Numerical Optimization on Epidemiological SIR-models

Jonas Hjulstad

Abstract

This project is dedicated to exploring different methods of nonlinear optimization in the framework of CasADI in Python. Explicit Runge-Kutta 4 (ERK4) and collocation integrators are introduced and used to simulate trajectories for an epidemiological SIR-model, and ERK4 is used to perform sensitivity analysis on an epidemic. Optimal control objectives are defined to mitigate epidemic spread while keeping the costs of social distancing, vaccination and isolation of individuals low. The Primal-Dual Interior Point method (IPOPT), Pontryagin's Maximum Principle (PMP) and Sequential Quadratic Programming (SQP) are used to find optimal solutions for the defined objectives in combination with single and multiple-shooting methods. States, multipliers and objectives are illustrated over the solver iterations to show how the solvers converge. These results show that there are many local optimal solutions that can vary a lot depending on initial states, parameters and methods used. The custom designed primal-dual and PMP-methods struggle to converge for most objectives and configurations, while the IPOPT and QPOASES-solver are generally able to converge.

Contents

List of Figures

List of Tables

List of Algorithms

1 Epidemiological Model

1.1 SIR

Due to simple dynamics, the population subject to disease spread is divided into three groups: Susceptible (S), Infected (I) and Recovered(R), yielding the Kermack-Mckendrick model (SIR):

$$\begin{aligned} \dot{S} &= -\beta \frac{SI}{N_{pop}} \\ \dot{I} &= \beta \frac{SI}{N_{pop}} - \alpha I \\ \dot{R} &= \alpha I \end{aligned} \tag{1}$$

Where β is the infection rate in the population, α is the recovery rate and $N_{pop} = S + I + R$ is the total population.

With respect to optimal control of epidemics both β and α can be controllable depending on the situation. α is determined by the expected recovery time of the population, which may vary depending on the treatment given to the infected. Modeling of the COVID-19 pandemic is usually done by dividing these three groups into multiple subgroups, yielding more complex models which can be fitted to data. However, the work in this project will only be considering the SIR-model with fixed recovery rate α.

The main dynamic of interest is the nonlinear dynamic between the susceptible and infected group, which can be controlled by adjusting β, or in terms of the expected number of infections per individual, $\mathscr{R}_0$:

$$\mathscr{R}_0 = \frac{\beta N}{\alpha} \tag{2}$$

1.2 Equilibrium Points

The number of equilibrium points of the system depends on the reproduction number. When $\mathscr{R}_0 < 1$ the disease does not spread fast enough to beat the recovery rate, and will eventually die out.

$$\mathscr{E}_0 = (S_\infty, 0, R_\infty) \tag{3}$$

This is the disease-free equilibrium, which is the optimal goal for cumulative infection-minimizing control strategies. In the case of a higher reproduction number, another equlilibrium point occurs through bifurcation (Figure 1).

With $\mathscr{R}_0 > 1$ the disease-free equilibrium is unstable. An outbreak will cause temporary, locally unstable dynamics which will stabilize as the number of susceptibles decrease. The system will converge towards the endemic equilibrium. This equilibrium maximizes the cumulative number of infections.

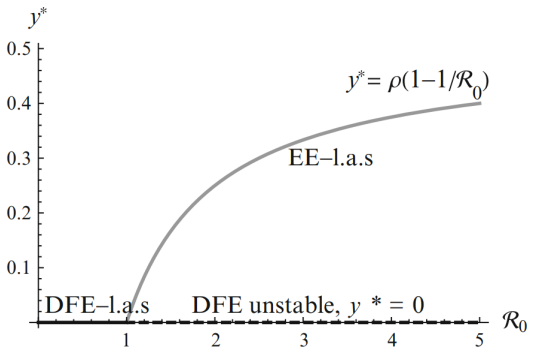

Figure 1: Bifurcation-diagram for a dimensionless SIR-model. (y^* is the dimensionless number of infected) [Martcheva 2015]

1.3 Stability

The local stability in each state can be analyzed with the jacobian of the differential equations:

$$J = \begin{bmatrix} -\frac{\beta I}{N} & \frac{\beta S}{N} & 0 \\ \frac{\beta I}{N} & \frac{\beta S}{N} - \alpha & 0 \\ 0 & \alpha & 0 \end{bmatrix} \tag{4}$$

The three eigenvalues of the jacobian are $\lambda_0 = 0$ and two complex values given by:

$$\lambda_{1,2} = \frac{\beta(S-I) - \alpha N}{2N} \pm K \tag{5}$$

$$\tag{6}$$

The unstable dynamics is given by the real part of the complex eigenvalues, and can be represented by the reproduction number:

$$Re[\lambda_{1,2}] > 0 \tag{7}$$

$$\frac{\mathscr{R}_0(S-I)}{N} > 1 \tag{8}$$

The most unstable eigenvalue occurs when $\mathscr{R}_0 = \mathscr{R}_{0,max}$ and $\max_{S,I}(S-I)$. Since S is monotonically decreasing, this will occur at $(S, I, R) = (S_0, I_0, R_0)$.

1.4 Control strategies

Reduction of the number of infected can be achieved using different strategies, including vaccination, quarantine and social distancing measurements. The different types of mitigation will affect the groups in the SIR model differently.

1.5 Social Distancing

In addition to minimization of the total number of infections, a penalty to the strictness of the social distancing policies can be introduced, resulting in the following minimization problem:

$$\min_{w} \Phi(w) = \min_{u_0=[\mathscr{R}_{0,0},\ldots,\mathscr{R}_{0,N-1}]} \sum_{k=0}^{N-1} \frac{I_{k+1}}{N_{pop}} - W_u \frac{(u_{0,k})^2}{(u_{0,max} - u_{0,min})^2} \tag{9}$$

The constant relationship between N_{pop}, α and β makes $\mathscr{R}_0$ and β only differ in scaling, which is the reason to why u is set to $\mathscr{R}_0$ for the social distancing control strategy. It should be noted that basic reproduction number $\mathscr{R}_0$ is viewed as the infection potential for one individual when the whole population is susceptible, which is not to be confused with the effective reproduction number:

$$R_{eff} = \frac{\mathscr{R}_0 I}{N_{pop}} \tag{10}$$

1.6 Vaccination

Vaccination can be implemented as a flow rate from the susceptible to the recovered group.

$$\dot{S} = -\beta \frac{SI}{N_{pop}} - uS \tag{11}$$

$$\dot{R} = \alpha I + uS \tag{12}$$

Increased vaccination will be represented as a positive contribution to the objective function.

$$\min_{w} \Phi(w) = \min_{u=[u_0,\ldots,u_{N-1}]} \sum_{k=0}^{N-1} I_k + W_u u_k \tag{13}$$

1.7 Isolation

Putting infected individuals in quarantine eliminates their potential infectiousness, and can be viewed as transfering them over to the recovered group.

$$\dot{I} = \beta \frac{SI}{N_{pop}} - (\alpha + u)I \tag{14}$$

$$\dot{R} = (\alpha + u)I \tag{15}$$

Increased quarantine measures will be represented as a positive contribution to the objective function.

$$\min_{w} \Phi(w) = \min_{u=[u_0,\ldots,u_{N-1}]} \sum_{k=0}^{N-1} I_k + W_u u_k \tag{16}$$

Quarantine rate range is more difficult to estimate due to underreporting, self-isolation and asymptomatic individuals.

1.8 Problem Parameters

For social distancing the reproduction number will be lower-bounded by the lowest estimated value from FHI's national report on the spread of Covid-19 in Norway (03/02-21,[Birgitte Freiesleben de Blasio 2021]).$\mathscr{R}_0$ will be upper bounded approximately to one of the highest reproduction number estimates for countries worldwide for Covid-19 ([Billah et al. 2020]). α will be fixed according to the expected time spent in the infectious class[Cevik et al. 2020].

$$\alpha = \frac{1}{9[\text{days}]} \approx 0.11 \tag{17}$$

$$\mathscr{R}_0 \in [0.5, 6.5] \tag{18}$$

The population size is set approximately to Norways total population, and the initial number of infected is set to capture the endemic transition over the course of a year. The full parameter set is available in the appendix.

Note: The purpose of the parameters is to give trajectory simulations for a system that is interpretable and relevant, this is not an attempt at modeling a true population in the Covid-19 pandemic. The assumption of homogenous population dynamics will most likely result in a bad approximation of the Norwegian population, so the challenges with system idenitification and verification are left out of this project.

1.9 Uncontrolled System

Figure 2 shows the epidemic trajectory with maximum reproduction number (uncontrolled scenario).

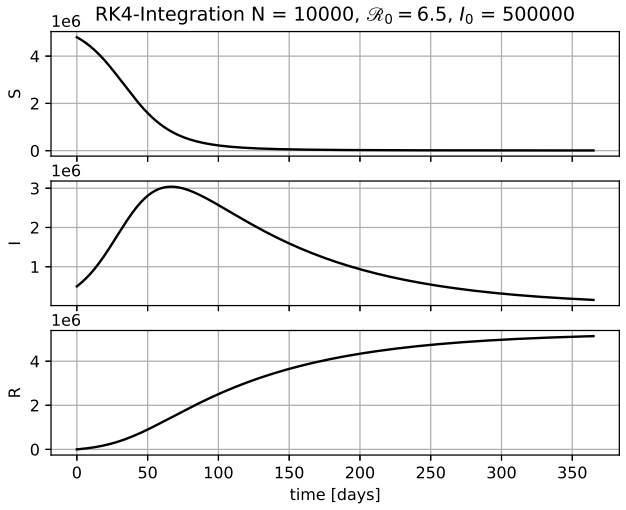

Figure 2: Uncontrolled SIR-model (N - RK4 steps per day)

The rather long contol horizon of a year is included to provide long-term control strategies that do not only postpone an endemic situation.

2 Integrators: Explicit Runge-Kutta 4

A common explicit RK4-scheme is given as:

$$\dot{x} = F(x, u) \tag{19}$$

$$x_{k+1} = x_k + \frac{h}{6}k_1 + 2k_2 + 2k_3 + k_4 \tag{20}$$

$$\tag{21}$$

With k's given by:

$$k_1 = F(x_k, u_k) \tag{22}$$

$$k_2 = F(x_k + \frac{h}{2}k_1, u_k) \tag{23}$$

$$k_3 = F(x_k + \frac{h}{2}k_2, u_k) \tag{24}$$

$$k_4 = F(x_k + hk_3, u_k) \tag{25}$$

$$\tag{26}$$

Yielding the following buther-tableau:

0				
$\frac{1}{2}$	$\frac{1}{2}$			
$\frac{1}{2}$	0	$\frac{1}{2}$		
1	0	0	1	
	$\frac{1}{3}$	$\frac{1}{6}$	$\frac{1}{6}$	$\frac{1}{3}$

Table 1: RK4 butcher tableau

Figure 3 shows the stability region of explicit runge-kutta methods.

The bilinear SIR-model will have unstable regions which require the integrator step length to be set sufficiently low in order to get accurate steps. Performing a sufficient number M of ERK4 steps per control step is necessary.

2.1 Sensitivities

2.1.1 Variational Approach

General state and input sensitivities can be calculated from partial derivatives of the ODE:

$$\dot{A}(t) = \frac{\partial F}{\partial x}|_{x(t),u_k} A(t), \qquad A(t_k) = I \tag{27}$$

$$\dot{B}(t) = \frac{\partial F}{\partial x}|_{x(t),u_k} B(t) + \frac{\partial F}{\partial u}|_{x(t),u_k}, \qquad B(t_k) = 0 \tag{28}$$

Where the ODE for state sensitivity $A(t)$ contains one term, while ODE for input sensitivity $B(t)$ contains two terms due to u_k's indirect impact on $F(t, x, u)$

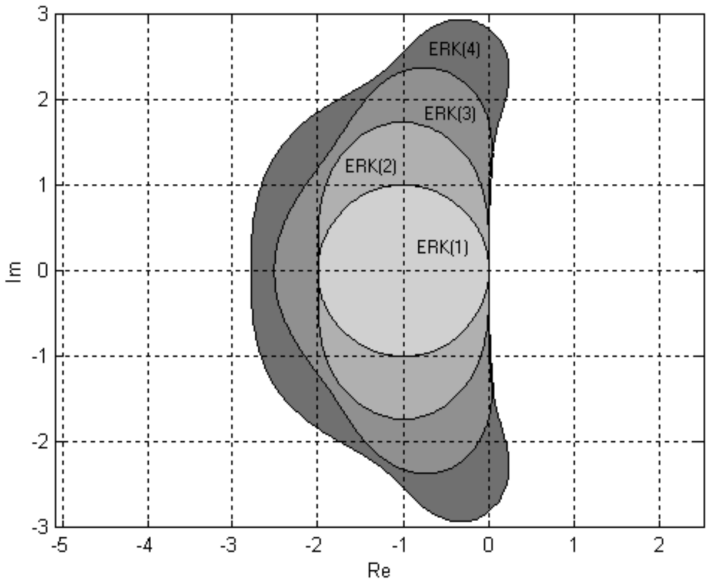

Figure 3: Runge-Kutta stability regions[Egeland and Gravdahl 2002]

through x. Implementing sensitivities using the variational approach will yield additional integration errors in the extra ODEs (compared to algorithmic differentiation).

2.1.2 Algorithmic Differentiation

In algorithmic differentiation, the problem is first discretized, then the sensitivities are obtained from the resulting algorithm. This effectively limits the integration error of the sensitivities to some magnitude of the state integration error. The sensitivity equations can be derived using equations 27 and 28. With

Algorithm 1: RK4 Integration Algorithm

Data: x_k, u_k, h, N
for $i = 0 : N-1$ **do**
  $k_1 = F(x_k, u_k)$
  $k_2 = F(x_k + \frac{h}{2}k_1, u_k)$
  $k_3 = F(x_k + \frac{h}{2}k_2, u_k)$
  $k_4 = F(x_k + hk_3, u_k)$
  $x_{k+1} = x_k + \frac{h}{6}k_1 + 2k_2 + 2k_3 + k_4$
end

full state trajectories provided, the sensitivities can be calculated afterwards ac-

cording to algorithm 2.

Algorithm 2: RK4 Sensitivity Calculation

Data: $[x_0, \ldots, x_N], [u_0, \ldots, u_N], h, N$
for $i = 0 : N - 1$ **do**
  $C_x = \frac{\partial}{\partial x}(\frac{h}{6}k_1 + 2k_2 + 2k_3 + k_4)|_{x_k, u_k}$
  $C_u = \frac{\partial}{\partial u}(\frac{h}{6}k_1 + 2k_2 + 2k_3 + k_4)|_{x_k, u_k}$
  $A_{k+1} = (I + C_x)A_k$
  $B_{k+1} = (I + C_x)B_k + C_u B_k$
end

Here, the evaluation points are updated with the integrated state, which results in different sensitivity errors than the variational approach. For the upcoming SIR-model, the algorithm will be implemented to yield sensitivities iteratively with the integration.

2.1.3 Implementation on the SIR-model

This implementation use social distancing as control input with $u = \beta$. Both approaches require partial derivatives of the SIR-ODE:

$$\frac{\partial F}{\partial x} = \frac{\partial \begin{bmatrix} \frac{-ux_1x_2}{N_{pop}} \\ \frac{ux_1x_2}{N_{pop}} - \alpha x_2 \\ \alpha x_2 \end{bmatrix}}{\partial x} = \begin{bmatrix} -\frac{ux_2}{N_{pop}} & -\frac{ux_1}{N_{pop}} & 0 \\ \frac{ux_2}{N_{pop}} & \frac{ux_1}{N_{pop}} - \alpha & 0 \\ 0 & \alpha & 0 \end{bmatrix} \tag{29}$$

$$\frac{\partial F}{\partial u} = \frac{\partial \begin{bmatrix} \frac{-ux_1x_2}{N_{pop}} \\ \frac{ux_1x_2}{N_{pop}} - \alpha x_2 \\ \alpha x_2 \end{bmatrix}}{\partial u} = \begin{bmatrix} \frac{-x_1x_2}{N_{pop}} \\ \frac{x_1x_2}{N_{pop}} \\ 0 \end{bmatrix} \tag{30}$$

Using equations 29 and 30, the two approaches for sensitivity were implemented with the RK4 scheme in Python.

The susceptible and infected sensitivities with respect to the susceptible and infected populations (plots $[1:2, 1:2]$) show that the most fragile timepoints of the epidemic occurs when the product of susceptibles and infected are highest. The third column shows sensitivities with respect to the recovered individuals, which shows that the recovered individuals do not impact the epidemic.

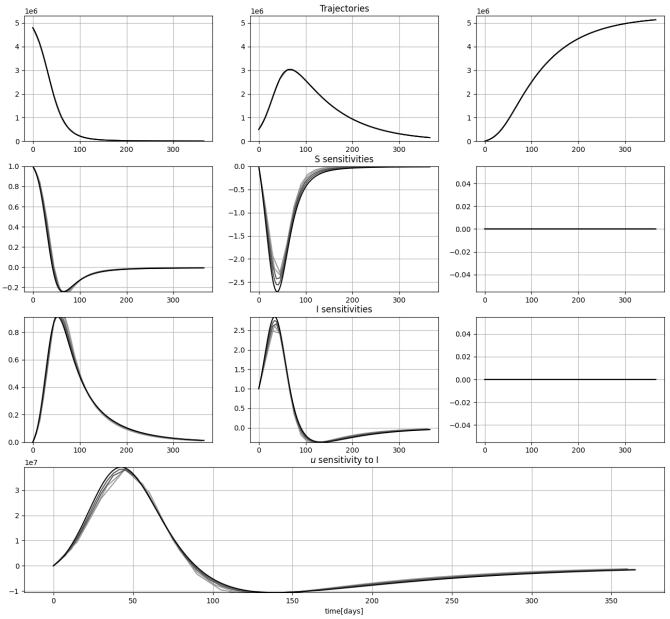

Figure 4: RK4-scheme sensitivities and trajectories with Δt linearly spaced between $[15, 0.01]$, gradually from gray to black.

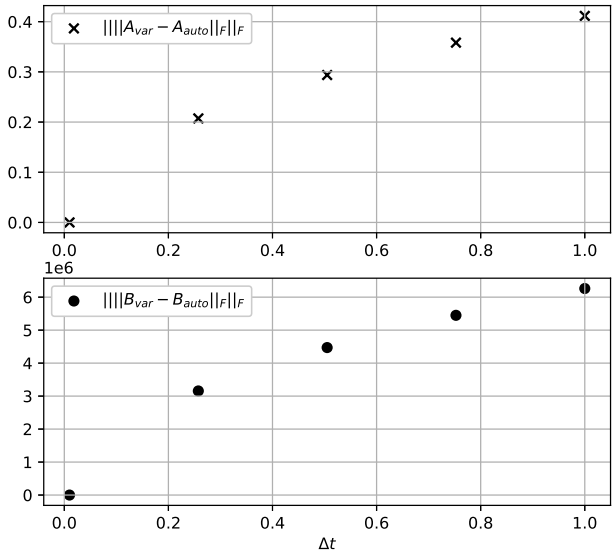

Figure 5: Frobenius-norm error between variational and autodifferentiation sensitivity evaluations

2.2 Integrators: Collocation

A collocation integrator constructs the trajectory using parameters θ on a time grid of $d+1$ points in $[t_k, t_{k+1})$ ($[\tau_0, \tau_1, \ldots, \tau_d]$). Parameters θ are constrained to equal the value of the trajectory at these points, giving the full trajectory as a sum of lagrange polynomials:

$$P_i(\tau) = \prod_{j=0, j\neq i}^{d+1} \frac{\tau - \tau_j}{\tau_i - \tau_j} \tag{31}$$

$$s_k(\theta, \tau) = \sum_{j=0}^{d+1} \theta_j P_j(\tau) \tag{32}$$

Matrices $C \in \mathbb{R}^{(d+1)\times(d+1)}$ and $B, D \in \mathbb{R}^{(d+1)}$ can be constructed, respectively containing coefficients for the derivatives and the end state for the polynomials. This makes it possible to calculate $\boldsymbol{\theta}$, $\dot{\boldsymbol{\theta}}$ and the objective value accumulated over each timestep t_k. Figure 6 illustrates the properties of the coefficients.

The derivative of the trajectory can be constrained to match the ODE in order to approximate the true trajectory at the chosen timepoints:

$$g_{\text{col}} = \dot{s}_k(\theta, \tau) - F(\theta_i, t_k + \tau_i) = \sum_{j=0}^{d+1} \theta_j \dot{P}_j(t_k + \tau_i) - F(\theta_i, t_k + \tau_i) = 0 \tag{33}$$

$\forall k, i \in [0, \ldots, N-1], [0, \ldots d]$

2.2.1 Sensitivities

$g_{col} = 0$ can be solved with newton-raphson, yielding $\frac{\partial g_{col}}{\partial \theta}$ as a biproduct. The sensitivities of the integrator with respect to state and control input can be derived from the implicit function theorem, yielding:

$$\frac{\partial \theta}{\partial x} = \frac{\partial g_{col}}{\partial \theta}^{-1} \frac{\partial g_{col}}{\partial x} \qquad \frac{\partial \theta}{\partial u} = \frac{\partial g_{col}}{\partial \theta}^{-1} \frac{\partial g_{col}}{\partial u} \tag{34}$$

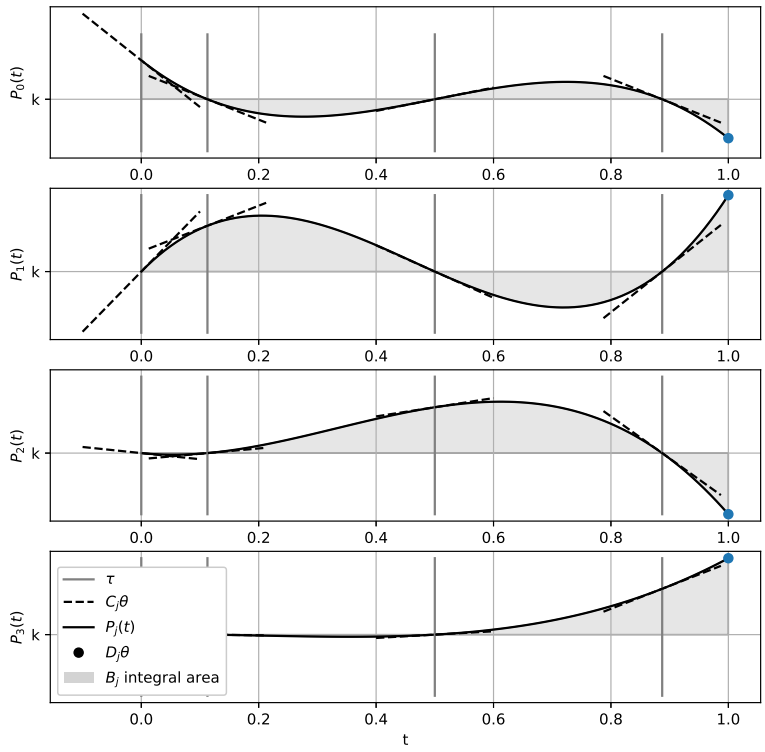

Figure 6: Lagrange Polynomials using legendre-timepoints for $d = 3$

3 Single Shooting

The optimization problem can be formulated as an nlp with the system dynamics as constraints.

$$\begin{aligned} \min_{w} \quad & \Phi(w) \\ \text{s.t.} \quad & x_f = f(x_0, [u_0, \ldots, u_{N-1}]), \\ & u_{max} \geq u_k \geq u_{min}, \\ & P(I_{k+1} \geq i_{max,k+1}) \leq k_I \end{aligned}$$

Where $f(.)$ is the trajectory from initial to end state. The intermediate states of the system is not visible to the solver. $f(.)$ can be implemented as an iterative RK4-scheme:

Algorithm 3: Single-shooting problem construction and integration

Data: $x_k = x_0, u = [u_0, \ldots, u_{N-1}], h, N$
define f as a RK4-integrator repeated M times
for $k = 0 : N-1$ **do**
  $x_k, Q_k = RK4(@SIR, @Cost, x_k, u_k, h)$
  $J+ = Q_k$
  add $u_{min} - u_k, u_k - u_{max}$ to bounds
  set $u_{k,0}$
end
return x_k, J, U_0, *bounds*

The resulting variables, values and boundaries can be directly inserted into the CasADI-interface if derived symbolically, or directly intefaced with a solver when derived numerically.

3.1 Primal-Dual Interior Point Formulation

3.1.1 KKT Conditions

The inequalities can be replaced using the log-barrier method:

$$\begin{aligned} \min_{w_\tau} \quad & \Phi(w_\tau) - \tau \sum_{k=0}^{N-1} (\log(u_k - u_{min}) + \log(u_{max} - u_k)) \\ \text{s.t.} \quad & x_f = f(x_0, [u_0, \ldots, u_{N-1}]) \end{aligned}$$

$$h(w_\tau) = \begin{bmatrix} u_0 - u_{min} \\ u_{max} - u_0 \\ \vdots \\ u_{max} - u_{N-1} \end{bmatrix} \qquad \nu = -\tau \begin{bmatrix} h_0^{-1} \\ \vdots \\ h_{2(N-1)}^{-1} \end{bmatrix} \tag{35}$$

Which gives the Primal-Dual KKT-conditions:

$$\begin{aligned} \nabla\Phi(w) + \nabla h(w)\nu + \nabla f(x_0, [u_0, \dots, u_{N-1}])\lambda = 0 \\ g(w) = x_f - f(x_0, [u_0, \dots, u_{N-1}]) = 0 \\ \nu_i h_i(w) + \tau = 0 \\ h(w) < 0, \nu > 0 \end{aligned} \tag{36}$$

The complementarity slackness in the Primal-Dual method approximates the constraints for the original formulation, and will converge to them for small τ. The current formulation requires $h(w) < 0$, which can be relaxed with slack variable s. Furthermore the problem can be expressed in terms of true KKT conditions by setting $\nu = \mu$, which upper bounds the optimum error $|\mu^* - \nu^*|$ linearly with τ. This results in equation 37, where the equalities can be used for Newton-Rhapson steps.

$$\begin{aligned} \nabla\Phi(w) + \nabla h(w)\nu + \nabla f(x_0, [u_0, \dots, u_{N-1}])\lambda = 0 \\ g(w) = 0 \\ h_i(w) + s_i = 0 \\ \mu_i s_i - \tau = 0 \\ s > 0, \mu > 0 \end{aligned} \tag{37}$$

3.1.2 Line Search

In the case where the end-state of the trajectory is unconstrained, $g(w)$ can be omitted. The solution for each KKT problem can be found by using the Newton-Rhapson method on the KKT conditions.

$$r(w, \mu, \tau) = \begin{bmatrix} \nabla\mathcal{L}(w, \mu) \\ h(w) + s \\ \mu_i s_i - \tau \end{bmatrix} = 0 \tag{38}$$

$$\begin{bmatrix} d^w \\ d^\lambda \\ d^\mu \end{bmatrix} = \nabla r(w, \mu, \tau)^{-1} r(w, \mu, \tau) \tag{39}$$

It is not possible to guarantee invertibility of r for the SIR-model due to nonlinear dynamics. [Wächter and Biegler 2006] suggests ensuring invertibility by adding a diagonal term δ_w to the hessian of the lagrangian $\nabla_{ww}\mathcal{L}(w, \mu)$ in $\nabla r(w, \mu, \tau)$.

The steps can be constrained to decrease $r(w, \mu, \tau)$ at every iteration with the fraction-to-the-boundary rule:

$$\alpha_k^w = \max\{\alpha \in (0, 1] : w_k + \alpha d_k^w \geq (1 - \tau_j)x_k\} \tag{40}$$

$$\alpha_k^\mu = \max\{\alpha \in (0, 1] : \mu_k + \alpha d_k^\mu \geq (1 - \tau_j)\mu_k\} \tag{41}$$

$$\tag{42}$$

Where w and λ are updated according to α_k^w, and μ according to α^μ.

Armijo's backtracking line search ensures that the step in the quadratic approximated direction leads to descent.

Algorithm 4: Newton steps with Armijo's backtracking line search

Data: $w, t = 1, \alpha \leq \frac{1}{2}$
while $||\Phi(w + \Delta w)|| > ||\Phi(w) + \alpha t \nabla \Phi(w)^T \Delta w||$ **do**
  $t = \beta_a t$
end
return $w + t\alpha\Delta w$

Newton iterations are terminated when states are considered converged, $w_k - w_{k-1} < \Delta w_{tol}$.

3.2 Sequential Quadratic Programming Formulation

The problem can be approximated using the hessian of the lagrangian combined with the gradient of the objective and active constraints:

$$\mathcal{L}(x, \lambda, \mu) = \nabla\Phi(w) - \lambda\nabla_w g(w) - \mu\nabla_w h(w) \tag{43}$$

$$\begin{aligned} \min_d \quad & \frac{1}{2} d^T \nabla^2_{ww} \mathcal{L}(w) d + \nabla\Phi(w)^T d \\ \text{s.t.} \quad & \nabla g(w)^T d + g(w) = 0, \\ & \nabla h(w)^T d + h(w) \leq 0 \end{aligned} \tag{44}$$

3.2.1 Active Set Strategy

After determining the current active inequality constraints, the the sequential step and multipliers can be determined from the Karush-Kuhn-Tucker conditions of the problem.

$$\begin{bmatrix} \nabla_{ww}\mathcal{L}(w, \lambda, \mu) & \nabla g(w) & \nabla h_A(w) \\ \nabla g(w)^T & 0 & 0 \\ \nabla h_A(w)^T & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \\ \mu_A \end{bmatrix} = - \begin{bmatrix} \Phi(w) \\ g(w) \\ h_A(w) \end{bmatrix} \tag{45}$$

In the SIR-problem formulation we have the following boundary inequalities:

$$h(w) = \begin{bmatrix} u_0 - u_{min} \\ u_{max} - u_0 \\ \vdots \\ u_{max} - u_{N-1} \end{bmatrix} \qquad \nabla h(w) = \begin{bmatrix} 1 \\ -1 \\ \vdots \\ 1 \\ -1 \end{bmatrix} \tag{46}$$

Determining the active set for boundaries is simpler than for other inequalities, since they come in pairs where only one can be active at each iteration.

In the case of independent inequality constraints, the active set has to be determined by testing different steps with different active inequalities until positive multipliers and a feasible step has been achieved.

3.2.2 Line Search with Inequalities

Newtons method minimizes the quadratic approximation of objective $\Phi(w)$. Due to the discontinuity of inequality constraints, any $\Phi(w)$ subject to them can be solved by imposing newton steps on sequential approximation of $\Phi(w)$. In order to ensure that the objective converges within the inequality constraints, the line search can be performed on a T_1 merit function.

$$T_1(w) = \Phi(w) + \nu \sum_{i=1}^{m} \max(0, h_i(w)) \tag{47}$$

Algorithm 5: SQP Line Search Algorithm for inequalities

Data: $w, \mu, g = []$
while $||\nabla \mathcal{L}||$ *or* $\max(0, h) \geq tol$ **do**
  Formulate SQP (44); Solve for Δw (equation 45, $h_A = h$); Perform line search with merit function to update w, μ; (equation 4, $\Phi = T_1$)
end
return w, μ

Algorithm 6: Secure introspection

Data: $w, \mu, g = []$
while $||\nabla \mathcal{L}||$ *or* $\max(0, h) \geq tol$ **do**
  Formulate SQP (44); Solve for Δw (equation 45, $h_A = h$); Perform line search with merit function to update w, μ; (equation 4, $\Phi = T_1$)

3.3 IPOPT with CasADI's Symbolical Framework

A single-shooting optimization problem was formulated with CasADI using the symbolical framework, for all three control strategies. Algorithm 7 summarizes the basic loop for converging without δ or step-corrections.

Algorithm 7: Primal-Dual Interior Point with CasADI symbolical framework

Data: $w = w_0 \in \mathbb{R}^{N_u + N_\mu + N_s}, \tau = 1$
while $(\tau > \tau_{tol})$ *and* $(\Delta w > \Delta w_{tol})$ **do**
  while $w_k > w_{k,tol}$ **do**
    $w_k = w_k - \nabla r(w, \mu, \tau)^{-1} r(w, \mu, \tau)$
  end
  $\Delta w = w_k - w_{k,old}$
  $\tau = \beta_\tau \tau$
end
return w_k

3.4 Interfacing IPOPT in CasADI

3.4.1 Implementation

Single-shooting is implemented in CasADI using $X_0 \in \mathbb{R}^3, U \in \mathbb{R}^N$ as MX-variables. The ODE is integrated M times for each control interval, resulting in a total of $N \times M$ integration steps. Inequality constraints are passed as variable bounds, while equality constraints are passed as regular constraints to the solver.

Algorithm 8: Single-shooting with IPOPT

Data: $x_k = x_0, U = MX \in \mathbb{R}^N, J = 0, h, N$
Construct integrator $f(@SIR, x, u, h)$ with CasADI-symbolics
for $i = 0 : N - 1$ **do**
  $x_k, Q_k = f([@SIR, @Cost], x_k, U[i], h)$
  $J+ = Q_k$
  add $u_{min} - u_k, u_k - u_{max}$ to bounds
  set $u_{k,0}$
end
solver = nlpsol('solver', 'ipopt', {J, U})
sol = solver(U_0,bounds)

3.5 Interfacing QPOASES in CasADI

3.5.1 Implementation

Passing 'sqpmethod' to *nlpsol*(.) makes it possible to solve the same problem formulation as algorithm 8 (IPOPT and QPOASES are run from the same .py-script in the implementation).

4 Multiple Shooting

The state trajectory can be divided into multiple segments that are constrained to be connected. This gives the nlp additional constraints that can be temporarily violated in order to (possibly) achieve a faster convergence to a different local optimum.

$$\begin{aligned} \min_{w} \quad & \Phi(w) \\ \text{s.t.} \quad & f(x_0, u_0) - x_1 = 0, \\ & f(x_k, u_k) - x_{k+1} = 0, \\ & \vdots, \\ & f(x_{N-1}, u_{N-1}) - x_N = 0, \\ & u_{max} \geq u_k \geq u_{min} \end{aligned}$$

Where $f(.)$ is the trajectory from x_k to x_{k+1}. The RK4-integrator can be replaced by collocation constraints resulting in a direct collocation scheme.

4.1 Interfacing IPOPT in CasADI

4.1.1 Implementation

Algorithm 9: Multiple-Shooting problem construction and integration (RK4)

Data: $X = MX \in \mathbb{R}^{3\times(N+1)}, U = MX \in \mathbb{R}^{N}, h, N, M$
Construct integrator $f(@ODE, x, u, h)$ with CasADI-symbolics
add $X[:, 0] - x_0$ to constraints
for $i = 0 : N - 1$ **do**
  set $u_{k,0}$
  $x_k, Q_k = f(@ODE, @Cost, x_k, U[k], h)$
  $J+ = Q_k$
  add $u_{min} - u_k, u_k - u_{max}$ to bounds
  add $x_{min} - x_k, x_k - x_{max}$ to bounds
  add $x_k - X[:, k + 1]$ to constraints
  set $x_{k,0}$
end
solver = nlpsol('solver', 'ipopt', {J, [U,X]})
sol = solver($[U_0, X_0]$,bounds, constraints)

In order to get comparable performance to single shooting, the multiple shooting scheme is initialized with the same initial trajectory for the lifted states.

5 Pontryagin's Maximum Principle

Optimal control with PMP will consider the continous formulations of the control problems.

$$\begin{aligned} \min_{w} \quad & \Phi(w) \\ \text{s.t.} \quad & \dot{x}(t) = F(x(t), u(t)), \\ & u_{max} \geq u(t) \geq u_{min}, \quad \forall t \in [0, T) \end{aligned}$$

Where $F(.)$ is the SIR ODE-dynamics.

A hamiltonian can be defined as the 'lagrangian at each timestep'.

$$H(x, \lambda, u) = L(x(t), u(t)) + \lambda^T F(x(t), u(t)) \tag{48}$$

The objectives are either convex quadratic/linear with respect to u, and the constraints have linear relations to the control input for all control strategies. This makes it possible to determine optimal constrained control input u_c^*:

$$u(t) = \arg\min_{u} H(x, \lambda, u) \tag{49}$$

$$u_c^*(t) = \begin{cases} u_{min} \text{ if } u^* < u_{min} \\ u^* \text{ if } u_{min} \leq u^* \leq u_{max} \\ u_{max} \text{ if } u^* > u_{max} \end{cases} \tag{50}$$

The equality constraint multipliers are updated using the jacobian of the hamiltonian, yielding the following differential equations and conditions:

$$\begin{aligned} \dot{x} &= F(x, u) \\ \dot{\lambda} &= -\nabla H_x(x, \lambda, u) \\ x(t_0) &= x_0 \\ \lambda(t_f) &= 0 \end{aligned} \tag{51}$$

The two point boundary value problem must be solved. Since λ_0 is unavailable $\lambda(t_f)$ must be solved by repeatedly 'guessing' λ_0 and simulating the system forward. The initial guess can be corrected using Newton-Rhapson for each simulation:

$$\lambda(t_0) = \lambda(t_0) - \frac{\partial \lambda(t_f)}{\partial \lambda(t_0)}^{-1} \lambda(t_f) \tag{52}$$

5.1 Single-Shooting

The analytical optimal solution for u removes the need for nonlinear optimization and iterations, but CasADI's symbolical framework is still useful for constructing the integrator.

Algorithm 10: Single-shooting with PMP

Data: $x_k = x_0, u = [u_0, \ldots, u_{N-1}], \lambda_k = \lambda_0, h, N$
Construct integrator $f(ODE, x, u, h)$ with CasADI-symbolics
while $\lambda(t_f) > tol$ **do**
  for $i = 0 : N-1$ **do**
    $u^*_{c,k} = \arg\min_u H(x, \lambda_k, u)$
    $x_k = f(@SIR, x_k, u^*_{c,k}, h)$
    $\lambda_k = f(@(-\nabla_x H(x, \lambda_k, u^*_{c,k})), x_k, u^*_{c,k}, h)$
  end
  $\lambda_0 = \lambda_0 - \frac{\partial \lambda(t_f)}{\partial \lambda_0}^{-1} \lambda(t_f)$
  $\lambda_k = \lambda_0$
end

5.2 Multiple-Shooting

Newton-Rhapson in PMP may struggle to converge both due to initial guess $\lambda(t_0)$ and the system dynamics. To account for this, the states can be lifted in order to give multiple sets of constraints to be satisfied.

$$\begin{bmatrix} x(t_0) - x_0 \\ x_1 - f(@SIR, x_0, u^*_{c,0}, h) \\ \lambda_1 - f(@(-\nabla_x H(x_0, \lambda_0, u^*_{c,0})), x_0, u^*_{c,0}, h) \\ \vdots \\ x_1 - f(@SIR, x_{N-1}, u^*_{c,N-1}, h) \\ \lambda_N - f(@(-\nabla_x H(x_{N-1}, \lambda_{N-1}, u^*_{c,N-1})), x_{N-1}, u^*_{c,N-1}, h) \\ \lambda_N \end{bmatrix} = 0 \tag{53}$$

Algorithm 11: Multiple-shooting with PMP

Data: $X = MX \in \mathbb{R}^{3\times(N+1)}, \lambda = MX \in$
$\mathbb{R}^{3\times(N+1)}, [u_{0,0}, \ldots, u_{0,N-1}], \lambda_0, h, N, r = []$
Set initial guess $S_0 = [x_0, \lambda_0, \ldots, x_f, \lambda_f]$
add to r:
- Initial state constraint $(X_0 = x_0)$
- $x_k - f(@SIR, x_k, u^*_{c,k}, h)\ \forall k$
- $\lambda_k - f(@(-\nabla_x H(x, \lambda_k, u^*_{c,k})), x_k, u^*_{c,k}, h)\ \forall k$
End costate constraint $(\lambda_f = 0)$
while $\Delta S_k < \Delta S_{tol}$ **do**
  for $k = 0 : N-1$ **do**
    $u^*_{c,k} = \arg\min_u H(S, u_{0,k})$
  end
  while $|r(S)| > tol$ **do**
    $S_k = S - \frac{\partial r(S,U)}{\partial S}^{-1} r(S, U)$
  end
  $\Delta S_k = S_k - S_{k,old}$
end

6 Results

6.1 Single-Shooting

6.1.1 CasADI-IPOPT

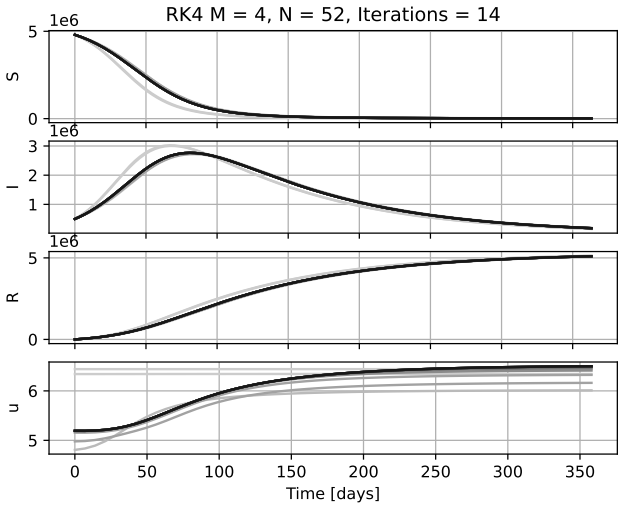

Figure 7: Single-Shooting Trajectory with social distancing objective using CasADI-IPOPT

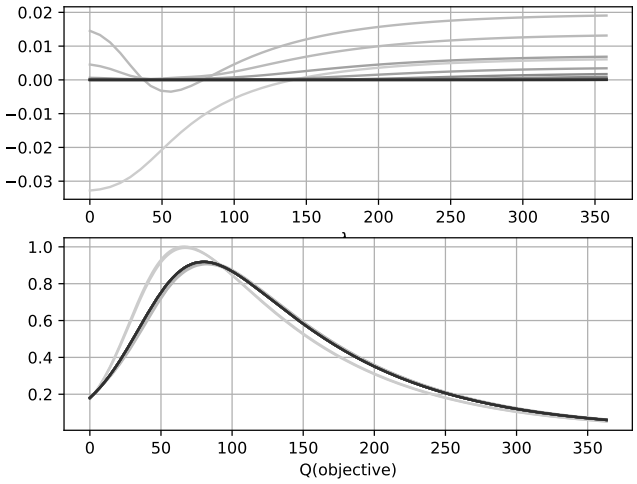

Figure 8: Single-Shooting bounds multipliers and objective with social distancing control using CasADI-IPOPT

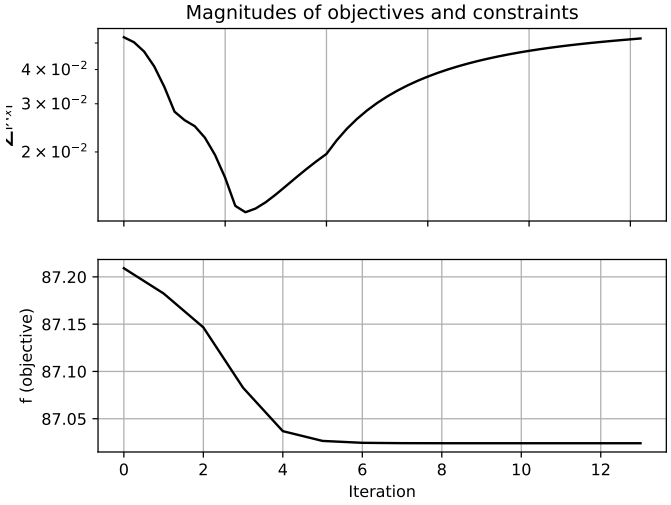

Figure 9: Single-Shooting multiplier and objective magnitudes with social distancing control using CasADI-IPOPT

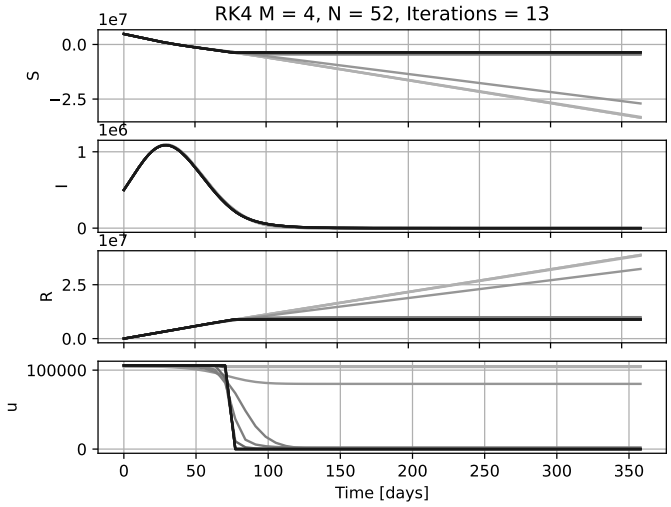

Figure 10: Single-Shooting Trajectory with vaccination control using CasADI-IPOPT

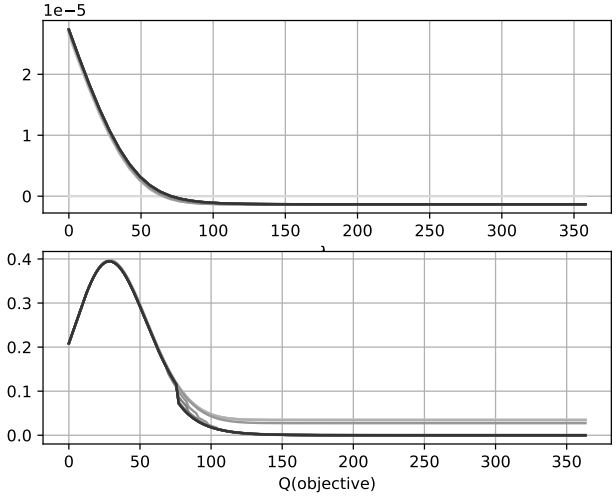

Figure 11: Single-Shooting bounds multipliers and objective with vaccination control using CasADI-IPOPT

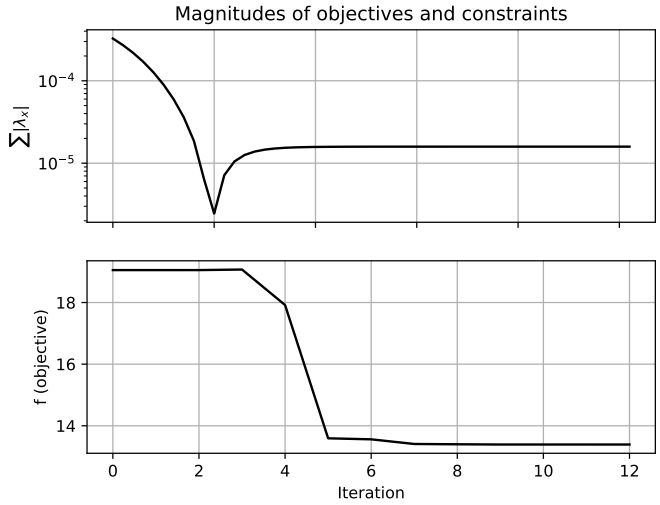

Figure 12: Single-Shooting multiplier and objective magnitudes with vaccination control using CasADI-IPOPT

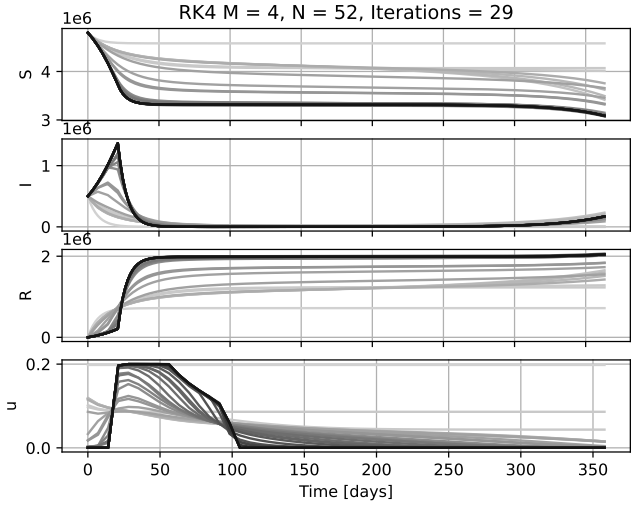

Figure 13: Single-Shooting Trajectory with isolation control using CasADI-IPOPT

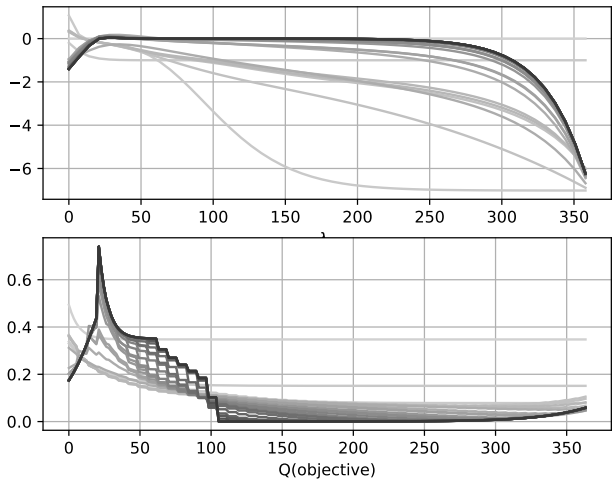

Figure 14: Single-Shooting bounds multipliers and objective with isolation control using CasADI-IPOPT

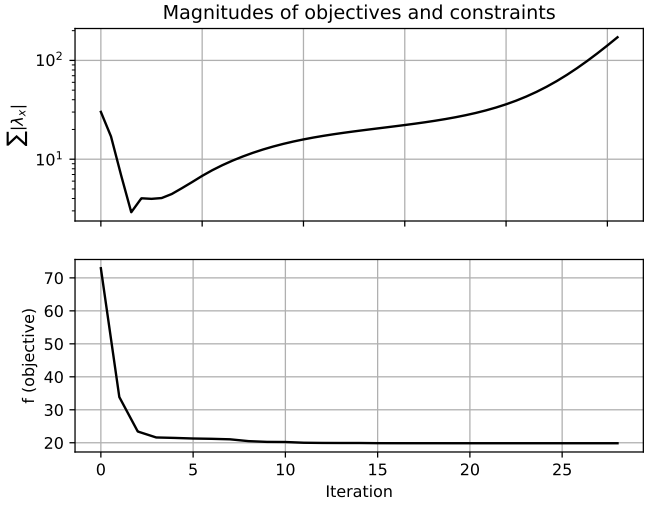

Figure 15: Single-Shooting multiplier and objective magnitudes with isolation control using CasADI-IPOPT

6.1.2 IPOPT with CasADI's Symbolical Framework

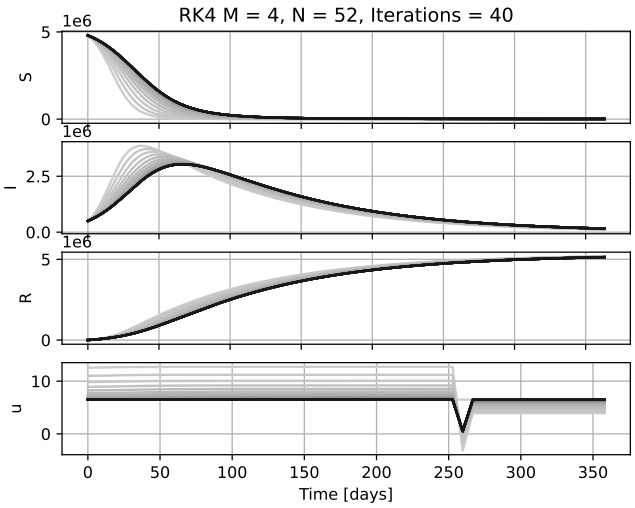

Figure 16: Single-Shooting Trajectory with social distancing objective using CasADI's symbolical framework (infeasible)

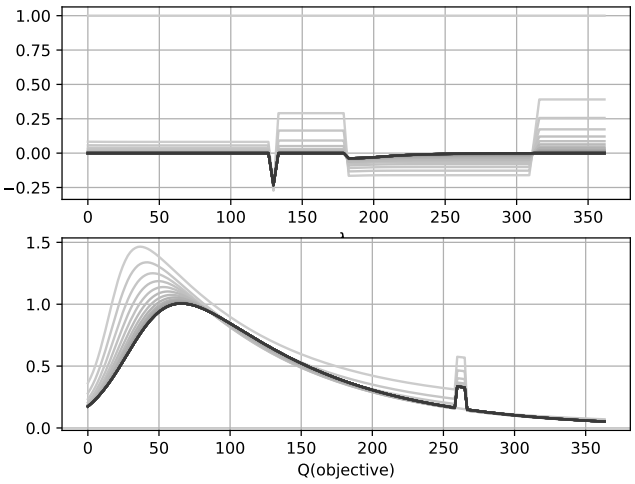

Figure 17: Single-Shooting bounds multipliers and objective with social distancing control using CasADI's symbolical framework (infeasible)

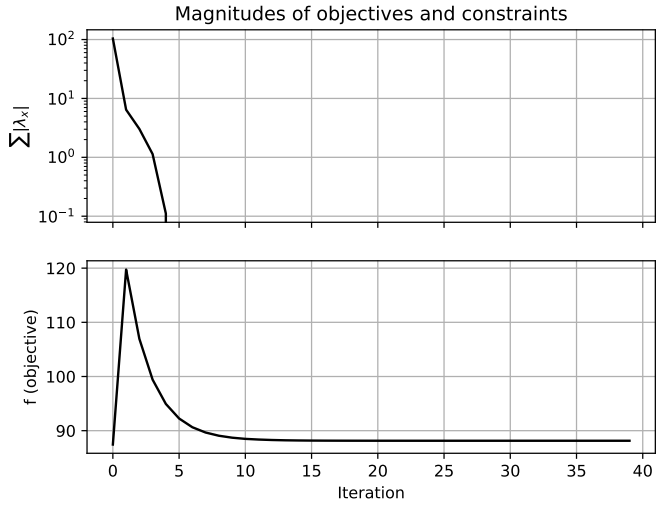

Figure 18: Single-Shooting multiplier and objective magnitudes with social distancing control using CasADI's symbolical framework (infeasible)

The vaccination and isolation control strategies did not converge under the used configuration, better initial parameters are required for convergence.

6.1.3 CasADI-SQP

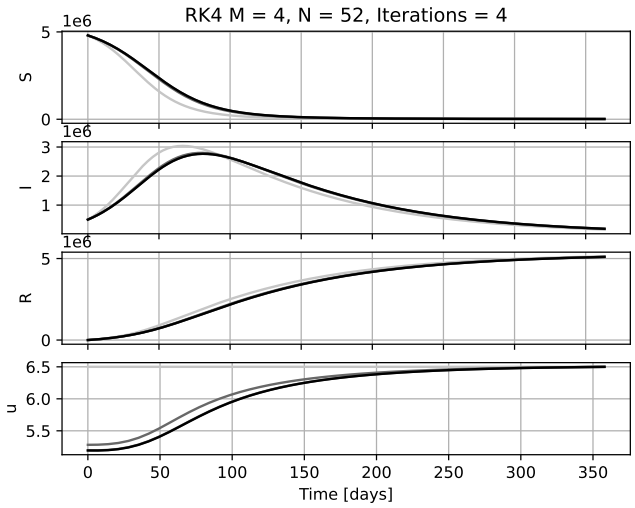

Figure 19: Single-Shooting Trajectory with social distancing objective using CasADI-SQP

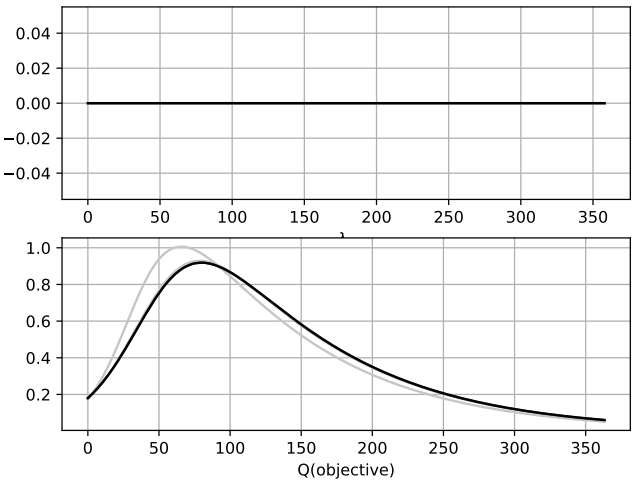

Figure 20: Single-Shooting bounds multipliers and objective with social distancing control using CasADI-SQP

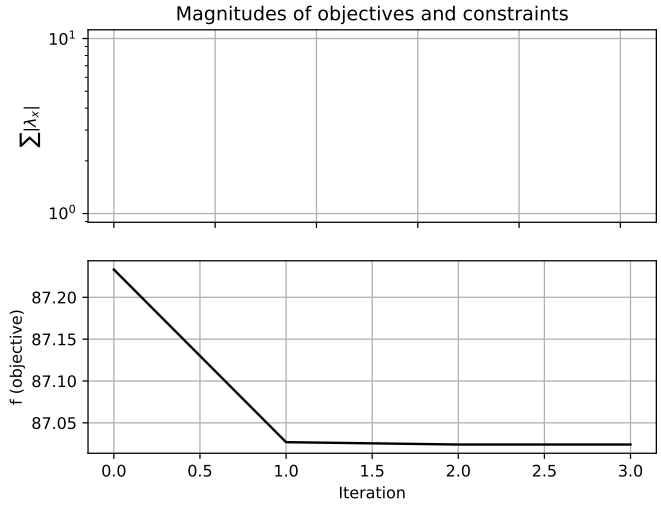

Figure 21: Single-Shooting multiplier and objective magnitudes with social distancing control using CasADI-SQP

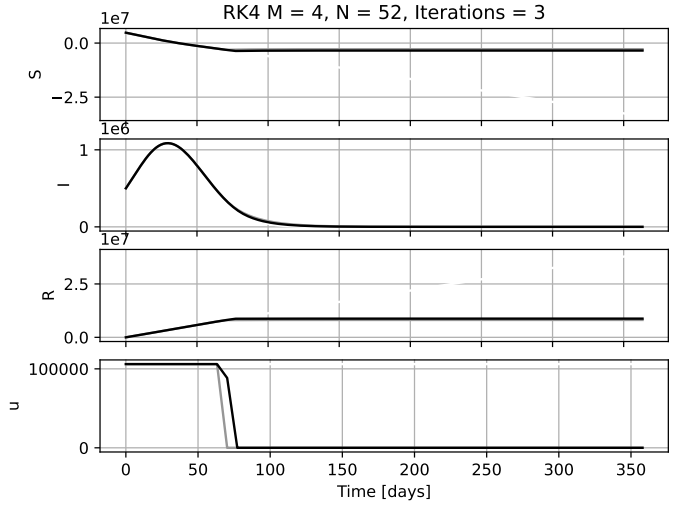

Figure 22: Single-Shooting Trajectory with vaccination control using CasADI-SQP

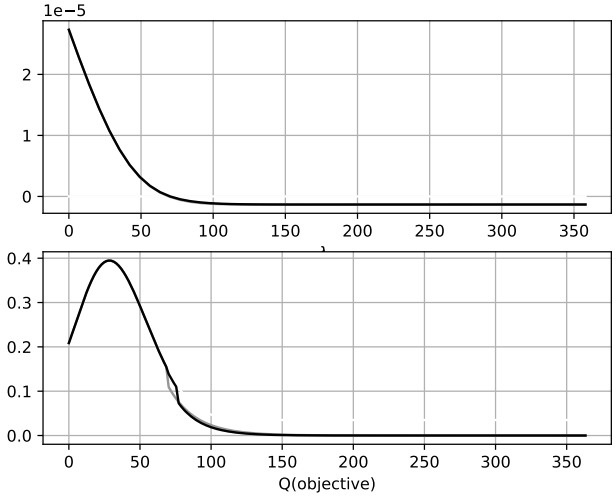

Figure 23: Single-Shooting bounds multipliers and objective with vaccination control using CasADI-SQP

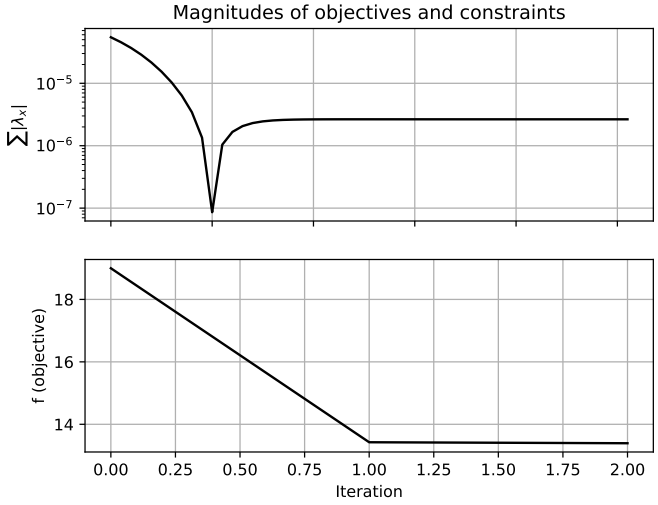

Figure 24: Single-Shooting multiplier and objective magnitudes with vaccination control using CasADI-SQP

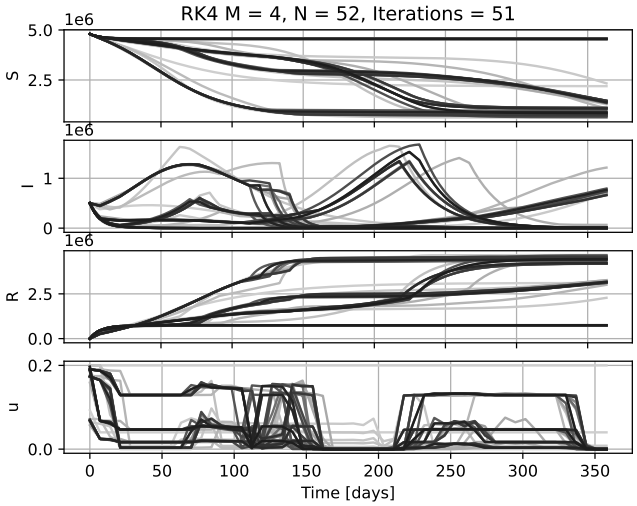

Figure 25: Single-Shooting Trajectory with isolation control using CasADI-SQP (infeasible)

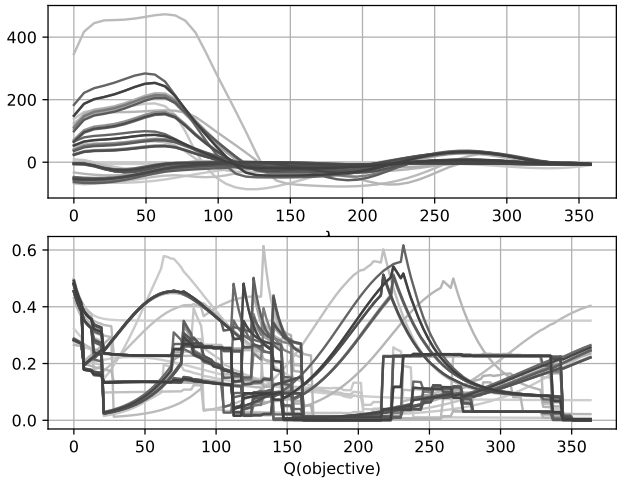

Figure 26: Single-Shooting bounds multipliers and objective with isolation control using CasADI-SQP (infeasible)

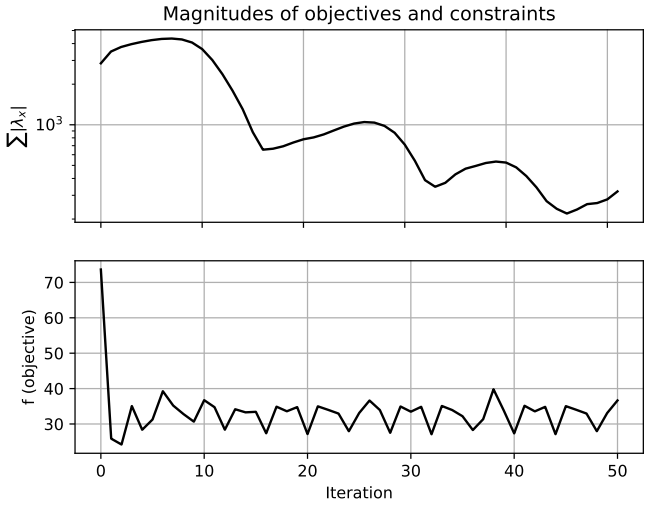

Figure 27: Single-Shooting multiplier and objective magnitudes with isolation control using CasADI-SQP (infeasible)

6.1.4 PMP with CasADI's Symbolical Framework

No convergence was found for Single-Shooting with PMP for any control strategy under the given configuration.

6.2 Multiple-Shooting

6.2.1 CasADI-IPOPT

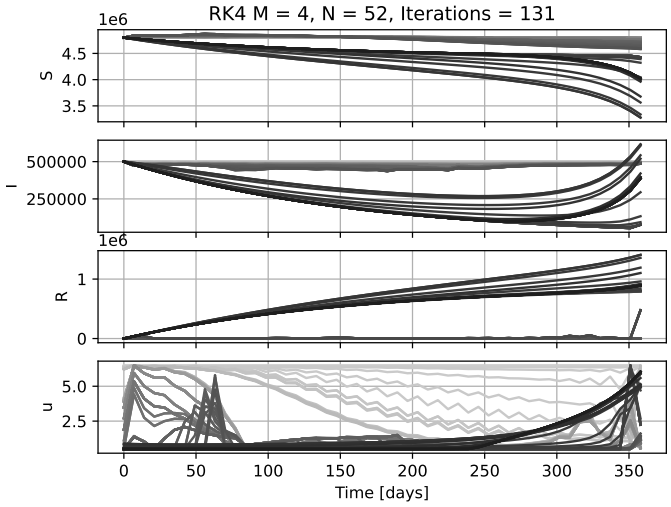

Figure 28: Multiple-Shooting Trajectory with social distancing objective using CasADI-IPOPT

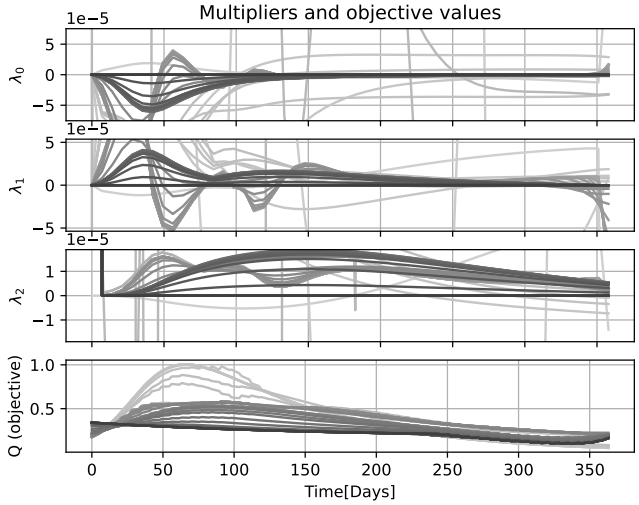

Figure 29: Multiple-Shooting bounds multipliers and objective with social distancing control using CasADI-IPOPT

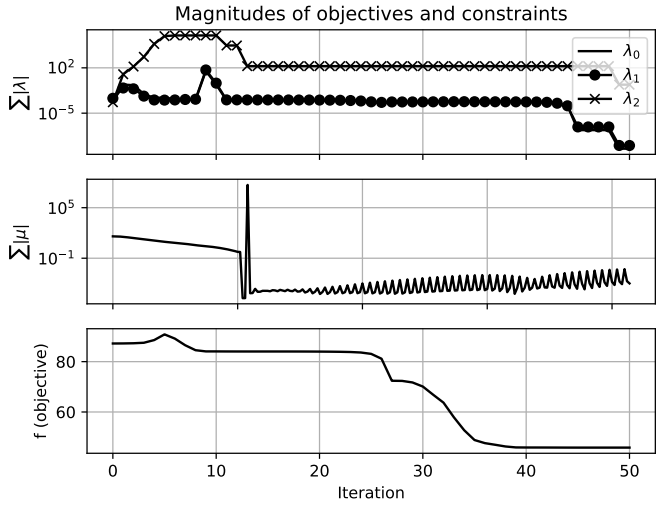

Figure 30: Multiple-Shooting multiplier and objective magnitudes with social distancing control using CasADI-IPOPT

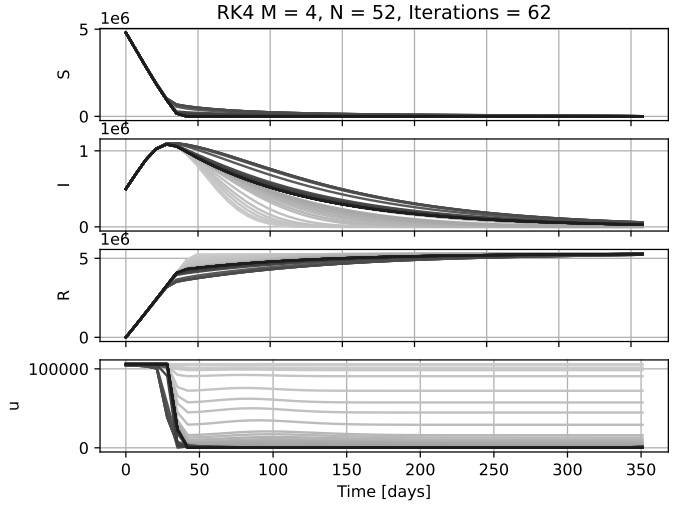

Figure 31: Multiple-Shooting Trajectory with vaccination control using CasADI-IPOPT

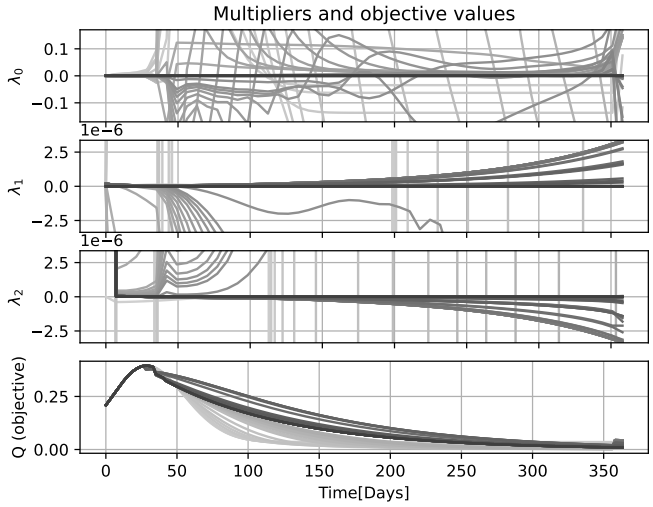

Figure 32: Multiple-Shooting bounds multipliers and objective with vaccination control using CasADI-IPOPT

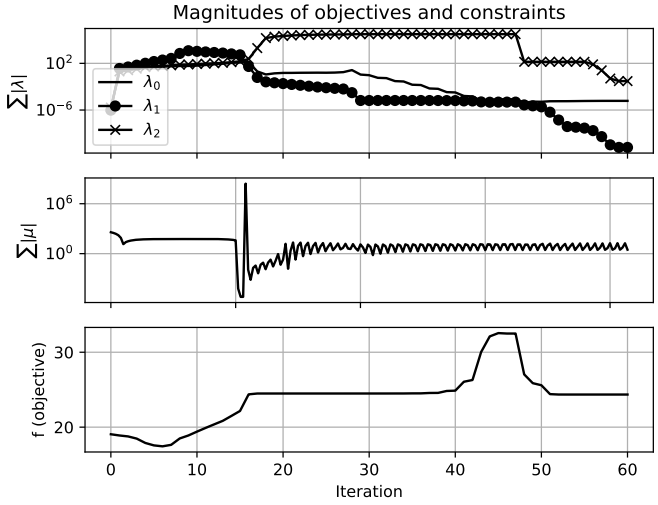

Figure 33: Multiple-Shooting multiplier and objective magnitudes with vaccination control using CasADI-IPOPT

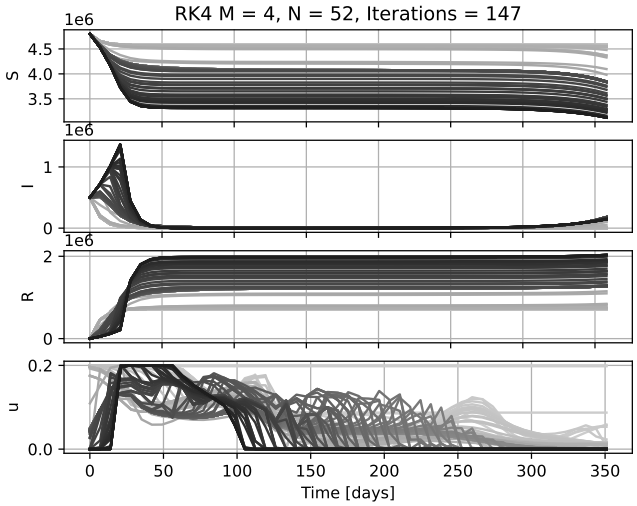

Figure 34: Multiple-Shooting Trajectory with isolation control using CasADI-IPOPT

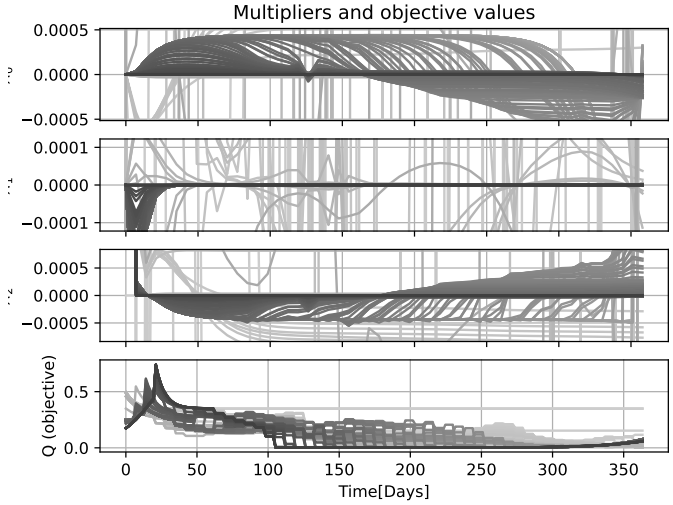

Figure 35: Multiple-Shooting bounds multipliers and objective with isolation control using CasADI-IPOPT

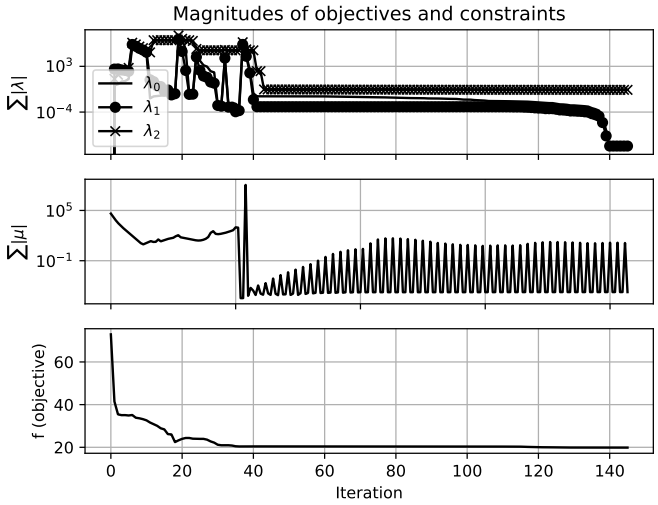

Figure 36: Multiple-Shooting multiplier and objective magnitudes with isolation control using CasADI-IPOPT

6.2.2 CasADI-SQP

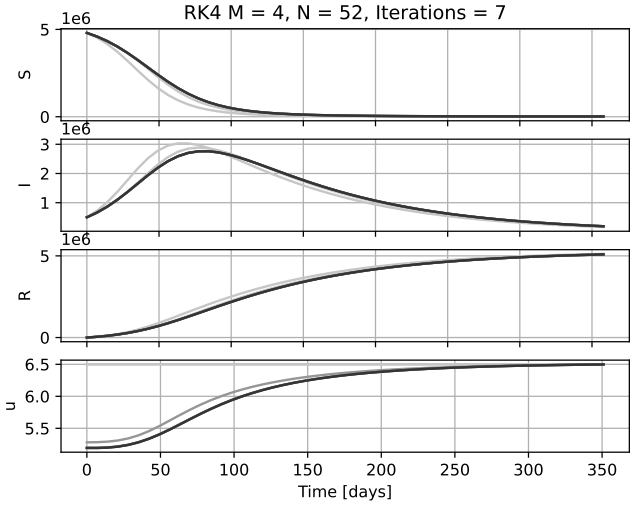

Figure 37: Multiple-Shooting Trajectory with social distancing objective using CasADI-SQP (bad convergence)

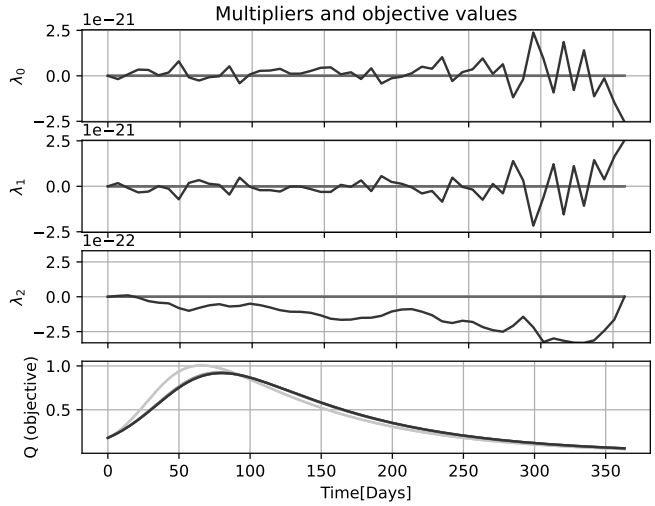

Figure 38: Multiple-Shooting bounds multipliers and objective with social distancing control using CasADI-SQP (bad convergence)

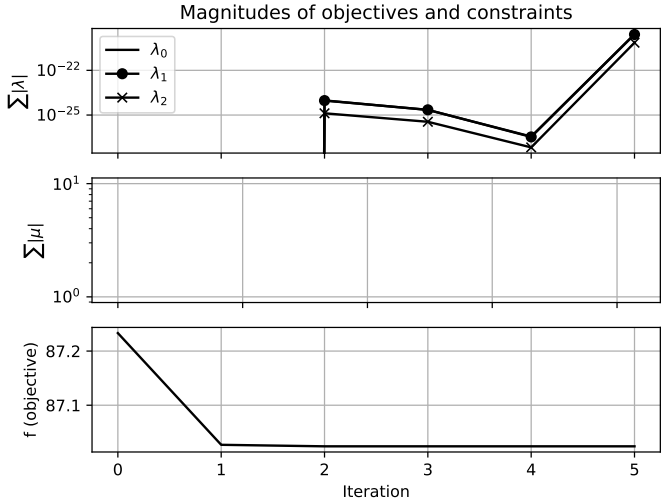

Figure 39: Multiple-Shooting multiplier and objective magnitudes with social distancing control using CasADI-SQP (bad convergence)

Vaccination and isolation control strategies did not converge.

6.2.3 IPOPT with CasADI's Symbolical Framework

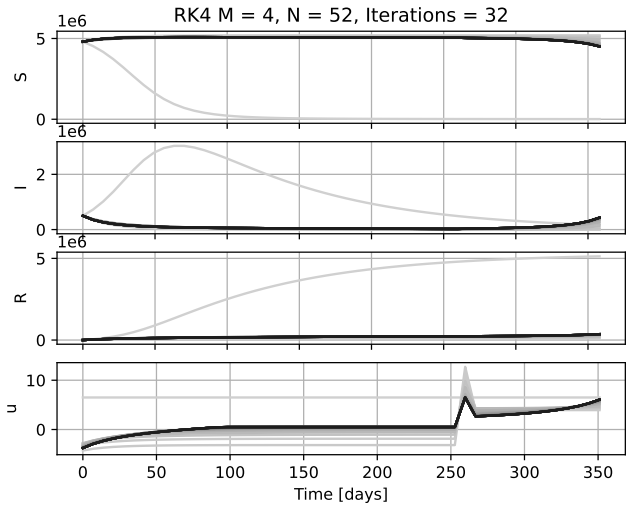

Figure 40: Multiple-Shooting Trajectory with social distancing objective using CasADI's symbolical framework (Infeasible)

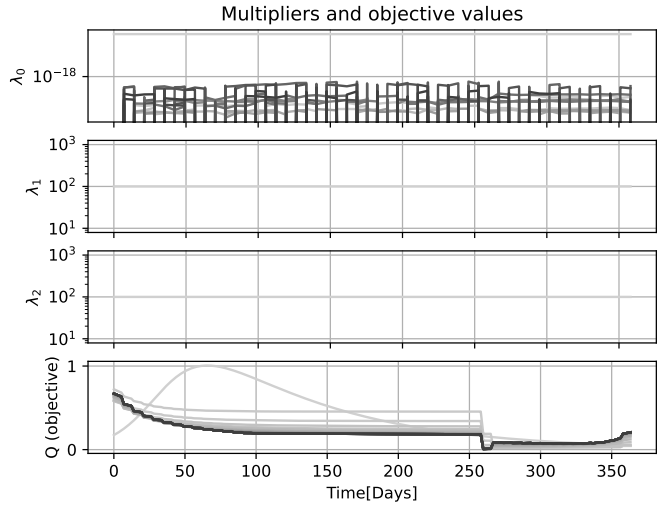

Figure 41: Multiple-Shooting bounds multipliers and objective with social distancing control using CasADI's symbolical framework (Infeasible)

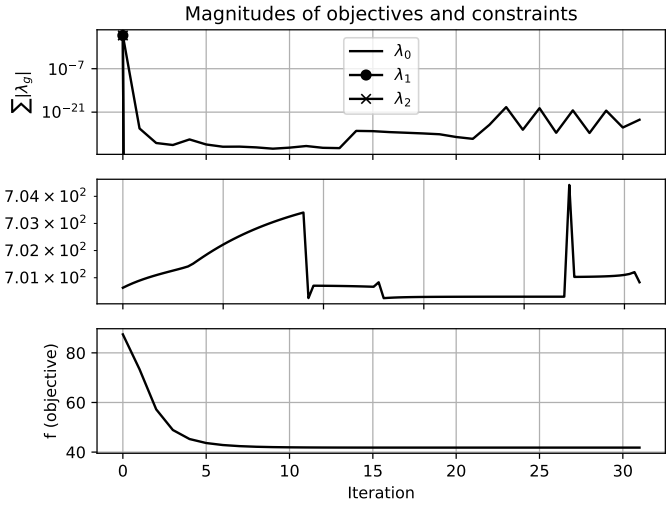

Figure 42: Multiple-Shooting multiplier and objective magnitudes with social distancing control using CasADI's symbolical framework(Infeasible)

The vaccination and isolation control strategies did not converge under the used configuration, better initial parameters are required for convergence.

6.2.4 PMP with CasADI's Symbolical Framework

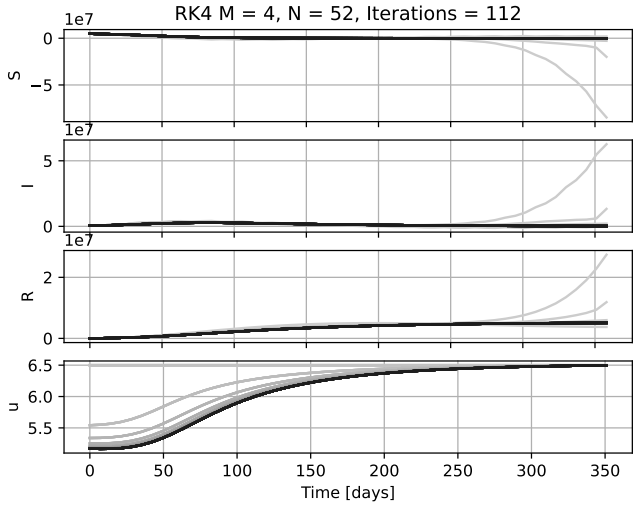

Figure 43: Multiple-Shooting Trajectory with isolation control using PMP

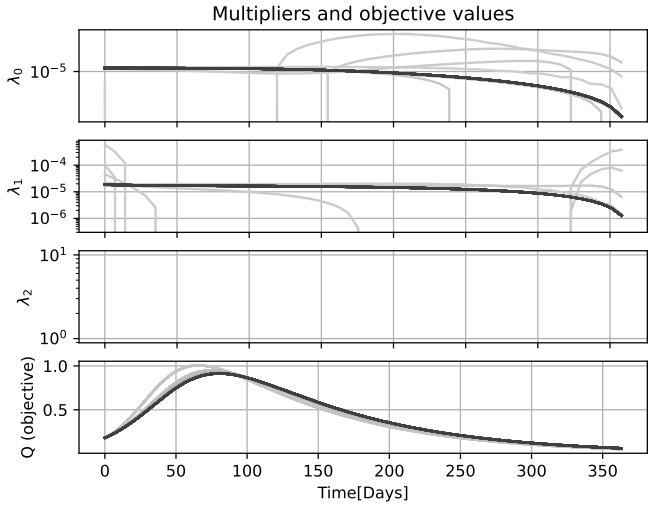

Figure 44: Multiple-Shooting bounds multipliers and objective with isolation control using PMP

The vaccination and isolation control strategies did not converge under the used configuration.

6.2.5 Direct Collocation with CasADI-IPOPT

6.2.6 CasADI-IPOPT

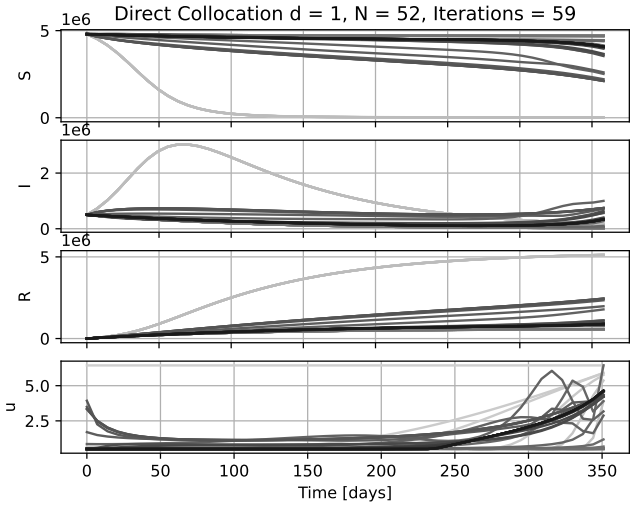

Figure 45: Multiple-Shooting Trajectory with social distancing control using Direct Collocation CasADI-IPOPT

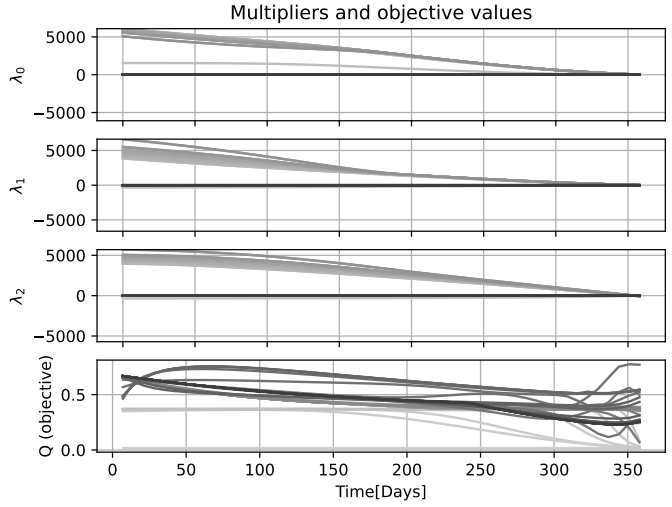

Figure 46: Multiple-Shooting constraint multipliers and objective with social distancing control using Direct Collocation CasADI-IPOPT

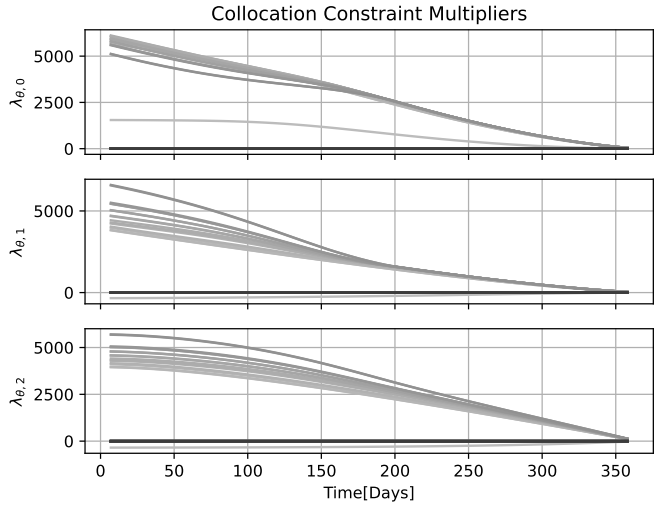

Figure 47: Multiple-Shooting collocation constraint multipliers and objective with social distancing control using Direct Collocation CasADI-IPOPT

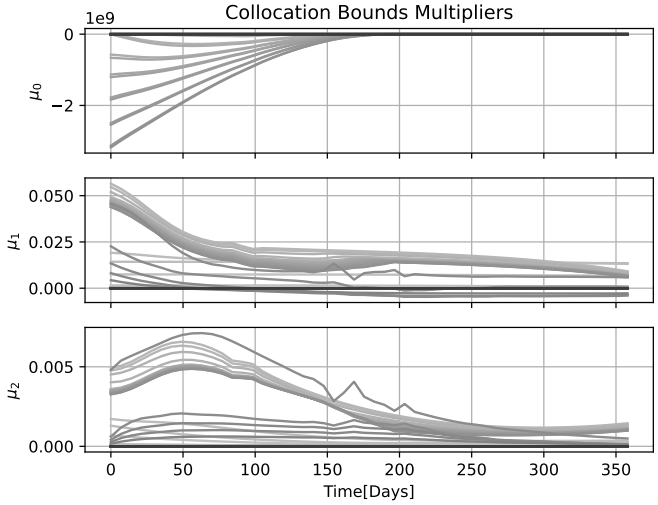

Figure 48: Multiple-Shooting collocation bounds multipliers with social distancing control using Direct Collocation CasADI-IPOPT

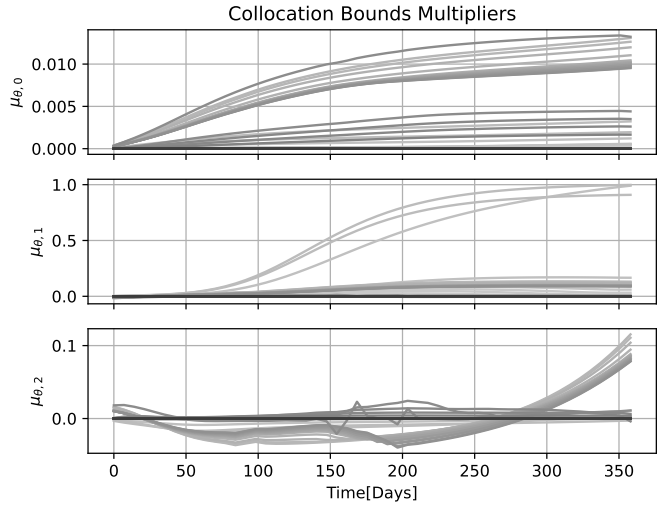

Figure 49: Multiple-Shooting collocation bounds multipliers with social distancing control using Direct Collocation CasADI-IPOPT

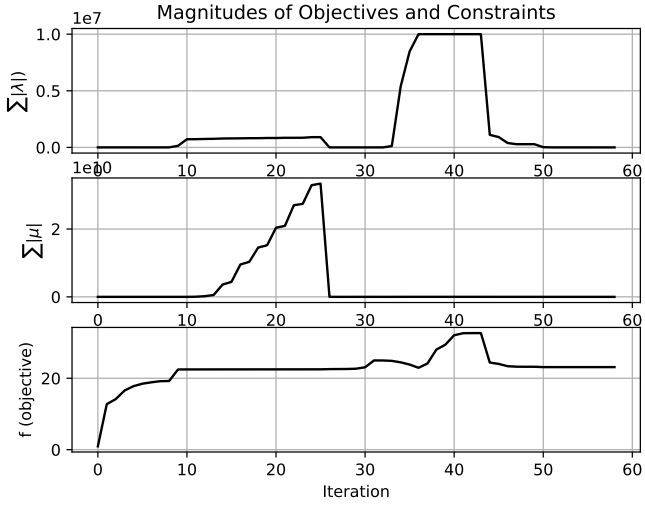

Figure 50: Multiple-Shooting objective values with social distancing control using Direct Collocation CasADI-IPOPT

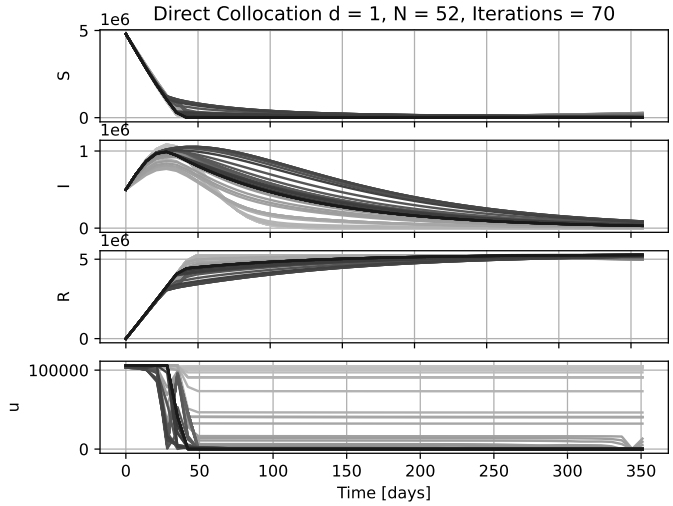

Figure 51: Multiple-Shooting Trajectory with vaccination control using Direct Collocation CasADI-IPOPT

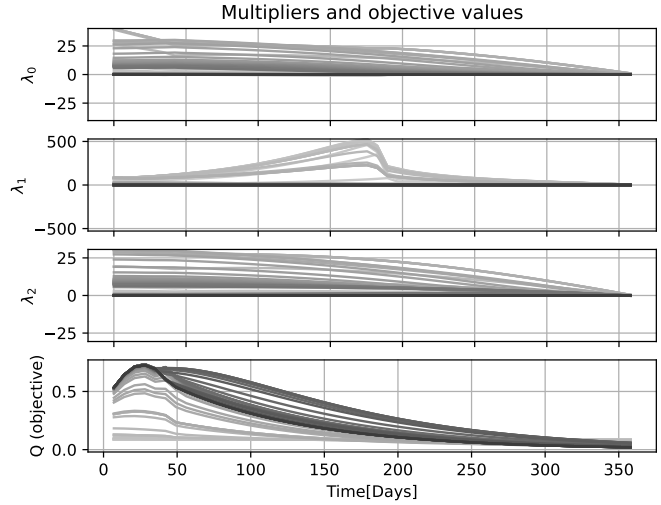

Figure 52: Multiple-Shooting constraint multipliers and objective with vaccination control using Direct Collocation CasADI-IPOPT

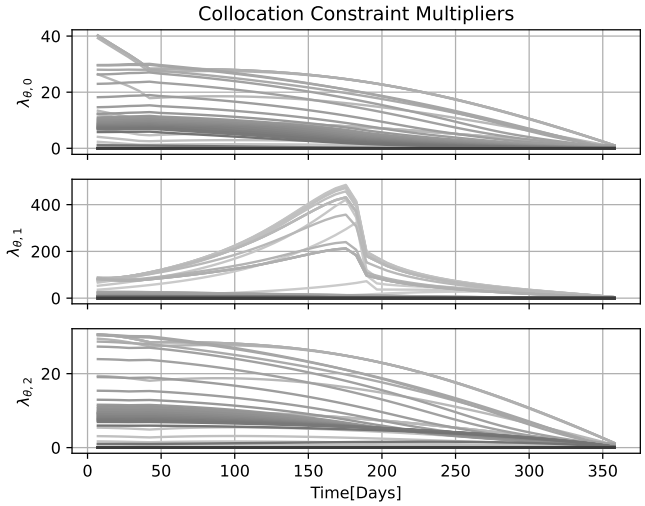

Figure 53: Multiple-Shooting collocation constraint multipliers and objective with vaccination control using Direct Collocation CasADI-IPOPT

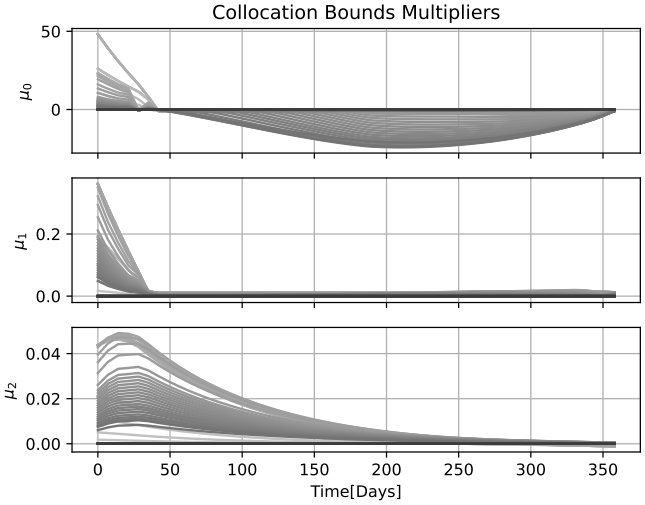

Figure 54: Multiple-Shooting collocation bounds multipliers with vaccination control using Direct Collocation CasADI-IPOPT

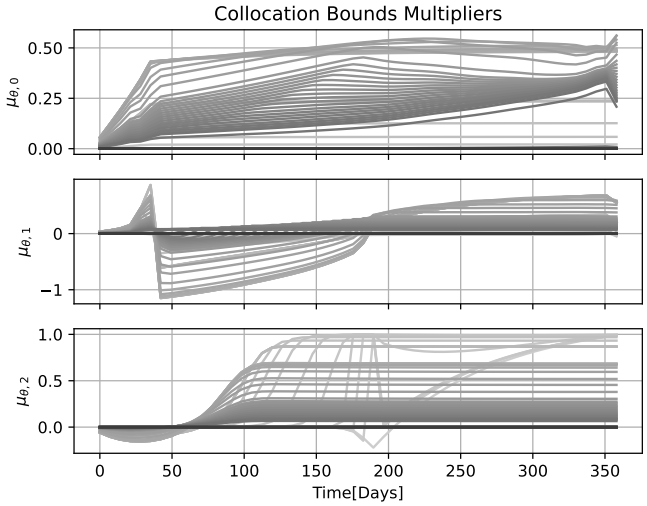

Figure 55: Multiple-Shooting collocation bounds multipliers with vaccination control using Direct Collocation CasADI-IPOPT

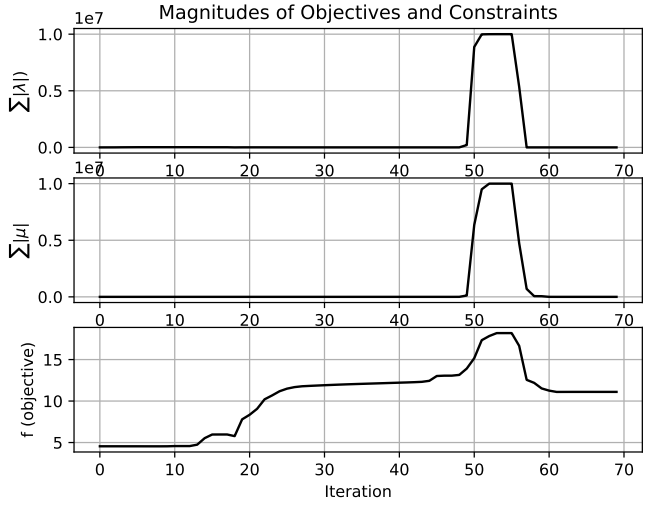

Figure 56: Multiple-Shooting objective values with vaccination control using Direct Collocation CasADI-IPOPT

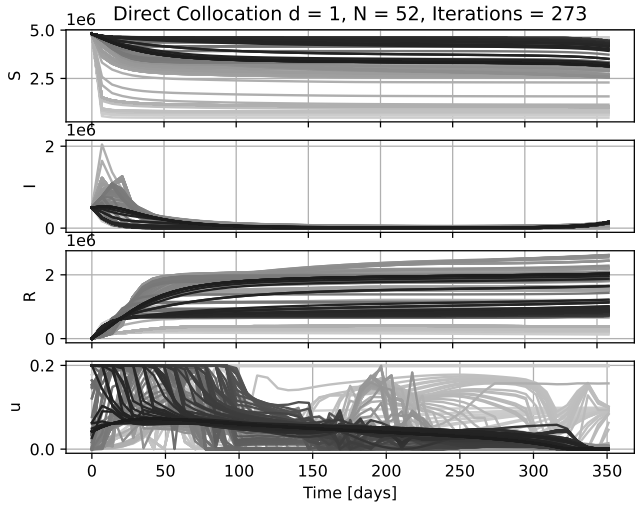

Figure 57: Multiple-Shooting Trajectory with isolation control using Direct Collocation CasADI-IPOPT

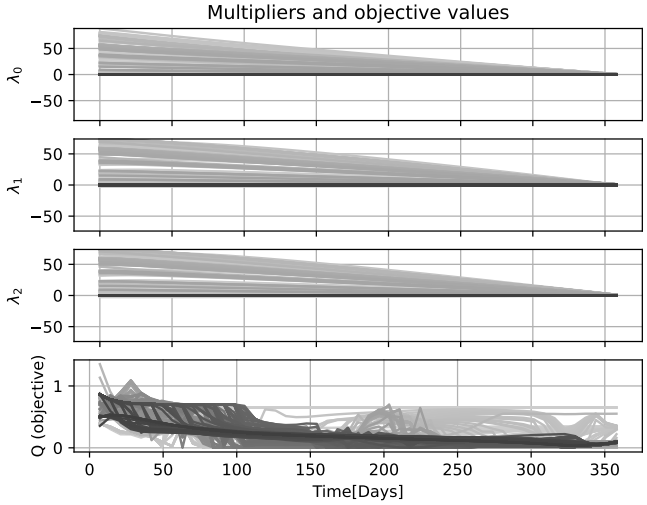

Figure 58: Multiple-Shooting constraint multipliers and objective with isolation control using Direct Collocation CasADI-IPOPT

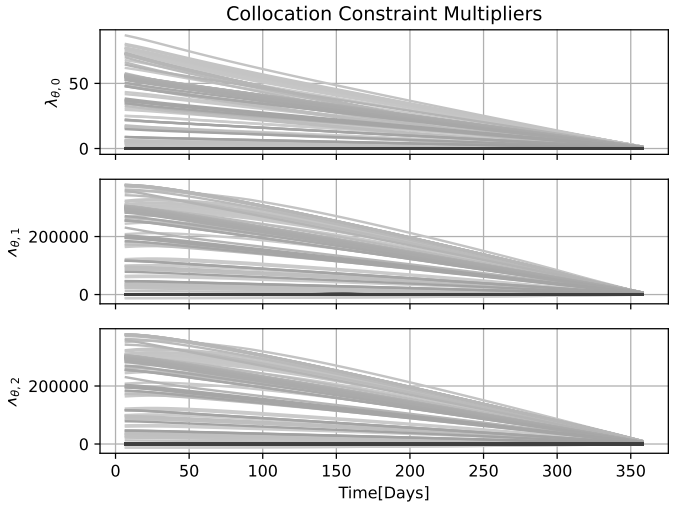

Figure 59: Multiple-Shooting collocation constraint multipliers and objective with isolation control using Direct Collocation CasADI-IPOPT

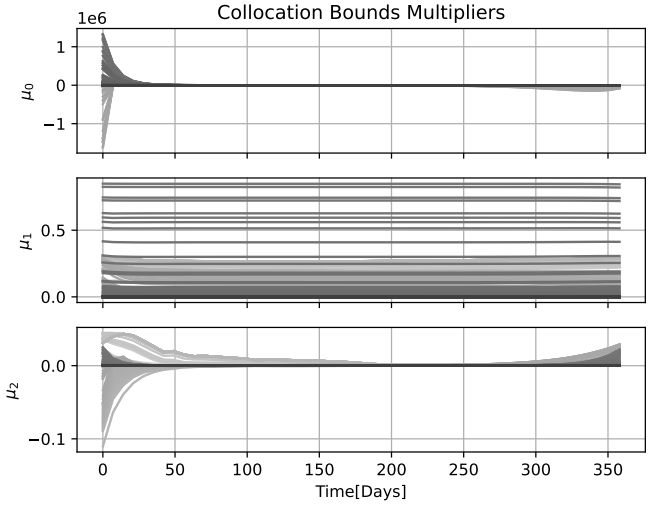

Figure 60: Multiple-Shooting collocation bounds multipliers with isolation control using Direct Collocation CasADI-IPOPT

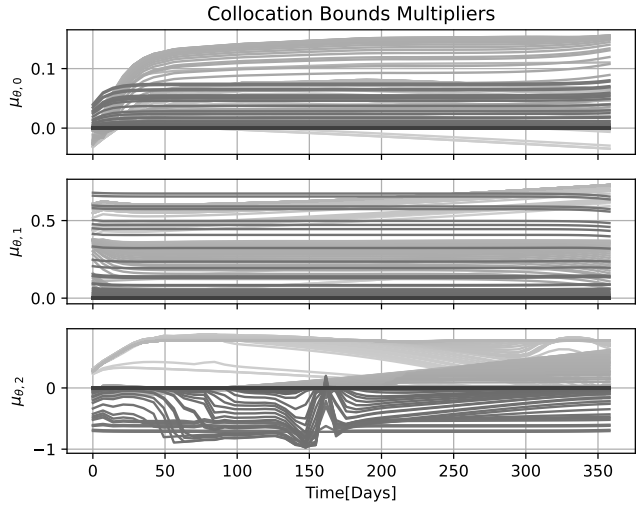

Figure 61: Multiple-Shooting collocation bounds multipliers with isolation control using Direct Collocation CasADI-IPOPT

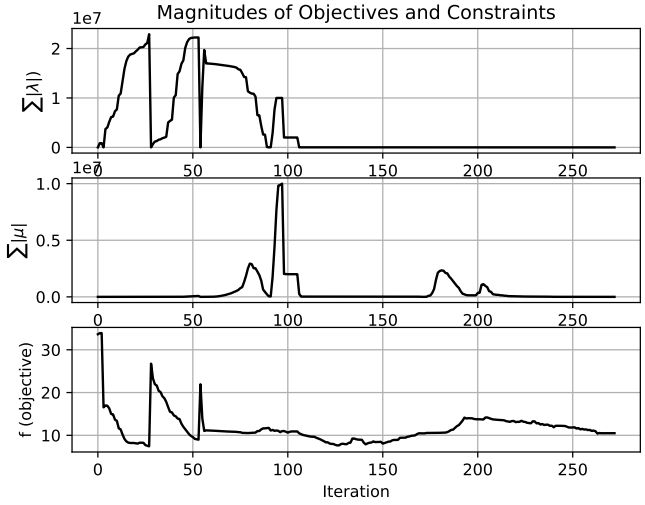

Figure 62: Multiple-Shooting objective values with isolation control using Direct Collocation CasADI-IPOPT

6.3 Comparison of convergence

Solver/Method	Social Distancing	Vaccination	Isolation
Custom IPOPT	40	N/A	N/A
Custom PMP	N/A	N/A	N/A
CasADI-IPOPT	14	13	29
CasADI-SQP	4	3	51

Table 2: Iterations for different solvers using single-shooting

Solver/Method	Social Distancing	Vaccination	Isolation
Custom IPOPT	N/A	N/A	N/A
Custom PMP	112	N/A	N/A
CasADI-IPOPT	131	62	147
CasADI-SQP	7^*	N/A	N/A
CasADI-IPOPT(Collocation)	59	70	273

Table 3: Iterations for different solvers using multiple-shooting.(*SQP converged without significant objective value decrease)

7 Discussion

7.1 Custom IPOPT and PMP

Under the given parameter configuration the custom Primal-Dual and PMP methods in general struggle to converge. Since there are no implemented methods to handle semi-definiteness (Like IPOPT's recovery steps and QPOASES' flipping bounds) the invertibility in the newton iterations are fragile.

Backtracking linesearch and adding positive diagonal elements to the hessian did not enable other forms of convergence either. Optimization for semidefinite hessians needs improvements to make these implementations viable.

7.2 IPOPT

The IPOPT-solver has been the most reliable solver in the scope of this project. Problems that have been infeasible for IPOPT have not been solvable for any of the other methods either.

Convergence should in general also be faster than SQP for the larger problems, and did mostly converge in an acceptable amount of iterations (< 200). The high number of iterations for the convergence using collocation is likely caused by bad initial values. Still, the IPOPT-solver stands out as the most robust solver for the problem formulation. It could therefore be useful to further study the recovery and backtracking methods implemented in this solver, which is well documented in the papers provided.

7.3 SQP

QPOASES converged faster for single-shooting objectives than IPOPT, but the objective values during its iterations are questionably smaller than those for IPOPT. The result for single-shooting vaccination seems more comparable to the workload required by IPOPT, but in this case IPOPT converges in significantly fewer iterations. As expected SQP did not perform well with multiple-shooting and did not yield any satisfactory results for the chosen parameters. The limitations of the QPOASES-solver should be studied to better tune the solver and improve performance.

7.4 Convergence of Trajectories

7.4.1 CasADI-IPOPT

For single-shooting social distancing the IPOPT solver is coherent with the integrator sensitivity analysis. It imposes restrictions in the most control-sensitive time period, allthough these were relatively weak restrictions (Figure 7). In multiple-shooting it tries to impose restrictions from the end of the year, and slowly transitions to a lockdown scenario lasting 250 days (Figure 28). Multiple-shooting reduced the objective better than single-shooting, which shows that

this approach can achieve different convergence even with the same initial trajectory. For vaccination and isolation the methods converge towards similar solutions. The resulting vaccination strategy is to vaccinate as many as possible early on and later stop vaccinating when the number of susceptibles drop (Figures 10, 31). In contrast the optimal isolation strategy does not initiate immediately but is initiated once a substantial amount of the population is infected (Figures 13, 34). At this point isolation is performed until the disease is nearly exterminated, the defined control input will never be able to exterminate the disease.

Using direct collocation with Radau timepoints for multiple-shooting with CasADI-IPOPT resulted in convergence with fewer iterations. The control input strategies are also similar to those proposed under ERK4 for social distancing and vaccination, but there are very different types of convergence. Due to hard initial constraint violations in direct collocation the constraint multipliers end up with a different convergence affected by constraint and bound multipliers of a much larger magnitude. This could be an indication of a higher tolerance for bad initial values in the collocation scheme, which makes sense with respect to how its L-stability outperforms ERK4 for locally unstable systems.

Radau timepoints

7.4.2 QPOASES

QPOASES is able to converge fast for the single shooting problems, with the exception of the isolation strategy (Figure 25) where the solver encounters indefinite hessians and fails to converge. The objective value entered an oscillatory state and had to be terminated after 1000 iterations. The solver struggles to determine when to stop the first round of isolation control ($0 \rightarrow\sim 150$days) and implements a second round ($\sim 210 \rightarrow 340$days) as a consequence. For social distancing and vaccination (Figures 19, 22) the convergence is much faster. Comparing these convergences with CasADI-IPOPT it is possible to see how the damping from logarithmic barriers impairs convergence.

QPOASES does not converge for multiple-shooting methods in general but managed to find a solution for social distancing. The stepsizes are reduced to insignificant magnitudes in the first QP with a fixed number of active constraints. From this experience IPOPT should be the first approach to solving the multiple-shooting problems, but SQP methods is worth considering for problems of smaller dimensions if computational efficiency is needed.

7.4.3 Mutliplier Trajectories

Both logarithmic and linearly scaled plots of the multiplier convergences have been included, but both of them struggle to capture and illustrate full convergence. The logarithmic plots did in some cases provide spikes (rapid value decrease in multipliers at single time instants) that correlated well with decreased objective values in the following iterations, but the plots become harder to interpret when the magnitudes of the multipliers are more uniformly distributed.

The linearly scaled plots show a more detailed convergence but is harder to scale correctly.

Implementing a dimensionless SIR-model ($N_{pop} = 1, t \in [0, 1]$) and omitting the recovered compartment would likely simplify the scalings and make the multiplier-plots more viable. This could also improve the numerical feasibility of the problems and make it easier to determine initial conditions and parameters.

7.4.4 Iteration Magnitude Plots

The magnitude plots uncover how the solvers prioritize objective minimization against the search for feasible constraint and bounds multipliers. The iteration plots can be used in combination with the other plots to find the regions and time instants of the trajectories that the solver struggle to find good solutions for.

7.5 Objectives and input bounds

7.5.1 Social Distancing

The abstract cost of imposing social distancing restrictions on a population can come in many other shapes than a quadratic cost. A linear cost was also considered, but most solvers had difficulties with convergence in this case. Using $u = \mathscr{R}_0$ made it easier to determine if the convergence problem was stability issues or solver issues. By upper bounding the input to $u \leq 1$ the system becomes locally stable for the full control horizon.

7.5.2 Vaccination

Properly limiting capacity was important to ensure convergence for the vaccination strategy. Allowing vaccination of the full population over a few weeks lead to a close to binary decision for the solver, where it either was worth it to fully vaccinate the population, avoid the epidemic and introduce a dominant linear flow rate from S to R, or it was better to not vaccinate at all. The solvers had difficulties making this decision. Exchanging the objective with a quadratic one improved convergence in this case, it can be compared to the damping effect of implementing a logarithmic barrier against the upper bounds.

7.5.3 Isolation

Isolation control does not impair the bilinear infection dynamics in the same way as vaccination. The number of infected that can be isolated are itself depending on the bilinear dynamic, and thus it is unable to eradicate the epidemic without quarantining the full population. There are issues with the introduced control strategy, it does not account for isolated susceptibles. A solution to this would be to introduce a separate compartment Q for quarantined individuals, resulting in a SIQR-model.

7.6 Notes on the Implementation

Many aspects of the CasADI implementation and data retrieval have been omitted from the report, but the project has provided a thorough introduction to the Python application interface. The symbolical framework of CasADI has many interesting features which could make custom solver implementations more viable, like the possibility of translating and compiling symbolical functions before execution, or during execution with a Just In Time (JIT) compiler.

When the features needed from CasADI are provided and well documented it is easy to implement things, but when the abstractions become unclear, and the inner workings of the solvers are disguised, it is a lot harder to determine the right course of action. Porting implementations to C++ could be a good solution to this problem.

8 Future Work

8.1 Network Optimization with Mean-Field Approximations

Real-world disease spread is often not approximated well by simple ODE-models, but require spatial information to explain the correct dynamics. Stochastic network models are therefore used to better capture epidemics.

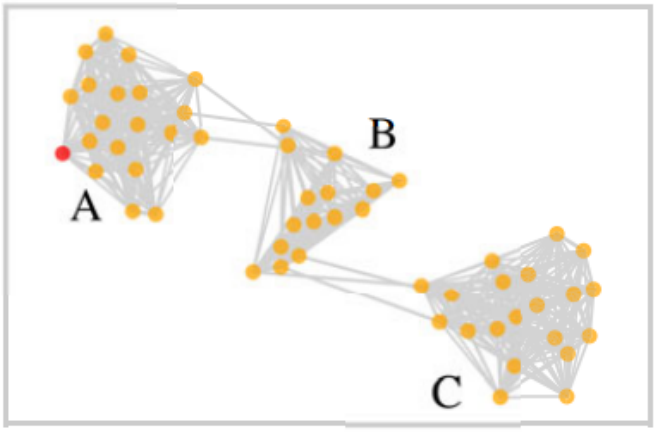

Figure 63: Interaction network consisting of 3 clusters [Bussell et al. 2018]

Network models unfortunately lack many of the analytical properties of ODEs, but allows other kinds of information to be used to identify parameters, which is crucial considering the lack of verification data for the fitted ODE-models. Bussell et al. 2018 shows on an interaction network with 3 clusters that optimal control can be improved for network models by approximating ODEs for multiple clusters. This network approximation can possibly be implemented as a differential algebraic equation where each deterministic SIR-model additionally are constrained to stochastic perturbations from the others.

$$
\begin{aligned}
\min_{w} \quad & \Phi(x,z,u) = \Phi_A(x_A,z,u_A) + \Phi_B(x_B,z,u_B) + \Phi_C(x_C,z,u_C) \\
\text{s.t.} \quad & F_A(x_A(t),z(t),u_A(t)) = \dot{x}_A(t), \\
& F_B(x_B(t),z(t),u_B(t)) = \dot{x}_B(t), \\
& F_C(x_C(t),z(t),u_C(t)) = \dot{x}_C(t), \\
& G_A(x_A(t),z(t),u_A(t)) = 0, \\
& G_B(x_B(t),z(t),u_B(t)) = 0, \\
& G_C(x_C(t),z(t),u_C(t)) = 0
\end{aligned}
$$

Where G_A, G_B and G_C constrain ODE A, B and C to account for the stochastic interactions between the clusters. G will contain stochastic distributions because of the stochastic network dynamics, but defining confidence intervals for robust optimal control could make this solveable.

8.2 System Identification Methods in Optimization

The IPOPT and SQP methods can be viewed as iterative, sequential approaches that first find alternative system dynamics to optimize, followed by obtaining the optimal solution for the system. The logarithmic barriers and quadratic programs are used to give locally feasible search directions that do not take previous iterations into account.

In the search for globally optimal solutions it may be beneficial to retain some information from the previous iterations and try to identify a model for the underlying global structure. In this sense it would be interesting to apply linear autoregressive methods like ARX, or the nonlinear equivalent NARX in combination with embedded feature selection methods like the FROLS algorithm [Billings 2013], in order to form a model that searches for a global minimum instead of local ones.

8.3 Alternative to Multiple-Shooting in PMP

There is a fundamental volume conservation between the states and costates when using PMP that introduce issues when the control horizon of the problem become too long. Because of the volume conservation it is impossible for both states and costates to be stable at the same time. This raises numerical issues and is likely why the single-shooting PMP approach never managed to converge.

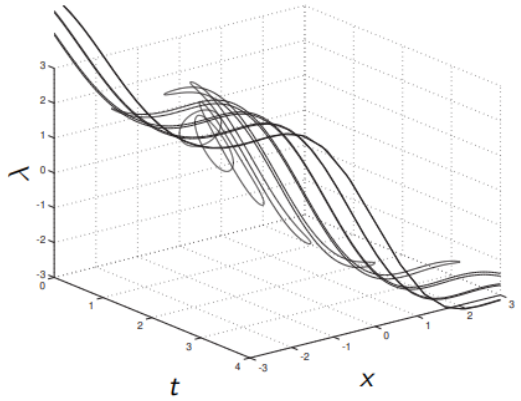

Figure 64: Conservation of volume in the state/costate-space [Gros n.d.]

By introducing multiple-shooting the problem is broken down into shorter, maintainable horizons. This can also be percieved as replacing the true system dynamics with a discontinous one, but is this discontinuity desirable? Are there alternative ways to stabilize states/costates without multiple-shooting? The following objective propose a solution with a quadratic penalty to altering system equations, with the aim of stabilizing the states and costates.

$$\min_{u,\hat{\theta}} \quad H(x,\lambda,u,\hat{\theta}) + \begin{bmatrix} F(x,u,\hat{\theta}) \\ -\nabla H_x(x,\lambda,u,\hat{\theta}) \end{bmatrix}^T W_{\hat{\theta}} \begin{bmatrix} F(x,u,\hat{\theta}) \\ -\nabla H_x(x,\lambda,u,\hat{\theta}) \end{bmatrix} + \Delta\theta^T W_{\Delta\theta} \Delta\theta$$

Where $\hat{\theta}$ are ODE-parameters and $\Delta\theta = \hat{\theta} - \theta$ is the deviation from the true parameters of the system. The middle term is introduced to penalize rapid changes in states/costates, while the third term penalize large parameter deviations from the true system. Alternatively it can be controlled with strict inequalities.

$$\begin{aligned} \min_{u,\theta} \quad & H(x,\lambda,u,\theta) \\ \text{s.t.} \quad & \begin{bmatrix} F(x,u,\theta) \\ -\nabla H_x(x,\lambda,u,\theta) \end{bmatrix} \leq K_\theta, \\ & |\Delta\theta| \leq K_{\Delta\theta} \text{ or } \Delta\theta = 0 \end{aligned}$$

Solving this problem for each time iteration will not lead to the same local optimal solution that PMP provides, and it may not even be able to converge to one at all. Quantifying the impact of nonlinear term parameters is difficult, but testing this approach would be interesting.

8.4 Solution Space Visualization with Dimensionality Reduction

The increasing dimensionality that comes with longer time horizons makes it difficult to distinguish locally converging regions to infeasible ones. Because of the curse of dimensionality it is impossible to fully preserve high-dimensional relationships when mapping it to lower-dimensional spaces, but there are methods of preserving the most 'relevant' information. It would be interesting to apply linear and nonlinear dimensionality reduction techniques to the data obtained from solvers, using grid search to choose different initial states and parameters. Examples of techniques to apply are Principal Component Analysis (PCA, linear) and T-distributed Stochastic Neigborhood Embedding (t-SNE, nonlinear).

8.5 Mixed-Integer NLPs with Bonmin

Bonmin [Bonami and Lee 2013] combines the IPOPT-solver with different algorithms in order to evaluate MINLPs. Mixed-integer solutions is relevant for optimal social distancing control. It is difficult to accurately determine the impact of nationwide restrictions in political decision making, but hard restrictions on schools, kindergartens and workplaces are clearer. Additionally, some countries have introduced nation/county-wide alert levels/restrictions that also can be captured by integer-valued inputs.

9 Conclusion

This project has given a good insight into the challenges with nonlinear optimal control methods and integrators and their sensitivities. The bilinear dynamics of the SIR epidemiological model has been addressed and to some extent optimally controlled using different mitigation strategies. Analyzing the convergence of the different solvers have revealed both advantages and disadvantages with the different solvers. The tuning has given a good insight into the challenges with stability and initial parameters.

The optimal control performed in this project has provided some insight into the challenges with epidemic control, but the SIR-model remains too simple to be applied in practice. There is much work to be done in order to obtain epidemiological models that are both identifiable and verifiable, which makes it difficult to conclude on the quality of the optimal solutions of this project. The iteration data of the solvers has given valuable insight into the process of finding optimal solutions, and have made the different optimization methods easier to approach and understand. The same applies for the choices of parameters and initial states. There is also more work to be done in order to fully understand and visualize how the bilinear dynamics of the SIR-model shapes the solution space.

The implemented code is likely to be used for future projects both in the field of optimization and epidemiological models, and has given a thorough introduction to the CasADI library.

10 Code Access

Plot code and algorithms are located in a single git repository:

```
git clone https://github.com/jonasbhjulstad/Epidemiological_Numerical_Optimization.git
```

References

Billah, Md. Arif, Md. Mamun Miah, and Md. Nuruzzaman Khan (Nov. 2020). "Reproductive number of coronavirus: A systematic review and meta-analysis based on global level evidence". In: *PLOS ONE* 15.11, pp. 1–17. DOI: 10.1371/journal.pone.0242128. URL: https://doi.org/10.1371/journal.pone.0242128.

Billings, S.A. (2013). *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*. Wiley. ISBN: 9781118535554. URL: https://books.google.no/books?id=SaQ2AAAAQBAJ.

Birgitte Freiesleben de Blasio, Francesco Di Ruscio (2021). *Situational awareness and forecasting for Norway*. URL: https://www.fhi.no/contentassets/e6b5660fc35740c8bb2a32bfe0cc45d1/vedlegg/nasjonale-og-regionale-rapporter/2021.02.03.national-and-regional-corona-report-engelsk.pdf.

Bonami, Pierre and Jon Lee (2013). *BONMINUsers' Manual*.

Bussell, E.H. et al. (2018). "Applying optimal control theory to complex epidemiological models to inform real-world disease management". In: *bioRxiv*. DOI: 10.1101/405746. eprint: https://www.biorxiv.org/content/early/2018/08/31/405746.full.pdf. URL: https://www.biorxiv.org/content/early/2018/08/31/405746.

Cevik, Muge et al. (2020). "SARS-CoV-2, SARS-CoV-1 and MERS-CoV viral load dynamics, duration of viral shedding and infectiousness – a living systematic review and meta-analysis". In: *medRxiv*. DOI: 10.1101/2020.07.25.20162107. eprint: https://www.medrxiv.org/content/early/2020/07/29/2020.07.25.20162107.full.pdf. URL: https://www.medrxiv.org/content/early/2020/07/29/2020.07.25.20162107.

Egeland, Olav and Jan Gravdahl (Jan. 2002). *Modeling and Simulation for Automatic Control*.

Gros, Sébastien (n.d.). *Numerical Optimal Control Lecture 8: Indirect Optimal Control*.

Martcheva, M. (Jan. 2015). "An Introduction to Mathematical Epidemiology". In: 61. DOI: 10.1007/978-1-4899-7612-3.

Wächter, Andreas and Lorenz Biegler (Mar. 2006). "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming". In: *Mathematical programming* 106, pp. 25–57. DOI: 10.1007/s10107-004-0559-y.

A Parameter Tables and Initial values

$\mathscr{R}_0$	6.5	N_{pop}	$5.3e^6$
α	$\frac{1}{9}$	I_0	$5e^5$
T	365	R_0	0
M	4	N	52
d	3	polynomial	Radau

Table 4: Default ODE and integrator parameter values

Initial trajectories for multiple-shooting are set to match the uncontrolled epidemic (Figure 2). Collocation-specific states are initialized to $\dot{x}_0$.

Objective/Parameter	u_{min}	u_{max}	W_u
Social Distancing	0.5	6.5	0.1
Vaccination	0	$0.02N_{pop}$	1
Isolation	0	0.2	1

Table 5: Control objective parameter values

$\Delta w_{tol}/S_{tol}$	$1e^{-3}$	τ_{tol}	$1e^{-6}$
β_τ	0.6	λ_0	1
μ_0	1	s_0	1

Table 6: Custom PMP/IPOPT parameter values (All λ, μ, s are set to λ_0, μ_0, s_0)