mead algorithm converges on a local minimum, impeded by the one-dimensional nature of the problem.

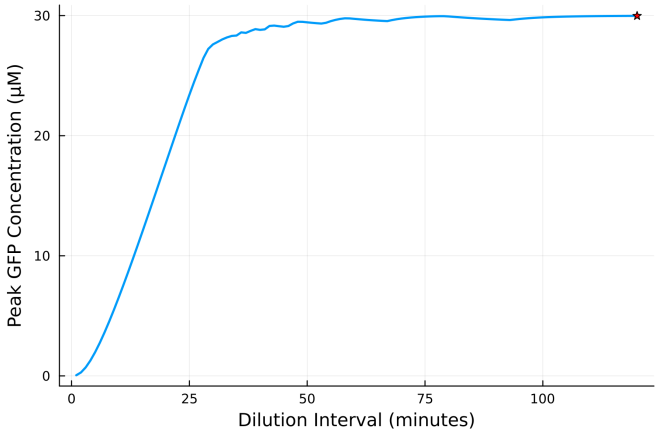


Figure 3.10: Relationship between increasing the dilution interval (τ) and the corresponding peak GFP concentration reached by the system. Maximum peak GFP concentration shown with a star at $\tau = 120$ minutes, resulting in 29.98 μM of GFP. Increasing the dilution interval beyond 30 minutes results in increased peak GFP concentrations, after which the effect plateaus: beyond this point, increasing the dilution interval provides minimal improvements to the peak GFP concentration. Observing the graph in Figure 3.6b, a longer dilution interval provides enough time for the energy in the system to be fully consumed within the first dilution cycle, producing a maximum peak early on in the simulation as high levels of GFP get translated in the first few dilution cycles before mRNA begins to be diluted out.

Combined peak GFP optimisation

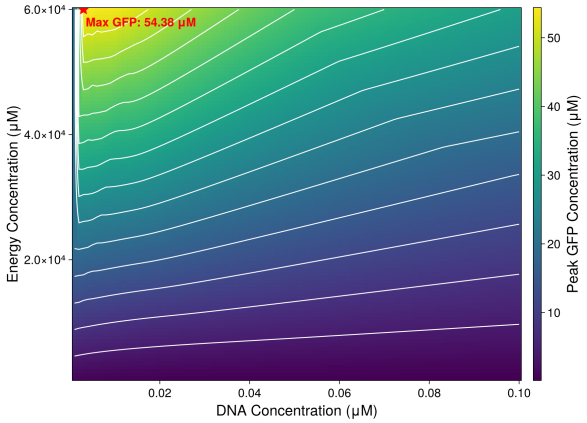
Finally, we apply the optimisation algorithms to the combined problem described in Equation 3.13. Using the optimisation results from the individual experiments, the initial guesses can be refined, allowing the Nelder-Mead and LBFGS algorithms to converge closer to the global minimum (see Table 3.9). The algorithm results can be seen displayed visually in Figure 3.11.

Method	Termination Status	Optimal τ (min)	Peak GFP (µM)
BBO	MaxIters	119.9999	29.9794
Nelder-Mead	Success	30.0250	27.5400
LBFGS	Success	20.0000	17.7115
IPNewton	Fail	-	-

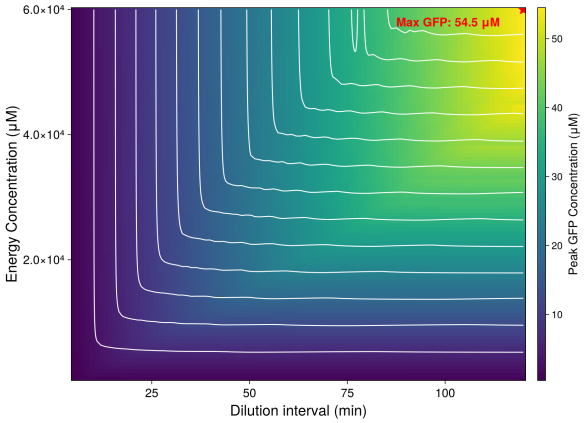
Table 3.8: Results from several algorithms attempting to optimise the dilution interval of the system to maximise peak GFP concentration. Initial guess at $\tau = 20$ minutes. Differential evolution optimisation successfully converges on the optimum dilution interval, as confirmed by the relationship between τ and peak GFP shown in Figure 3.10. Nelder-Mead fails to correctly navigate the one-dimensional problem space, while the gradient-based methods fail to compute derivatives, with LBFGS failing to move from the initial guess and interior point failing completely.

Method	Termination Status	Energy (µM)	DNA (µM)	Dilution Interval (τ)	Peak GFP (µM)
BBO	MaxIters	59999.99999999	0.00318	119.80	54.49
Nelder-Mead	Success	55000.00000000	0.01200	150.03	48.24
LBFGS	Success	59999.99860177	0.00448	77.41	53.96
IPNewton	Fail	-	-	-	-

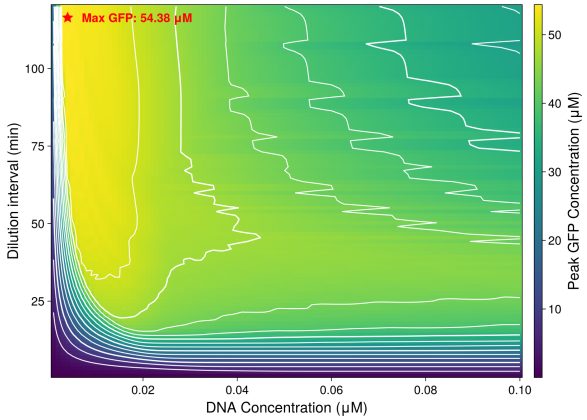
Table 3.9: Results from several algorithms attempting to maximise peak GFP concentration by changing the DNA, energy and dilution interval of the system. Initial guesses of $e_0 = 55000$ µM, $D_0 = 0.1$ µM and $\tau = 100$ minutes allowed the LBFGS algorithm to escape local minima and converge near to the optimum solution, reaching high peak GFP concentrations of close to 54.49 µM, the global optimum achieved by BBO. The Nelder-Mead algorithm, although achieving a peak GFP concentration of 48.24 µM, once again broke the constraints of the optimisation problem assigned to it (reaching a dilution interval of 150 minutes), illustrating the difficulty of applying simplex-based algorithms to constrained problems. Once more, the interior point approach fails to calculate the Hessian, resulting in complete solver failure.



(a) $\tau = 120$ minutes.



(b) D_0=0.00318 µM.



(c) e_0=60000 µM.

Figure 3.11: Visual representation of the optimisation results shown in Table 3.9, displaying the effect of energy and DNA (3.11a), energy and tau (3.11b), and DNA and tau (3.11c) on the corresponding peak GFP concentration of the system. Each heat map plotted with optimal third variable calculated by the BBO algorithm from Table 3.7. **(a)** shows that high e_0 values correlate to higher peak GFP concentration, with a low D_0 concentration of 0.00318 producing the greatest GFP concentration in **(c)** as well. Similarly, **(b)** shows that longer dilution intervals correspond to high peak GFP concentrations.

3.2.3 Optimisation of time taken to reach steady state

As defined in Equations 2.10-2.12, we explore the effect on steady-state characteristics of changing energy concentration, DNA concentration and the dilution interval of the system. We begin by minimising the time taken for the system to reach steady state:

$$\min_{e_0, D_0, \tau} \quad t_{\text{ss}} \quad \text{subject to:} \begin{cases} 30000\ \mu M \leq e_0 \leq 60000\ \mu M, \\ 0.001\ \mu M \leq D_0 \leq 0.02\ \mu M, \\ 5\ \text{min} \leq \tau \leq 120\ \text{min}. \end{cases} \tag{3.17}$$

Energy

We start by exploring the effect of energy concentration on the time taken for all species to reach steady state, shown in Figure 3.12. Steady state is only detected by the function upon completion of C_{ref} dilution cycles. This means that several e_0 values can reach steady state at the same dilution period, hence the flat intervals on the graph. Interestingly, while at low energy values the system reaches steady state quickly, the maximum time to reach steady state is calculated at around 2.4×10^4 µM, decreasing again for energy concentrations beyond this point.

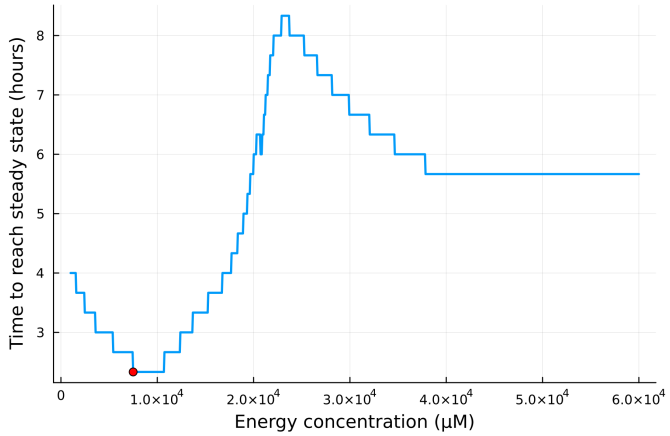


Figure 3.12: Relationship between the energy concentration and the time taken for every species in the system to reach steady state. Minimum time shown in red at $e_0 = 7496.50$ µM, reaching steady state after 2.33 hours. Performed at a required number of cycles of $C_{req} = 3$.

To explain this feature, we graph the GFP concentration over time at several relevant energy values (see Supp. Figure A.8a), finding minimal difference between the G_m curve at $e_0 = 24,000$ µM and $e_0 = 40,000$ µM. Assuming that the steady state function finds a point of fluctuation

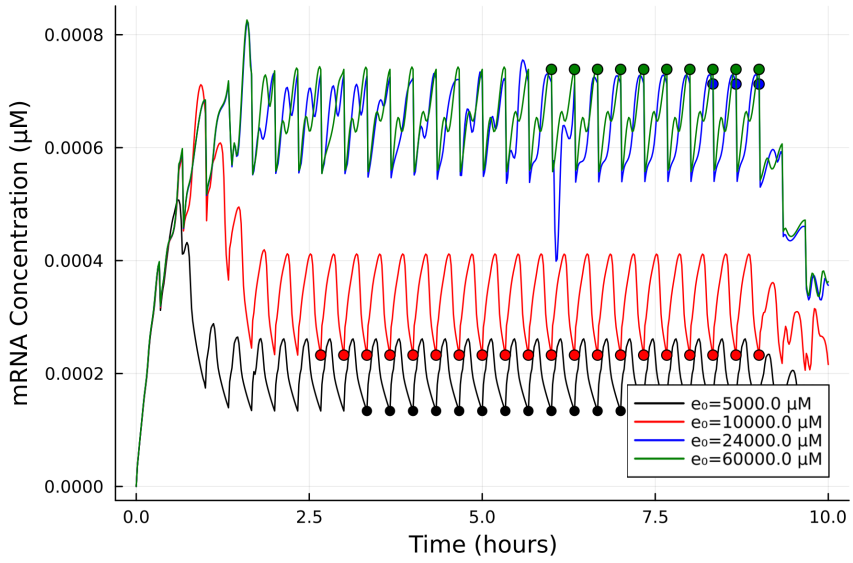


Figure 3.13: Figure showing the mRNA concentration over time at several energy concentrations. Steady states shown with markers and reached at 3.33h (black), 2.67h (red), 8.33h (blue) and 6.00h (green). Simulation run until 10 hours due to solver errors returning negative values for low concentrations of mRNA after the washout phase. Fluctuations can be observed at an energy concentration of $e_0 = 24,000$ that imply the system is not at steady state. Re-solving the system using an alternate ODE solver corrected these fluctuations (see Figure A.9).

in one of the other species concentrations, we proceed to plot each species concentration over time individually (see folder in supplementary code), finding that the mRNA concentration over time displays instability for e_0 = 24,000 μM, but not at e_0 = 40,000 μM (see Figure 3.13). We hypothesise that this is due to solver errors when dealing with the stiffness of the system, particularly the minimal mRNA concentrations: recalculating the species concentrations over time using Julia's `Rosenbrock23()` (see Supp. Figure A.9) and `Rodas5()` algorithms does not produce the fluctuations that can be observed in Figure 3.13. Nevertheless, we adjust the steady-state function to only use the absolute difference between dilution cycles of the G_m curve, reducing the reliance on precise mRNA concentration calculations, returning the relationship shown in Figure 3.14.

DNA

We analyse the effect of DNA on the time taken to reach steady state, with the steady state function modified to detect solely for GFP concentration, observing that increasing the DNA concentration results in a faster time taken for G_m to reach steady state (see Figure 3.15). Prior experiments reveal that a higher DNA concentration does not necessarily correspond to a higher GFP output

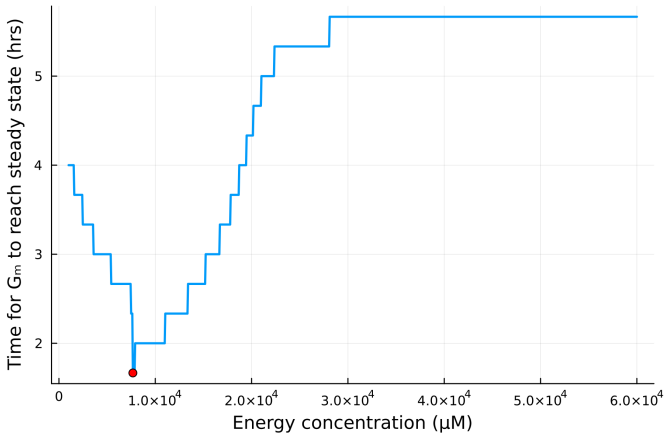


Figure 3.14: Relationship between the energy concentration and the time taken for mature GFP (rather than all system species) to reach steady state. At low energy concentrations, less energy in the system means less TX-TL can occur, reducing the time taken for the minimal GFP concentration to reach a steady state of production. Minimum time shown in red at $e_0 = 7673.6737$ μM, reaching steady state after 2 hours and 6 dilutions. Performed at a required number of cycles of $C_{req} = 2$.

(see Figure 3.3 and 3.4). Beyond a certain threshold, increasing DNA concentration results in energy allocation to transcription rather than translation. Thus, at higher DNA concentrations less GFP is produced, and the system can reach steady state faster (see Supp. Figure A.10).

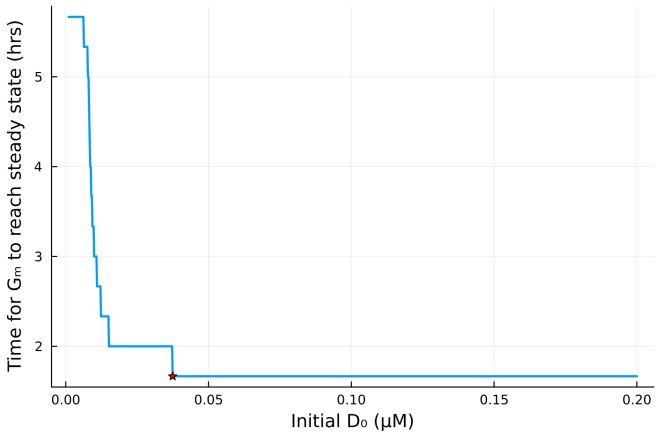


Figure 3.15: Relationship between the DNA concentration and the time taken for the mature GFP concentration to reach steady state. Minimum time shown in red at D_0=0.03745 μM, reaching steady state after 2 hours and 6 dilutions. Performed at a required number of cycles of $C_{req} = 2$.

Dilution interval

Understanding the relationship between dilution interval and the time taken to reach steady state seems more intuitively straightforward: the longer the dilution interval, the longer it takes to achieve the required number of dilutions for the system to reach steady state. Plotting this relationship confirmed this trend, albeit with some fluctuation between around 25-50 minutes that is not explored in detail here (see Figure 3.16).

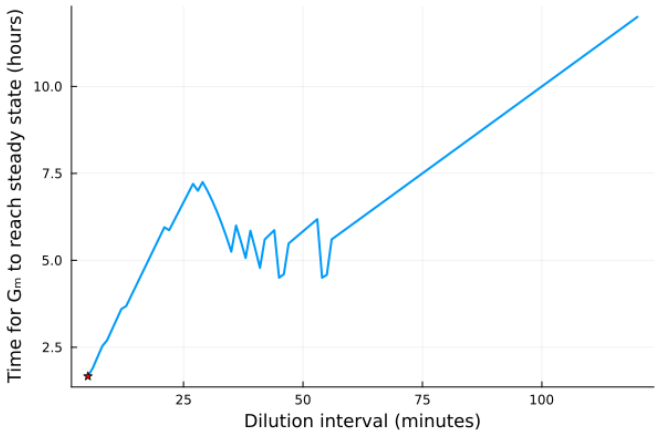


Figure 3.16: Relationship between the dilution interval and the time taken for mature GFP to reach steady state. Minimum time shown in red at a dilution interval of 5 minutes, reaching steady state after 1.67 hours and 20 dilutions. C_{req}=3.

Combined optimisation of time taken to reach steady state

Performing a combined optimisation simulation reveals the results shown in Table 3.10. Due to both Nelder-Mead and the gradient-based algorithms failing to escape local minima, Julia's Particle Swarm Optimisation algorithm `PSO()` was also included in the results (with 50 particles and a maximum iteration number of 100) to verify the accuracy of the differential evolution solution [56].

Algorithm	Optimum e_0 (µM)	Optimum D_0 (µM)	Optimum τ (min)	Time to Steady State (h)
BBO	1000.0016	0.199998	5.0194	0.3346
PSO	1000.0	0.2	5.0192	0.3346
NM	75000.025	0.01	20.0	5.6667
LBFGS	50000.0	0.01	20.0	5.3333
IPNewton	50000.0	0.01	20.0	5.3333

Table 3.10: Optimisation results for minimising the time taken to reach steady state. Initial guesses at $e_0 = 50000$ µM, $D_0 = 0.01$ µM and $\tau = 20.0$ minutes. The Nelder-Mead algorithm struggled to remain in the constraints applied to the optimiser, extending the energy concentration beyond the upper bound of 60000 µM. Gradient-based methods failed to escape local minima, struggling due to the discrete nature of the function. Both the differential evolution algorithm and the particle swarm algorithms successfully converged on optimum values for each decision variable, reaching steady state within 0.3346 hours.

3.2.4 Optimisation of GFP concentration at steady state

Prioritising the time taken to reach steady state is less relevant to biotechnological applications without also considering the GFP output at this steady state: low energy concentrations and high DNA concentrations are not conducive to high GFP output, and shorter dilution intervals reduce resource efficiency through frequent dilutions. Considering the GFP concentration at steady state is more relevant to real-world situations, and can be easily supplied to the optimisation algorithms by changing the output of the optimisation function to return the G_m concentration once at steady state, rather than the time taken to reach steady state. Results for the optimisation of the GFP concentration at steady state are shown in Table 3.11.

Algorithm	**Optimum e_0 (µM)**	**Optimum D_0 (µM)**	**Optimum τ (min)**	**GFP concentration at steady state (µM)**
BBO	59999.98	0.00746	36.00	40.29
PSO	60000.00	0.00761	35.97	40.28
Nelder-Mead	75000.03	0.01000	20.00	-1.0e6
LBFGS	50000.00	0.01000	20.00	-1.0e6
IPNewton	50000.00	0.01000	20.00	-1.0e6

Table 3.11: Optimisation results for maximising the GFP concentration at steady state. Initial guesses at $e_0 = 50000$ µM, $D_0 = 0.01$ µM and $\tau = 20.0$ minutes. A penalty value of 1.0e6 is used to represent that a combination of decision variables prevented the system from reaching steady state in the simulation time-frame. Both the BBO and PSO algorithms achieve similar GFP concentrations at steady state, with an optimal τ value slightly greater than for the total GPF optimisation, and with DNA values between that of the peak GFP and total GFP optimisation experiments.

Discussion

The application of these optimisation algorithms provides a number of insights into how cell-free systems can be improved. Analysing one dimensional Lotka-Volterra problems emphasises the importance of gradient calculations and considering appropriate initial guess values. At low dimensions, graphical representations can help elucidate the relationship between system variables and desired outcomes. However, as the number of variables increases, optimisation algorithms are necessary to explore the multidimensional problem space and converge on optimum solutions. For non-continuous CFPS, gradient-based algorithms are poorly suited to computing the derivatives of non-continuous functions. However, informed initial guesses for optimal conditions can improve their ability to escape local minima and converge at correct solutions. Of the derivative-free algorithms used, Nelder-Mead performs poorly for low-dimensional problems without sufficiently informative initial guesses, but converges better on global optima for multidimensional problems, albeit struggling to remain within defined constraints. Evolutionary algorithms display versatility across problems, making few assumptions about the underlying fitness landscape and incorporating stochasticity into their search processes. In the future, further stochastic optimisation techniques, such as particle swarm, could be used to build on this work [57].

Understanding the effect of energy, DNA and dilution interval on CFPS allows biotechnologists to optimise these systems to maximise efficiency and minimise cost. Compared to initial experiments completed by the group, the total GFP, peak GFP, time taken to reach steady state and GFP concentration at steady state were all improved individually (see Table 4.1). Future work could aim to simultaneously optimise some of these targets, necessitating appropriate weighting in the objective function for each target: for example, minimising both the time taken to reach steady state while simultaneously maximising the GFP concentration at this steady state. We

also reveal here the threshold at which increasing energy and DNA no longer provides meaningful improvements to GFP production, a consideration important for improving resource efficiency. In future work, incorporating a cost function to optimise the cost of increasing these components relative to GFP returns could further improve design strategies.

	Total GFP produced (µM)	Peak GFP produced (µM)	Time taken to reach steady state (hours)	GFP at steady state (µM)
Microchemostat	125.5792	17.7115	5.33	17.5449
Optimised	442.4672	54.7768	0.3346	46.41

Table 4.1: Table showing the results of the simulation when run at the original microchemostat values compared to simulations performed with values optimised for each problem. Each optimisation performed separately; future work should aim to explore optimising all targets simultaneously.

Validating these optimisation results using experimental data is another future research prospect. Given the coarse-grained nature of the cell-free model, it is expected that the experimental results will differ from the optimisation results calculated here. The model uses simplified kinetics and neglects stochastic effects that are important in small-volume cell-free systems [10]. Additionally, real microchemostat experiments will inevitably show deviations from the ODE model due to nuclease degradation, protein misfolding, or batch-to-batch extract variation [58].

Running microchemostat experiments with the variables calculated by the optimisation algorithms and comparing them to the simulation results could help validate the utility of these approaches to improving experimental setups. Learning from experimental data allows for the incorporation of additional considerations that help make the optimisation algorithms more relevant to industrial applications. For example, considering the effect of molecular crowding (for example, as GFP concentration increases during prolonged reaction periods) has gained increased attention: Ge et al. found that transcription was significantly enhanced in the presence of crowding agents, mimicking natural conditions in the cell [59]. Additionally, considering the system with alternative protein products (e.g. monoclonal antibodies) that require different energy and TX-TL considerations could extend the utility of the optimisation algorithms beyond just GFP production.

In conclusion, this project demonstrates that a simple cell-free model can be combined with derivative-free optimisation algorithms to reveal valuable design principles for understanding CFPS and improving system efficiency. Although entirely theoretical and limited by the coarse-grained nature of the ODE model, the approach outlined here provides a foundation for future research

into making cell-free biomanufacturing more robust, scalable, and economically viable.

Appendix

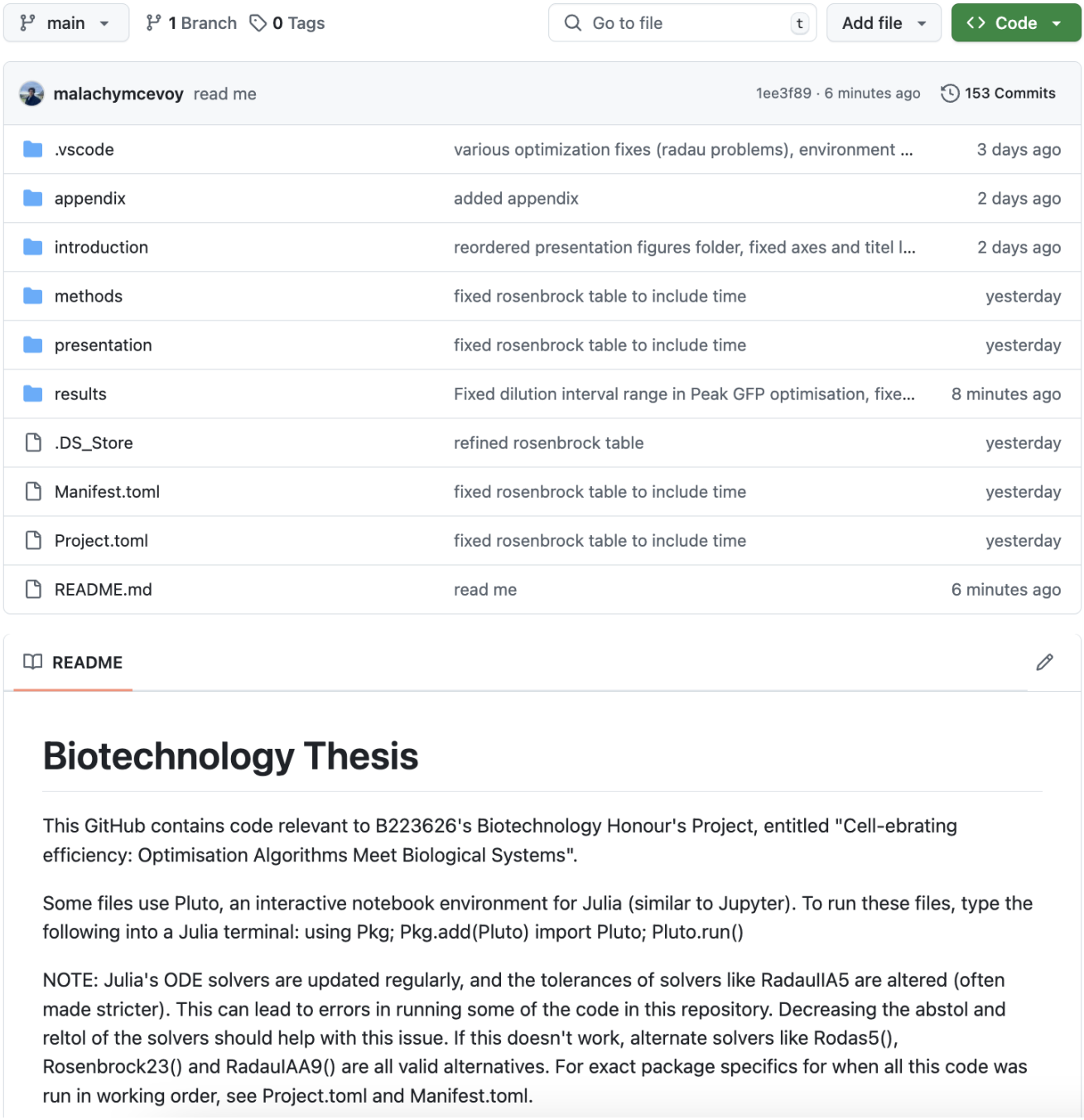


Figure A.1: All code relevant to this project can be found at https://github.com/malachymcevoy/biotechnologythesis/ in the respective section of the GitHub. Figures located in the Appendix will be located under the folder where they are described in-text.

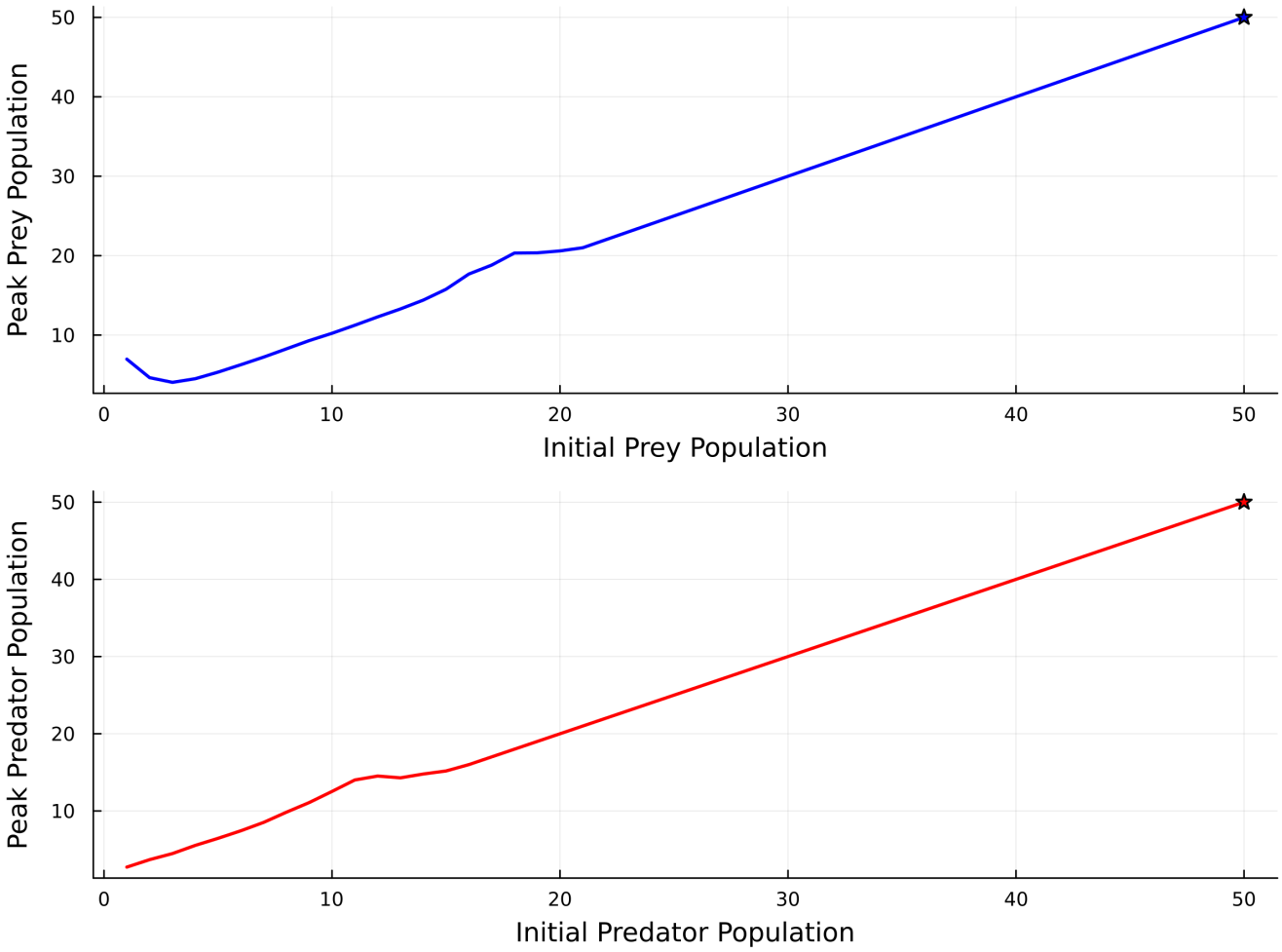


Figure A.2: Figure showing the relationship between increasing the initial populations of prey (blue) and predators (red) on their peak values over the course of the simulation, representing the optimisation functions in Equations 3.3 and 3.4. Maximum points are labelled with a star and are calculated using Julia's `maximum()` function. Peak prey population is 50, at a x_0 of 50 (blue star). Peak predator population is 50 at a y_0 of 50 (red star).

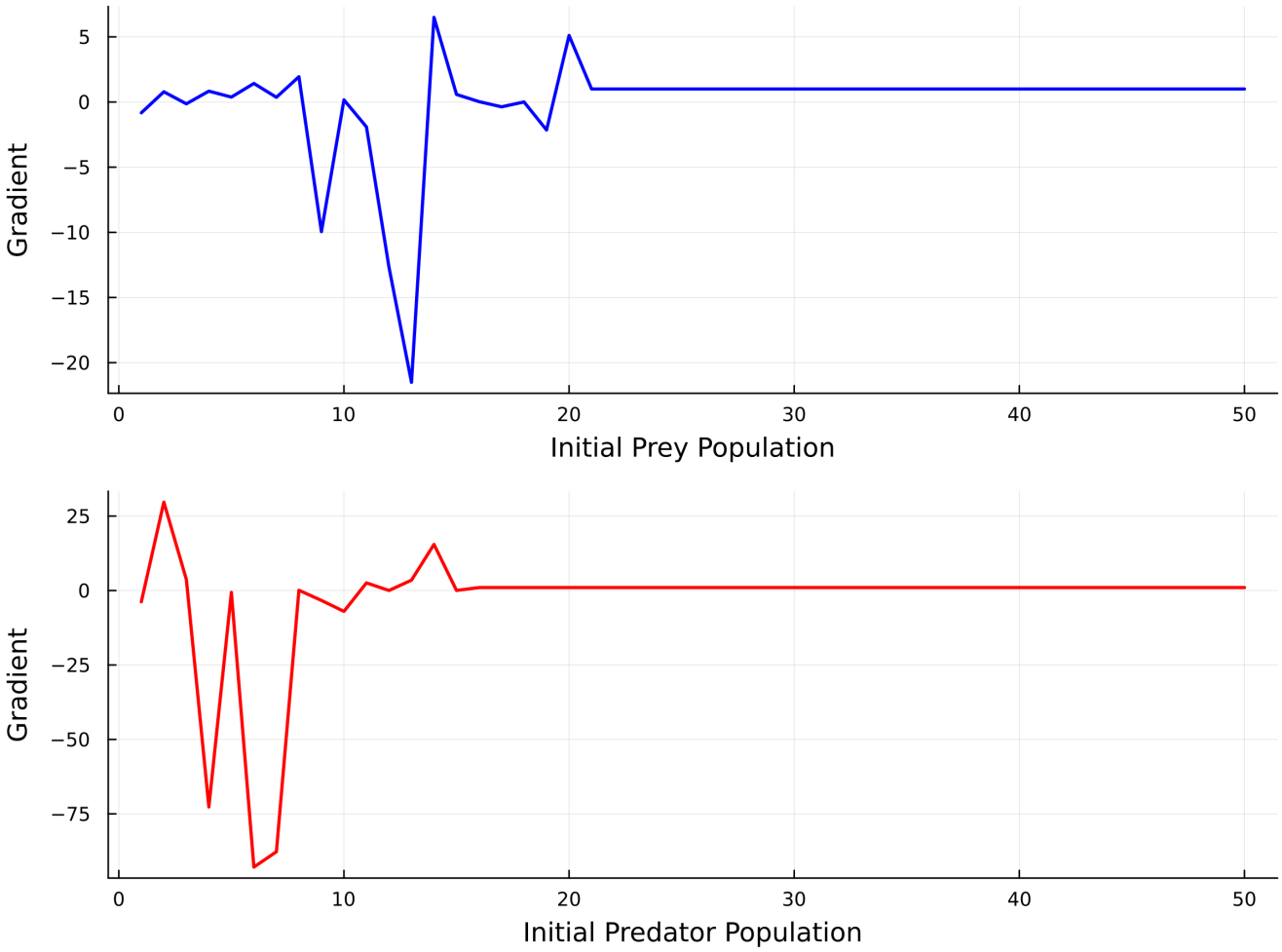


Figure A.3: Figure showing the gradient of the optimisation functions displayed in Figure A.2 and represented by Equations 3.3 and 3.4. Gradients computed by `AutoZygote()` across a range of x_0 and y_0 values and are accurate from about $x_0 = 22$ and $y_0 = 16$ onwards. However, before these points, gradient calculations are inaccurate and the landscape is rugged. This means that gradient-based algorithms struggle to escape local minima, especially for the extreme gradient decreases in the predator gradient, meaning that only gradient-free approaches successfully converge on optimum y_0 values.

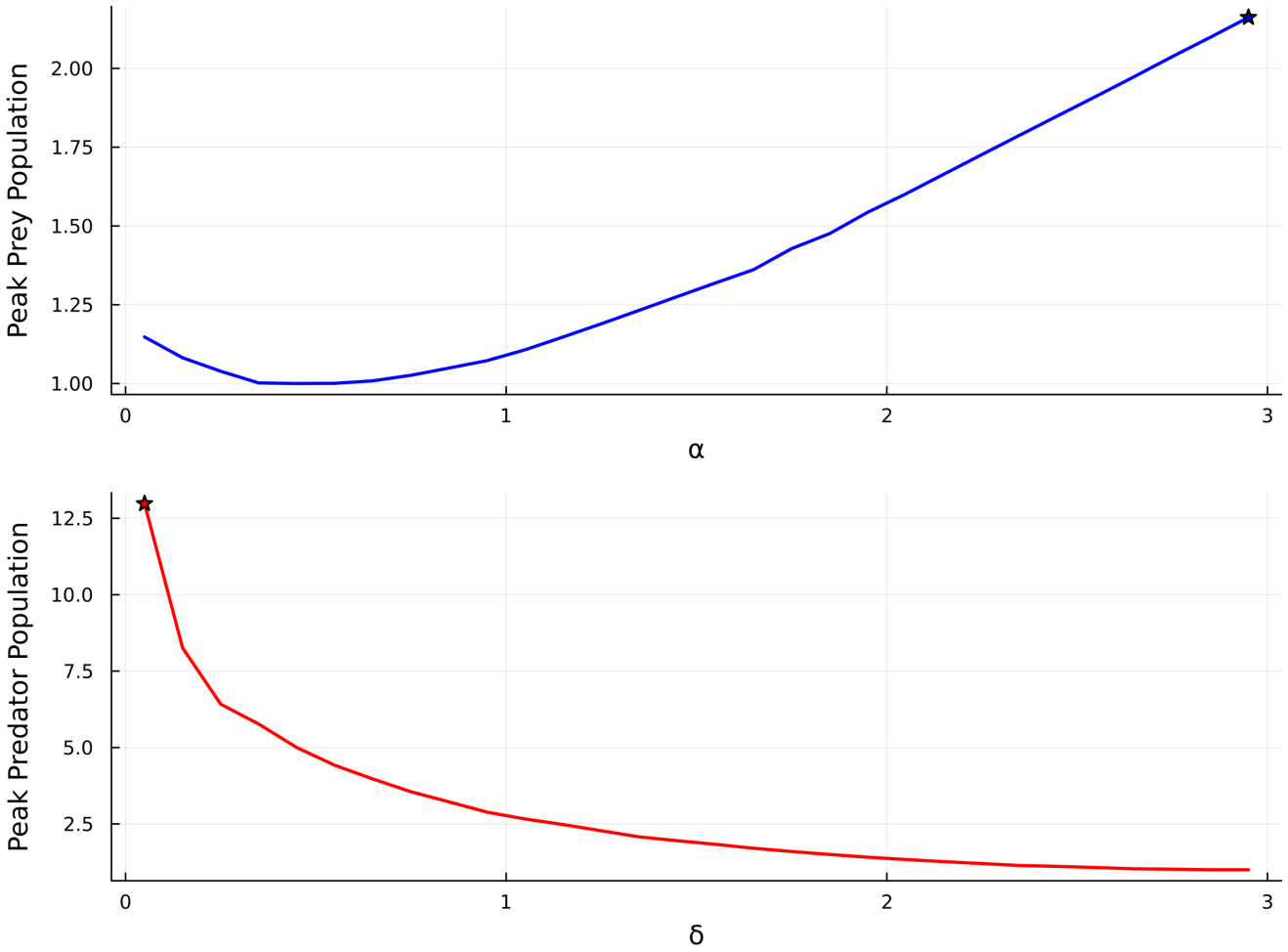


Figure A.4: Figure showing the relationship between increasing the prey birth rate (α) and the predator reproduction rate per prey consumed (δ), as represented by Equations 3.5 and 3.6. Unsurprisingly, a greater prey birth rate results in a greater peak prey population. Counter-intuitively, decreasing δ increases the peak predator population as it allows prey population to grow before predators surge (see interactive Lotka-Volterra simulation in supplementary code). Maximum points calculated using Julia's `maximum()` function as 2.1619 at $\alpha = 2.95$, and 12.9778 at $\delta = 0.05$.

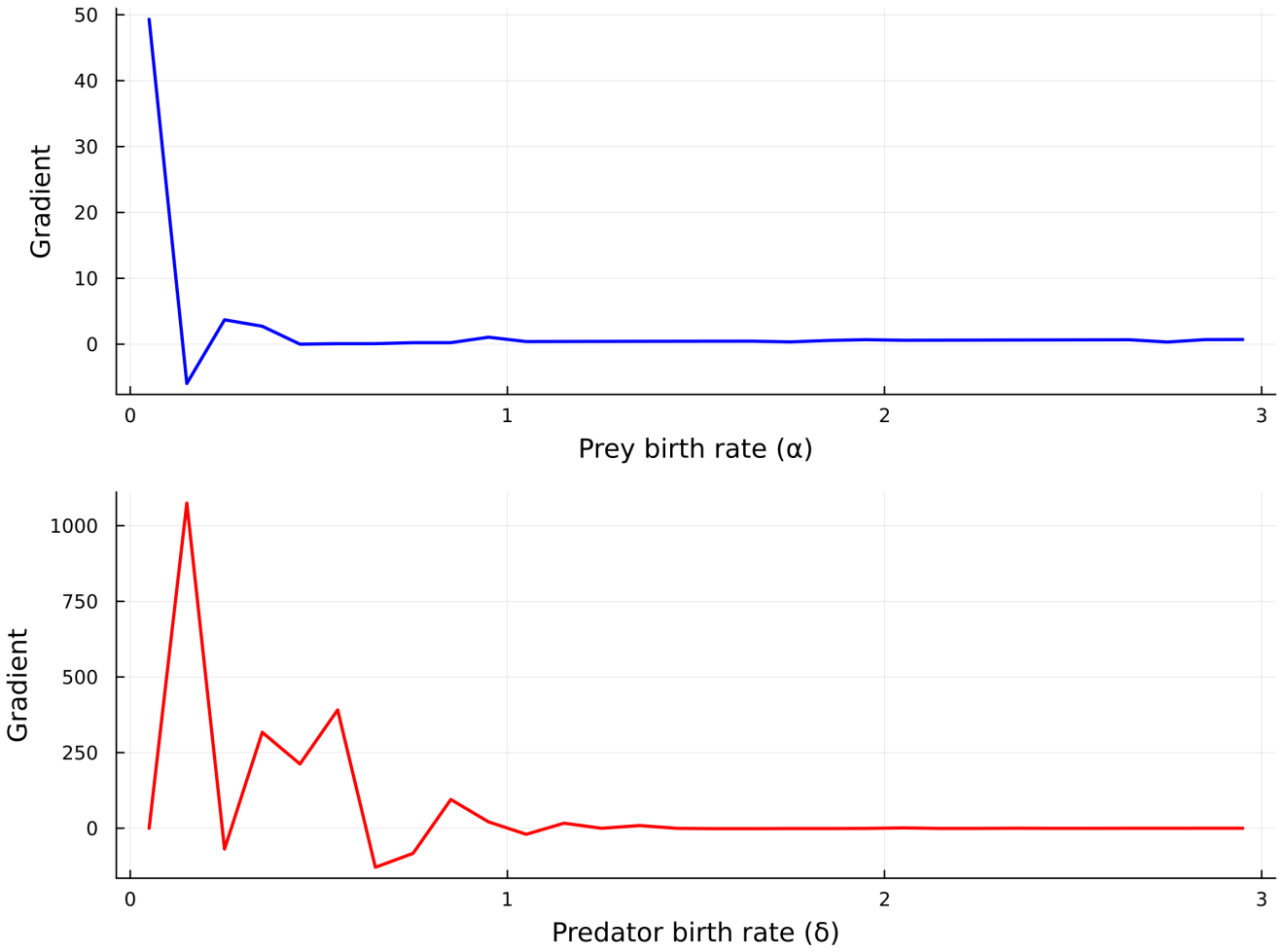


Figure A.5: Figure showing the gradients of the optimisation functions 3.5 and 3.6, shown visually in Figure A.4. Gradients computed by Zygote, with slight inaccuracies at low values of α, and an extremely rugged landscape at δ values below 1 due to the instability of the system. These local minima in δ's gradient landscape result in the LBFGS algorithm getting stuck at initial guesses below 1.

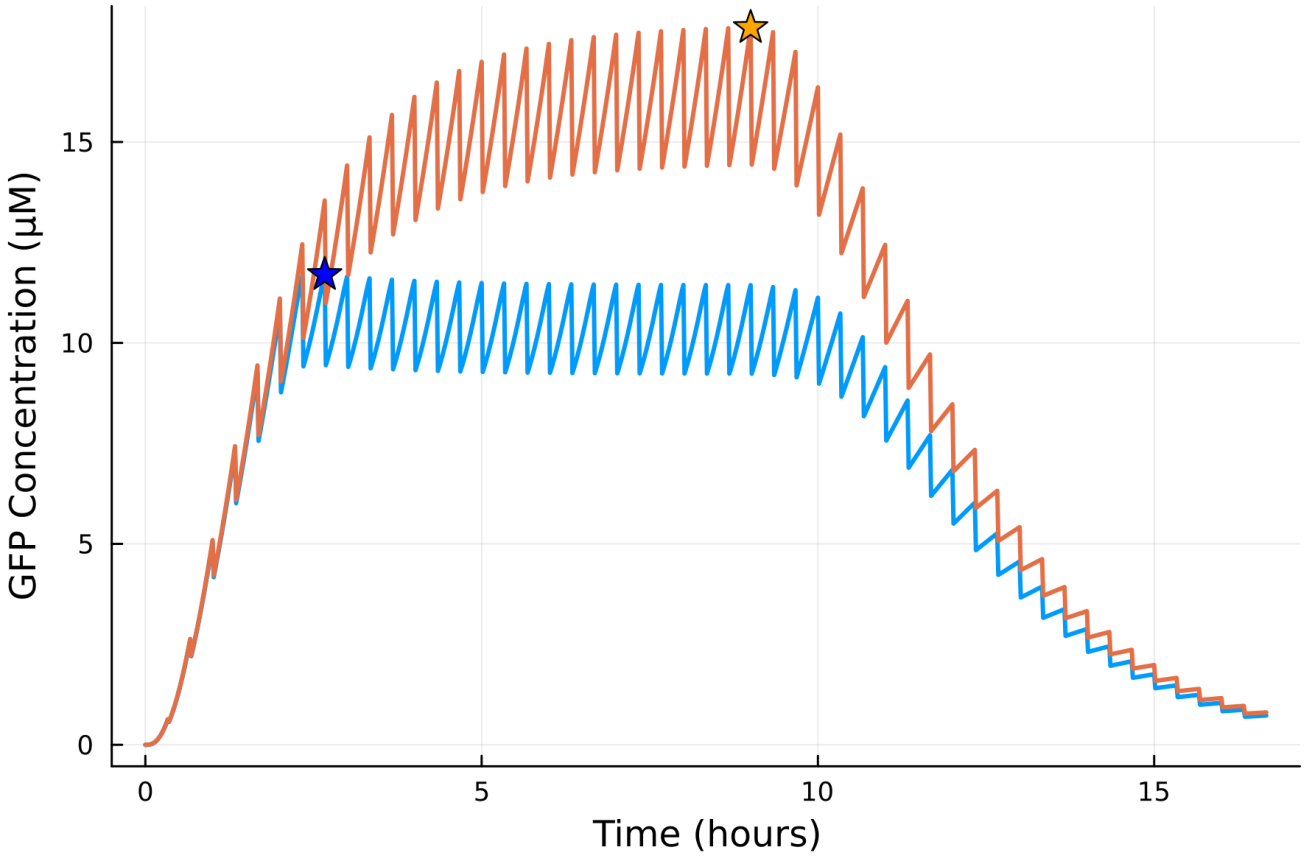


Figure A.6: GFP concentration over time shown at energy concentrations of 15000 µM (blue) and 60000 µM (red). Peak GFP concentrations is reached for $e_0 = 15000$ at 11.6995 µM (blue star) earlier in the experiment, as less time is taken to the energy in the system than at $e_0 = 60000$ µM where peak GFP concentration is achieved at 17.8462 µM (red star), using a greater amount of energy to build up a high GFP concentration by the end of the simulation.

Algorithm	**Termination Status**	**Optimum e_0 (µM)**	**Peak G_M (µM)**
Nelder-Mead	Success	59999.9889	17.8462
BBO	MaxIters	59999.9999	17.8462
L-BFGS	Success	59999.9999	17.8462
IPNewton	Success	59999.9999	17.8462

Table A.1: Results from several optimisation algorithms attempting to optimise the peak GFP concentration by changing energy concentration between 30000–60000 µM. Initial guess of 33600 µM. D_0=0.005 µM, τ=20 minutes. Interestingly, despite the one-dimensionality of the problem, Nelder-Mead successfully converges at the upper bound ot the problem space, although with slightly less accuracy than the other algorithms, all of which achieve a peak GFP concentration of 17.8462 µM.

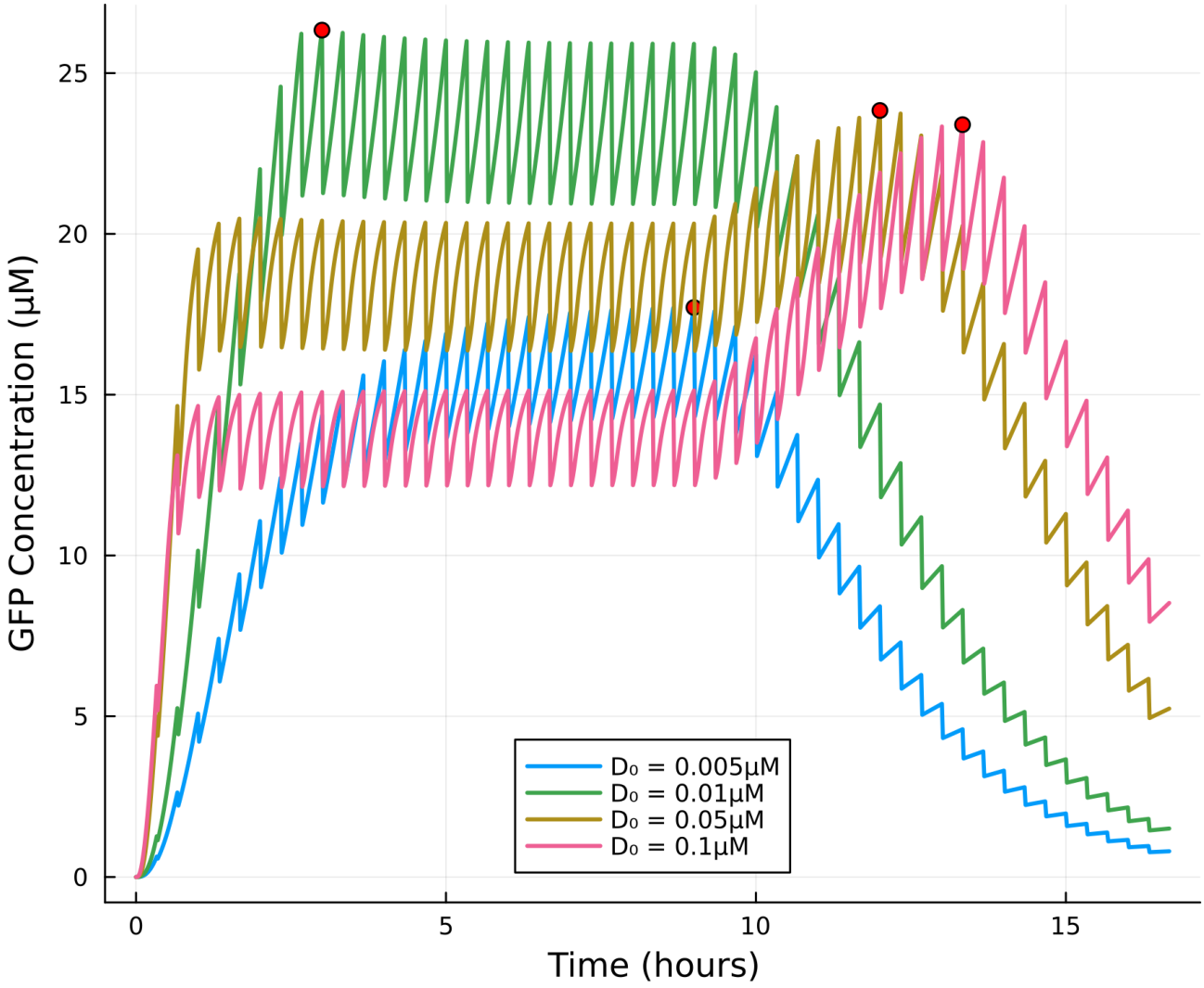
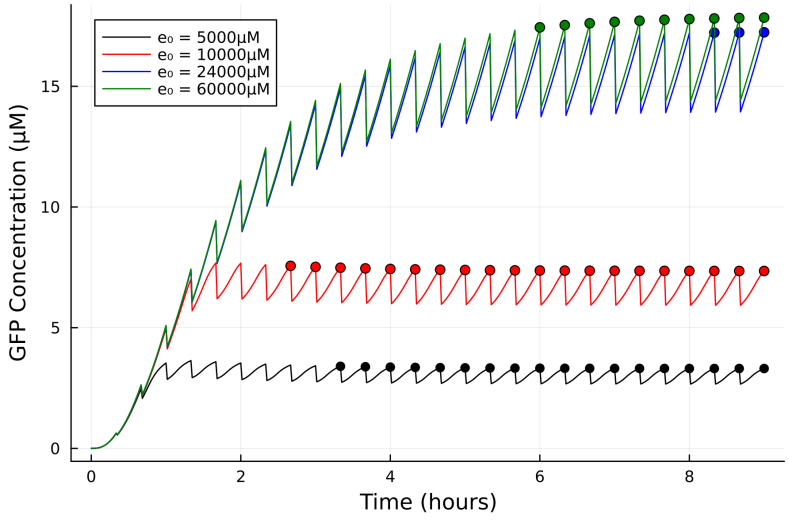


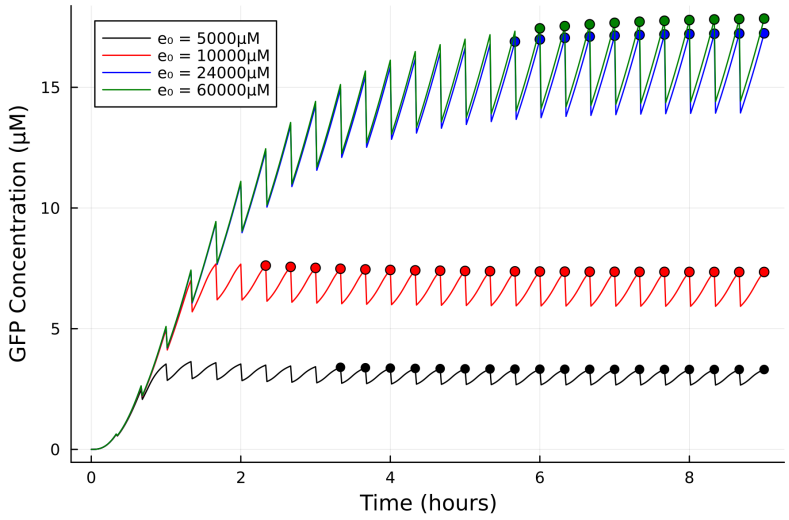
Figure A.7: Figure showing the GFP concentration over time at several values of DNA. Peak GFP concentrations shown in red at 17.712 μM for D_0=0.005 μM after 9 hours (blue), 26.337 μM for D_0=0.01 μM after 3 hours (green), 23.838 μM for D_0=0.05 μM after 12 hours (beige) and 23.397 μM for D_0 =0.1 μM after 13.33 hours (pink). Peak GFP values for DNA concentrations higher than 0.01 μM are only achieved after the washout phase ends; before this point, a large proportion of energy is diverted into transcription (see Figure 3.4), reducing the GFP produced.

Algorithm	**Termination Status**	**Optimal e_0 (µM)**	**Optimal D_0(µM)**	**Peak GFP (µM)**
BBO	MaxIters	60000.0000	0.01561	47.0098
Nelder-Mead	Success	59999.9999	0.01642	46.9560
LBFGS	Success	59999.9999	0.01561	47.0098
IPNewton	Success	60000.0000	0.01561	47.0098

Table A.2: Optimisation of energy and DNA concentration for peak GFP production using different algorithms. $\tau = 20$ minutes, and initial guesses at e_0=50000 µM and $D_0 = 0.02$ µM.



(a) Steady state detection applied to all species.



(b) Steady state detection applied to G_m.

Figure A.8: Figure showing the mature GFP concentration over time at energy concentrations of $e_0 = 5,000$ µM (black), $e_0 = 10,000$ µM (red), e_0 =24,000 µM (blue) and e_0=40,000 µM (green) with periods of steady state labelled with circles. **(a)** shows steady state detection applied to all species, resulting in steady state being reached much later for $e_0 = 24,000$ µM than $e_0 = 40,000$ µM, despite having an extremely similar G_m curve. **(b)** shows steady state detection applied to just G_m, excluding the fluctuations in mRNA that can be observed in Figure 3.13 and resulting in more accurate steady state calculations for G_m.

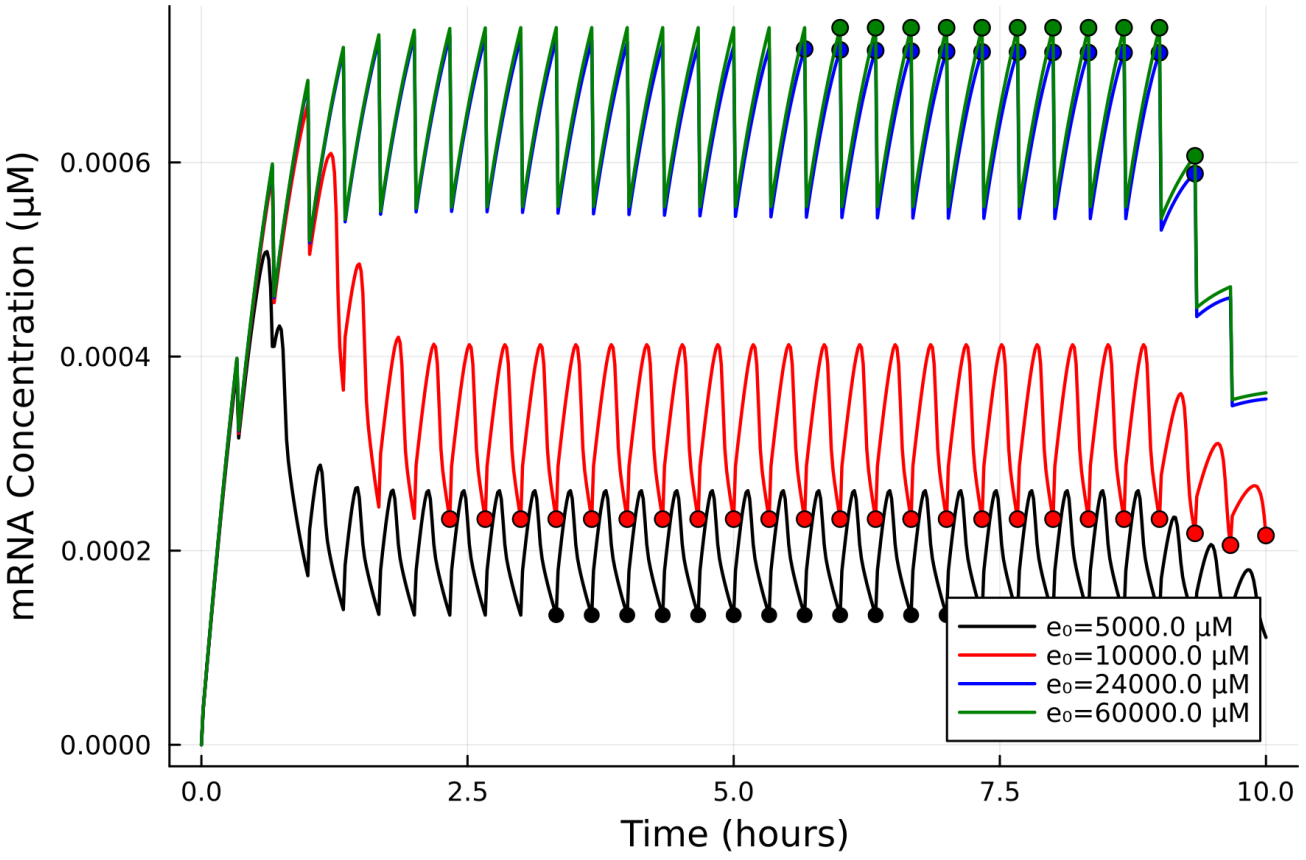


Figure A.9: Figure showing the mRNA concentration over time at several energy concentrations, solved using Julia's `Rosenbrock23` algorithm, rather than the `RadauIIA9()` algorithm used in Figure 3.13, reducing the numerical errors that lead to delayed steady state calculation for mRNA. Steady states shown with circles, showing more accurate calculations of steady state at 2.33h (red), 3.33h (black), 5.66h (blue) and 6.00h (green), more closely representing the effect of energy concentration on the time taken for the system species to reach steady state, as shown in Figure 3.14.

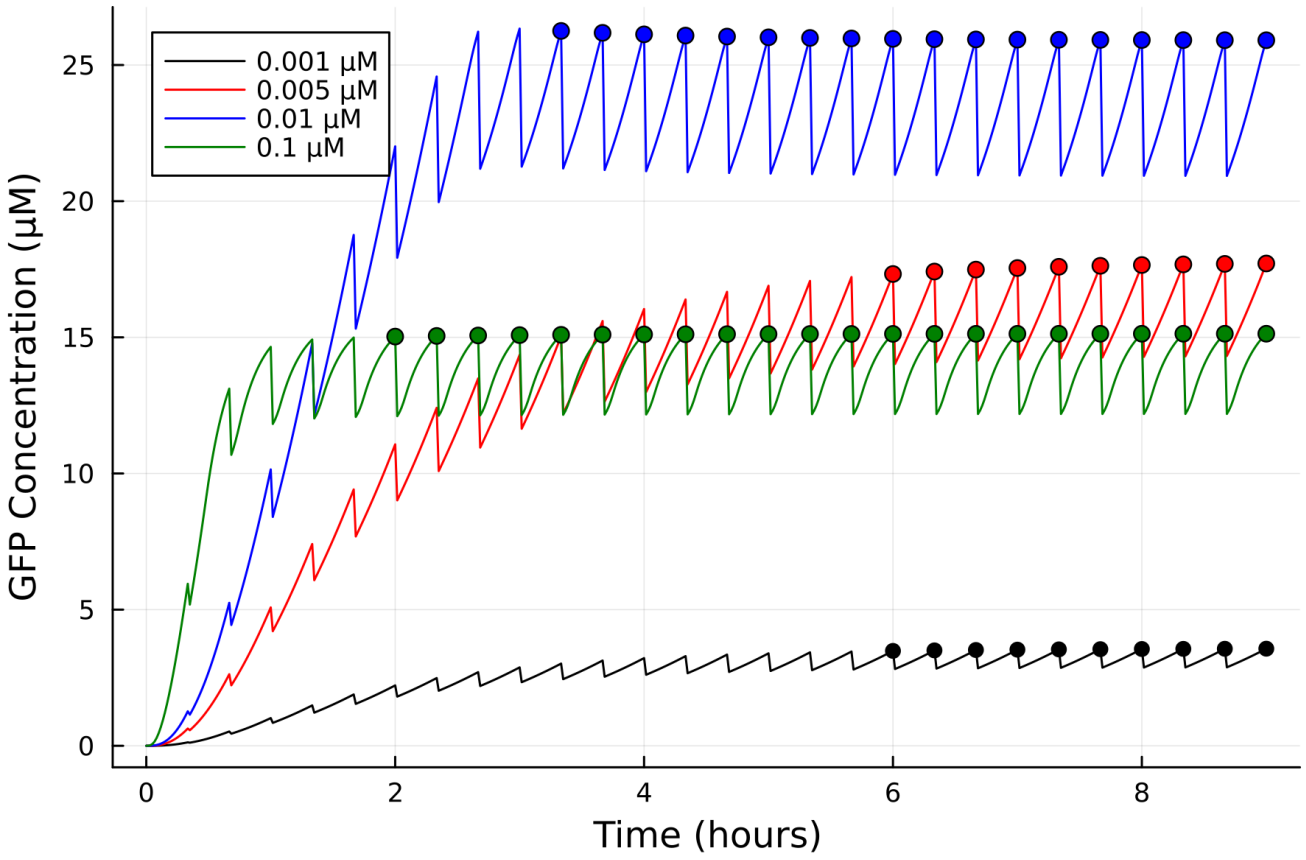


Figure A.10: Figure showing the concentration of mature GFP in the system over time at DNA input concentrations of D_0 = 0.001 µM (black), D_0 =0.005 µM (red), D_0=0.01 (blue) and D_0=0.1 µM (green) with periods of steady state shown with circles, starting at 6.00 h, 6.00 h, 3.33 h and 2.00 h respectively. High DNA concentrations reach steady state faster, because energy is redirected into transcription rather than translation and only small concentrations of GFP can be produced.

Bibliography

[1] Evangelos-Marios Nikolados, Andrea Y Weiße, and Diego A Oyarzún. Prediction of cellular burden with host–circuit models. *Synthetic Gene Circuits: Methods and Protocols*, pages 267–291, 2021.

[2] Erik D Carlson, Rui Gan, C Eric Hodgman, and Michael C Jewett. Cell-free protein synthesis: applications come of age. *Biotechnology advances*, 30(5):1185–1194, 2012.

[3] Nicole E Gregorio, Max Z Levine, and Javin P Oza. A user's guide to cell-free protein synthesis. *Methods and protocols*, 2(1):24, 2019.

[4] Yoshihiro Shimizu, Akio Inoue, Yukihide Tomari, Tsutomu Suzuki, Takashi Yokogawa, Kazuya Nishikawa, and Takuya Ueda. Cell-free translation reconstituted with purified components. *Nature biotechnology*, 19(8):751–755, 2001.

[5] Ashty S Karim, Fungmin Liew, Shivani Garg, Bastian Vögeli, Blake J Rasor, Aislinn Gonnot, Marilene Pavan, Alex Juminaga, Séan D Simpson, Michael Köpke, et al. Modular cell-free expression plasmids to accelerate biological design in cells. *Synthetic Biology*, 5(1):ysaa019, 2020.

[6] Amin SM Salehi, Mark Thomas Smith, Anthony M Bennett, Jacob B Williams, William G Pitt, and Bradley C Bundy. Cell-free protein synthesis of a cytotoxic cancer therapeutic: Onconase production and a just-add-water cell-free system. *Biotechnology journal*, 11(2):274–281, 2016.

[7] G. Kanter, J. Yang, A. Voloshin, S. Levy, J. R. Swartz, and R. Levy. Cell-free production of scfv fusion proteins: an efficient approach for personalized lymphoma vaccines. *Blood*, 109(8):3393–9, 2007.

[8] Naoki Goshima, Yoshifumi Kawamura, Akiko Fukumoto, and Nobuo Nomura. Human protein factory for converting the transcriptome into an in vitro–expressed proteome. *Nature Methods*, 5(12):1011–1017, 2008.

[9] Vaishali Thaore, Dimitrios Tsourapas, Nilay Shah, and Cleo Kontoravdi. Techno-economic assessment of cell-free synthesis of monoclonal antibodies using cho cell extracts. *Processes*, 8(4):454, 2020.

[10] Jan Müller, Martin Siemann-Herzberg, and Ralf Takors. Modeling cell-free protein synthesis systems—approaches and applications. *Frontiers in bioengineering and biotechnology*, 8:584178, 2020.

[11] Mathilde Koch, Jean-Loup Faulon, and Olivier Borkowski. Models for cell-free synthetic biology: make prototyping easier, better, and faster. *Frontiers in bioengineering and biotechnology*, 6:182, 2018.

[12] Iyappan Kathirvel and Neela Gayathri Ganesan. Computational strategies to enhance cell-free protein synthesis efficiency. *BioMedInformatics*, 4(3):2022–2042, 2024.

[13] Tobias Stögbauer, Lukas Windhager, Ralf Zimmer, and Joachim O. Rädler. Experiment and mathematical modeling of gene expression dynamics in a cell-free system. *Integrative Biology*, 4(5):494–501, 04 2012.

[14] Sabine Arnold, Martin Siemann-Herzberg, Joachim Schmid, and Matthias Reuss. Model-based inference of gene expression dynamics from sequence information. *Biotechnology for the Future*, pages 89–179, 2005.

[15] Anne Doerr, Elise De Reus, Pauline Van Nies, Mischa Van der Haar, Katy Wei, Johannes Kattan, Aljoscha Wahl, and Christophe Danelon. Modelling cell-free rna and protein synthesis with minimal systems. *Physical biology*, 16(2):025001, 2019.

[16] Wei-Hua Chen, Guanting Lu, Peer Bork, Songnian Hu, and Martin J Lercher. Energy efficiency trade-offs drive nucleotide usage in transcribed regions. *Nature communications*, 7(1):11334, 2016.

[17] Kathleen M Fisher. A misconception in biology: Amino acids and translation. *Journal of research in Science Teaching*, 22(1):53–62, 1985.

[18] Sabine Arnold, Martin Siemann, Kai Scharnweber, Markus Werner, Sandra Baumann, and Matthias Reuss. Kinetic modeling and simulation of in vitro transcription by phage t7 rna polymerase. *Biotechnology and bioengineering*, 72(5):548–561, 2001.

[19] Jonghyeon Shin and Vincent Noireaux. Efficient cell-free expression with the endogenous e. coli rna polymerase and sigma factor 70. *Journal of biological engineering*, 4:1–9, 2010.

[20] Kenneth A Johnson and Roger S Goody. The original michaelis constant: translation of the 1913 michaelis–menten paper. *Biochemistry*, 50(39):8264–8269, 2011.

[21] Barbora Lavickova, Nadanai Laohakunakorn, and Sebastian J Maerkl. A partially self-regenerating synthetic cell. *Nature Communications*, 11(1):6340, 2020.

[22] Henrike Niederholtmeyer, Viktoria Stepanova, and Sebastian J Maerkl. Implementation of cell-free biological networks at steady state. *Proceedings of the National Academy of Sciences*, 110(40):15985–15990, 2013.

[23] Philipp Städter, Yannik Schälte, Leonard Schmiester, Jan Hasenauer, and Paul L Stapor. Benchmarking of numerical integration methods for ode models of biological systems. *Scientific reports*, 11(1):2696, 2021.

[24] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[25] Kenneth V. Price. *Differential Evolution*, pages 187–214. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[26] Uri M Ascher, Steven J Ruuth, and Brian TR Wetton. Implicit-explicit methods for time-dependent partial differential equations. *SIAM Journal on Numerical Analysis*, 32(3):797–823, 1995.

[27] Yihai Yu. *Stiff problems in numerical simulation of biochemical and gene regulatory networks*. PhD thesis, University of Georgia, 2004.

[28] The JuliaLang Authors. Callback functions in julia, 2025. Accessed: 2025-04-22.

[29] Julio R Banga. Optimization in computational systems biology. *BMC systems biology*, 2:1–7, 2008.

[30] Ole Sigmund and Joakim Petersson. Numerical instabilities in topology optimization: a survey on procedures dealing with checkerboards, mesh-dependencies and local minima. *Structural optimization*, 16:68–75, 1998.

[31] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[32] Steven G. Johnson. Quadgk.jl: Adaptive gauss–kronrod integration in julia. `https://github.com/JuliaMath/QuadGK.jl`, 2022. Julia package version 2.4 or later.

[33] Deborah Hughes-Hallett, Andrew M Gleason, Patti Frazer Lock, and Daniel E Flath. *Applied calculus*. John Wiley & Sons, 2021.

[34] Jessica G Gaines and Terry J Lyons. Variable step size control in the numerical solution of stochastic differential equations. *SIAM Journal on Applied Mathematics*, 57(5):1455–1484, 1997.

[35] Arjang Hassibi, Haris Vikalo, and Ali Hajimiri. On noise processes and limits of performance in biosensors. *Journal of applied physics*, 102(1), 2007.

[36] Lorenz M Mayr and Peter Fuerst. The future of high-throughput screening. *SLAS Discovery*, 13(6):443–448, 2008.

[37] Patrick Kofod Mogensen and Asbjørn Nilsen Riseth. Optim: A mathematical optimization package in Julia. *Journal of Open Source Software*, 3(24):615, 2018.

[38] Haixin Wang, Lijun Qian, and Edward R Dougherty. Steady-state analysis of genetic regulatory networks modeled by nonlinear ordinary differential equations. In *2009 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 182–185. IEEE, 2009.

[39] Desmond J Higham. The tolerance proportionality of adaptive ode solvers. *Journal of Computational and Applied Mathematics*, 45(1-2):227–236, 1993.

[40] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[41] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[42] Jorge J Moré and Danny C Sorensen. Newton's method. Technical report, Argonne National Lab.(ANL), Argonne, IL (United States), 1982.

[43] Ronald Schoenberg. Optimization with the quasi-newton method. *Aptech Systems Maple Valley WA*, pages 1–9, 2001.

[44] Michael Innes, Alan Edelman, Chris Fischer, Christopher Rackauckas, Elias Saba, Viral B Shah, and Dhairya Tebbutt. Don't unroll adjoint: Differentiating ssa-form programs. *arXiv preprint arXiv:1810.07951*, 2019.

[45] Jarrett Revels, Miles Lubin, and Theodore Papamarkou. Forward-mode automatic differentiation in Julia. *arXiv preprint arXiv:1607.07892*, 2016.

[46] Arkadi S Nemirovski and Michael J Todd. Interior-point methods for optimization. *Acta Numerica*, 17:191–234, 2008.

[47] Brian Beavis and Ian M Dobbs. *Optimization and stability theory for economic analysis*. Cambridge university press, 1990.

[48] Osman Güler. Barrier functions in interior point methods. *Mathematics of Operations Research*, 21(4):860–885, 1996.

[49] John A Nelder and Roger Mead. A simplex method for function minimization. *The computer journal*, 7(4):308–313, 1965.

[50] Saša Singer and John Nelder. Nelder-mead algorithm. *Scholarpedia*, 4(7):2928, 2009.

[51] Christian Blum, Raymond Chiong, Maurice Clerc, Kenneth De Jong, Zbigniew Michalewicz, Ferrante Neri, and Thomas Weise. *Evolutionary Optimization*, pages 1–29. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[52] Dan Simon. *Evolutionary optimization algorithms*. John Wiley & Sons, 2013.

[53] Fritz Kindermann. Blackboxoptim.jl: Derivative-free optimization for julia. `https://github.com/robertfeldt/BlackBoxOptim.jl`. Accessed: 2025-05-04.

[54] Kenneth V Price, Rainer M Storn, and Jouni A Lampinen. The differential evolution algorithm. *Differential evolution: a practical approach to global optimization*, pages 37–134, 2005.

[55] James D Murray. *Mathematical biology: I. An introduction*, volume 17. Springer Science & Business Media, 2007.

[56] JuliaOpt. Particleswarmoptim.jl: Particle swarm optimization for julia, 2023. Accessed: 2025-04-27.

[57] Dongshu Wang, Dapei Tan, and Lei Liu. Particle swarm optimization algorithm: an overview. *Soft computing*, 22(2):387–408, 2018.

[58] Jared L Dopp, Yeong Ran Jo, and Nigel F Reuel. Methods to reduce variability in e. coli-based cell-free protein expression experiments. *Synthetic and Systems Biotechnology*, 4(4):204, 2019.

[59] Xumeng Ge, Dan Luo, and Jianfeng Xu. Cell-free protein expression under macromolecular crowding conditions. *PLoS One*, 6(12):e28707, 2011.