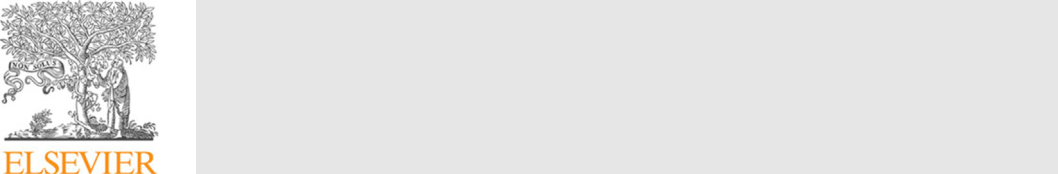
[Journal of Computational and Applied Mathematics 380 (2020) 112983](https://doi.org/10.1016/j.cam.2020.112983)



Contents lists available at [ScienceDirect](http://www.elsevier.com/locate/cam)



Journal of Computational and Applied

Mathematics

journal homepage: [www.elsevier.com/locate/cam](http://www.elsevier.com/locate/cam)

Direct and indirect optimal control applied to plant virus propagation with seasonality and delays



Benito M. Chen-Charpentier [∗](#page1), Mark Jackson

*Department of Mathematics, University of Texas at Arlington, Arlington, TX 76019-0408, United States of America*

a r t i c l e i n f o

*Article history:*

Received 1 February 2019

Received in revised form 1 May 2020

*Keywords:*

Plant virus propagation

Optimal control

Direct and indirect methods

a b s t r a c t

In many applications of mathematical modeling to biology, economics, social sciences and engineering, the objective is to find optimal solutions. Usually we want to minimize an objective function depending on a number of functions subject to constraints given, for example, by systems of differential equations. Two main numerical approaches are used to solve these optimal control problems, depending on whether the problem is optimized first and then discretized, or vice versa. Each of these two approaches has its advantages and disadvantages. In this paper we describe both methods an apply them to a plant virus propagation model, where the virus is propagated through a vector that bites the infected plants. The model includes delays due to the time the virus takes to infect the plant and the vector, and seasonality due to the dependence of the behavior on the seasons. The objective function is the total cost to a farmer of having infected plants, and includes the actual cost of a plant plus the cost of the controls which are insecticides and a predator species that preys on the insects. Numerical simulations are done using both methods and comparisons are made.

© 2020 Elsevier B.V. All rights reserved.

**1. Introduction**

Optimal control problems appear in many applications. For example, in controlling an epidemic of an infectious disease the rate of vaccination can be the optimal control to minimize the cost of the disease, in a vertical landing problem the optimal control can be the rate of de-acceleration to minimize fuel consumption, and in a tree harvesting problem the optimal control can be the rate of harvesting to maximize profit. Our problem of interest, as described in detail in Section [3](#page4) is a plant virus transmission problem with a vector propagating the virus with delays and seasonality, with two optimal controls, the amount of insecticide and or predators used to control the vectors, in order to minimize the total cost of the disease to the farmer. For many other applications in mathematical biology, engineering and economics see, for example, [[1](#page10)–[3](#page10)] and the references therein.

An Optimal Control Problem (OCP) is a constrained optimization problem with the constraint given by a dynamical system. The objective is to determine the state variables, the controls, the time independent parameters, and sometimes the initial and/or final times, subject to the state variables being the solution of a system of differential equations, path constraints and boundary conditions. The OCP usually needs to be solved numerically. There are two main approaches: use Pontryagin Maximum Principle [[4](#page10)] to first solve theoretically the optimization problem and then discretize this new problem numerically. A second, more recent approach, is to discretize the optimization problem and then find the optimal

∗ Corresponding author.

*E-mail addresses:* [bmchen@uta.edu](mailto:bmchen@uta.edu)(B.M.Chen-Charpentier),[mark.jackson@mavs.uta.edu](mailto:mark.jackson@mavs.uta.edu)(M.Jackson).

<https://doi.org/10.1016/j.cam.2020.112983>

0377-0427/© 2020 Elsevier B.V. All rights reserved.

* *B.M. Chen-Charpentier and M. Jackson / Journal of Computational and Applied Mathematics 380 (2020) 112983*

solution of this discrete problem [[5](#page10)]. The first method is known as an indirect method and the second as a direct method.

In this paper we evaluate, compare and apply both methods to a plant virus propagation problem.

Plants are a vital part of the planet ecosystem. They are a food source for man and many species. Many medicines are derived from plants. Their fibers are used for clothing and paper, and wood is an essential building material. Healthy plants are essential for a healthy environment. Unfortunately, just like humans and animals, plants are subject to many diseases, some caused by insects like beetles and many others by viruses like the tobacco mosaic virus and the tomato bushy stunt virus, that affect, not only the plants they are named after, but many other species. These viruses not only damage the plant but often kill it, and as a result, billions of dollars are lost every year because of crop loss due to virus caused diseases [[6](#page10)]. Most of the time, the virus is propagated by a vector, usually insects that bite infected plants, get themselves infected and then bite susceptible plants infecting them. Seasons alter the behavior of insects. They are very active in the warm months and not so active, almost dormant, in the cool months. To control the spread of the virus it is necessary to reduce the number of insects. The most common way is to use chemical insecticides. Unfortunately, these chemicals not only are expensive but also have toxic effects on humans, on animals and on the environment in general. An alternative approach is to use a predator species, such as birds or bats, to prey on the insects. These predator may be completely introduced or may already be naturally present and it may just be necessary to increase their number. Of course both control methods may be combined and the question is whether there is an optimal combination.

A mathematical model consisting of a system of ordinary and delay differential equations describes the interaction between plants, vectors and predators. This model can be used with constant coefficients or with periodic coefficients such as the infection and birth and death rates of the insects. This system of equations together with the initial conditions gives the constraints to the optimization problem. An objective function giving the cost to the farmer of the death plants and of the control methods, insecticide and predators, is introduced. The cost of the use of the insecticide may also include an environmental cost. Therefore we have an optimal control problem that may be solved by indirect or direct methods to determine the optimal amount of predators to introduce and insecticide to use. Of course, the additional constraints that populations and controls are non-negative need to be added.

In Section [2](#page2) the optimal control method is introduced, and both direct and indirect variations are explained. In Section [3](#page4) the virus propagation mathematical model is developed. The next section consists of numerical calculations using both methods applied to the virus propagation model. The last section is some conclusions.

**2. Optimal control**

An Optimal Control Problem (OCP) is a constrained optimization problem with the constraint given by a dynamical system. An optimal control problem is posed formally as follows: Determine the state (equivalently, the trajectory or path), *x*(*t*) ∈ R*n*, the control *u*(*t*) ∈ R*m*, the vector of static parameters *p* ∈ R*q*, the initial time, *t*0, and the terminal time,

* that optimize the objective function
* *T*

*L*(*x*, *u*, *p*)= *f* (*t*, *x*(*t*), *u*(*t*), *p*) *dt* +φ(*x*(*T* ))

*t*0

subject to

*x*′(*t*)= *g*(*t*, *x*(*t*), *u*(*t*), *p*),

the path constraints

*h*(*x*(*t*); *u*(*t*))≤0

and the boundary conditions

ψmin ≤ ψ(*x*(*t*0), *t*0, *x*(*T* ), *T* , *p*) ≤ ψmax.

The problem was presented using ordinary differential equations as the constraints but delay equations, partial differential equations, integral–differential equations and others may also be used.

There are three main methods of solving the OCP. The first one is Dynamic programming introduced by Bellman in 1952 [[7](#page10)] and later applied to calculus of variations and optimal control [[8](#page10),[9](#page10)]. It consists of discretizing then solving recursively the constrained optimization problem. But the two most common ones are indirect methods based on Pontryagin Maximum Principle [[4](#page10)] in which the problem is first optimized and then discretized, and direct methods that first discretize the problem using piecewise polynomials for both the state variables and the controls and then minimized numerically by finding the coefficients of the polynomials that minimize the discretized objective function [[5](#page10)].

*2.1. Indirect methods*

Indirect methods are based on Pontryagin Maximum Principle [[4](#page10),[10](#page10)]. To use this method, we need to determine the Hamiltonian which involves the integrand of the objective function and the inner product of the adjoint variables with the right hand side of the system of differential equations. Then the Hamiltonian is used to determine the adjoint equations,

|  |  |
| --- | --- |
| *B.M. Chen-Charpentier and M. Jackson / Journal of Computational and Applied Mathematics 380 (2020) 112983* | 3 |

the transversality condition at the end time, and the optimality condition. Let *u*∗ to be the optimal control functions and *x*∗be the corresponding optimal state values. Define the Hamiltonian by

*H*(*t*, *x*, *u*, λ)= *f* (*t*, *x*, *u*)+λ(*t*)· *g*((*t*, *x*, *u*)),

where λ*i*(*t*) is the adjoint for each *xi* for *i* = 1, . . . , *n*. Note *u*∗ maximizes *H*(*t*, *x*, *u*, λ). Moreover, *u*∗, *x*∗, and λ satisfy

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *xi*′(*t*)= | | ∂*H* | | = *gi*(*t*, *x*, *u*), *xi*(*t*0) = *xi*0 for *i* = 1, . . . , *n* | |  |  |
| ∂λ*i* | |  |  |
| λ*j*′ |  |  | ∂*H* | | |  |  |
| = − | |  | | , λ*j*(*T* ) = φ*xj* (*x*(*T* )) for *j* = 1, . . . , *n* | (1) |  |
| ∂*xj* | |  |

0 = ∂*H* at *u*∗ for *k* = 1, . . . , *m* (optimality condition)

∂*uk* *k*

These conditions determine our optimal solution.

**Pontryagin Maximum Principle** If*u*∗(*t*) and*x*∗(*t*) are optimal for the OCP, then there exists a piecewise differentiableadjoint variable λ(*t*) such that

*H*(*t*; *x*∗(*t*), *u*(*t*), λ(*t*))· *H*(*t*, *x*∗(*t*), *u*∗(*t*), λ(*t*))

for all controls *u* at each time *t*, where the Hamiltonian *H* is

*H* = *f* (*t*, *x*(*t*), *u*(*t*))+λ(*t*)*g*(*t*, *x*(*t*), *u*(*t*));

and

λ′ = −∂*H* , λ*j*(*T* ) = φ*x* (*x*(*T* )) for *j* = 1, . . . , *n*. □

*j* ∂*xj* *j*

The system of equations for *x*(*t*), *u*(*t*) and λ(*t*) needs to be solved numerically. A common method [[1](#page10),[11](#page10)] consists of solving sequentially the system of equations using what is known as the forward–backward sweep: First an initial guess of each control variable is made. Then all states *xi* are simultaneously solved forward in time using an Euler, Runge– Kutta method or any other integrator [[12](#page10)]. All adjoints λ*i* are simultaneously solved backward in time using the same integrator. If the integration points are not the same as the ones used for the forward integration, an interpolation needs to be done. Then each control *u*∗*i* is updated subject to its individual characterization using a Picard or similar iteration for the optimality condition. The process is repeated until convergence occurs. this method involves two iterations. The first one is the iteration between determination of the state variables, the adjoint variables and the control variables. The second is the iteration procedure to estimate the control variables using the optimality condition. As for any iterative procedure the convergence depends on the initial guess. A second implementation of the indirect method is to solve the equations for the state variables and for the adjoint simultaneously as a two-point boundary value problem. This can be done by using a finite difference discretization [[13](#page10)] or by a shooting method [[14](#page10)]. Then solving for the controls using the optimality condition and finally iterating between the two-point boundary value problem and the optimality condition until convergence. This second method is more computationally expensive. A third method is to use finite differences, or an, and solve the other local approximation of the state, adjoint and control variables, and solve for all variables simultaneously. This can be done using a Newton-like method [[12](#page10)] or even a nonlinear programming method [[15](#page10)].

The advantages of the indirect method are that it is easy: to implement, especially if the forward–backward iteration is implemented with a numerical integrator with uniform step size. It can have high accuracy if a high order integrator is used together with a small uniform step size. Usually Runge–Kutta of order 4 is a good choice. The Picard iteration is also cheap. The other versions of the indirect method are more expensive and the third one, as will be clear below, is more expensive than a direct method since it also requires the calculation of the adjoint variables.

But even the first version of the indirect method has some disadvantages. Among them are that it may be difficult to derive ∂*H*/∂*x* and ∂*H*/∂*u*, but a computer algebra program may help. Path constraints are difficult to implement and there may be problems with adjoint equations that may be ill-conditioned. But the main disadvantage is that it may be difficult to obtain convergence. The method may converge to a local minimum or not converge at all depending on what is the initial guess for the control functions. Solving together the equations for the states and the adjoint variables as a two-point boundary value problem may help.

Indirect methods have been the method of choice in mathematical biology problems but sometimes convergence is hard to obtain.

*2.2. Direct methods*

Direct Methods of Optimal Control are based on the following principle: Discretize the optimal control problem, then apply Nonlinear Programming (NLP) techniques to the resulting finite-dimensional optimization problem. They are not new but due to the difficulty of solving discrete optimization problems in many variables they have only been studied extensively over the last 30 years. They have proven successful for many complex applications. Their strength is that

* *B.M. Chen-Charpentier and M. Jackson / Journal of Computational and Applied Mathematics 380 (2020) 112983*

the take advantage of the power of state-of-the-art NLP solvers. There is no need for additional theory for different types of constraint equations, so they can be applied to systems of ordinary differential equations, discrete systems, differential–algebraic equations, partial differential equations, delay equations, etc.

The direct methods approximate the solution to the OCP by first discretizing the problem and then optimizing the discrete problems. There are two main variations, direct collocation methods and direct sequential methods. Both start by parameterizing the control by first subdividing the optimization horizon [*t*0, *T* ] into *ns* ≥ 1 control stages, *t*0 < *t*1 <

* · · < *tns* = *T* . Then in each subinterval [*tk*−1, *tk*] *u*(*t*) = *Uk*(*t*) is approximated using Lagrange interpolating polynomials of degree M. If there are more than one controls, the approximation is done for each control. The direct collocation method will convert the OCP into a finite-dimensional nonlinear programming problem (NLP) through discretization of not only the control but also state variables, *x*(*t*) = *Xk*(*t*), *tk*−1 ≤ *tk*, *k* = 1, . . . , *ns*, using Lagrange polynomials of degree N. The direct sequential method converts the OCP into a finite-dimensional NLP through discretization of the control variables only, while the differential equations are embedded in the NLP problem. That is, both types of direct methods minimize the objective function with respect to the coefficients of the collocation of the control *u* subject to the dynamic, path and boundary condition constraints. The dynamic constraints can be solved using a standard integrator for ordinary differential equations or for delay differential equations depending on the type of equations determining the constraints. The success of directs methods depends on their use of a robust, reliable and efficient NLP solver.

The advantages of direct methods is that no new theory needs to be developed for different types of dynamical constraints. Path constraints are easily accommodated by enforcing inequality constraints at the collocation points. The main disadvantage is the need to solve a very large-scale NLP problems in the variables which are the coefficients of the Lagrange polynomials, However there are very robust NLP solvers such as IPOPT which implements a primal–dual interior point algorithm. For information about the algorithm see [[16](#page10),[17](#page10)]. Another disadvantage is that the number of time stages and collocation points as well as the position of the collocation points for the state variables must be chosen a priori. In indirect methods an adaptive integration method may be used.

Even though there are several commercial and open source packages available that implement direct methods, they are not widely used in mathematical biology optimal control applications.

There are three important open source software packages for direct methods, CasADi (<https://web.casadi.org>), ACADO toolkit (<https://acado.github.io/>), and BOCOP (<http://www.bocop.org/>).

CasADi utilizes Python and MATLAB/Octave. It is a general-purpose tool for gradient-based numerical optimization with a strong focus on optimal control. Implements several variations of direct methods. The documentation is short and only a few examples using one control and ordinary differential equations are given. So it is not trivial to extend to several controls and delay equations [[18](#page10)]. ACADO toolkit utilizes C++ and Matlab. It is a software environment and algorithm collection for automatic control and dynamic optimization. Easy to use with multiple controls but no delay equation integrator [[19](#page10)]. Can also be used for parameter estimation and sensitivity analysis. BOCOP uses C++ and includes a GUI. Uses direct sequential method. BOCOP software that discretizes equations and the variables using a method of the user’s choice. Solves problems with multiple controls and the examples illustrate this as well as how to integrate ordinary differential and delay equations, but one has to write several routines with the examples also showing how to do it [[5](#page10)]. All three packages use the NLP solver IPOPT. Since the most critical part of solving an OCP with a direct method is the large-scale nonlinear optimization routine and since IPOPT is open source [[20](#page10)], the researcher need just implement the discretization method that better fits the particular OCP.

Direct methods have the advantage over indirect methods in that they are more straightforward to apply and more robust with respect to the initialization. The cost, however, is that some precision may be lost due to limitations in the number of optimization variables [[5](#page10)]. BOCOP first discretizes the dynamics equations and state and control variables using a method of the users choice. It then utilizes the NLP solver IPOPT which implements a primal–dual interior point algorithm. For information about the algorithm see [[17](#page10)]. All three open source. Commercial packages were not considered due to their cost.

**3. Mathematical model**

The mathematical model for plant virus propagation is an extension of the models presented in [[21](#page10)–[23](#page10)] and [[24](#page10)] and has the structure of a vector-based epidemiological model. It includes 6 populations: susceptible plants *S*(*t*), infected plants *I*(*t*), recovered plants *R*(*t*), susceptible insect vectors *X* (*t*), infected insect vectors *Y* (*t*), and predators *P*(*t*). Each variable gives the value of the respective population at time *t*. Susceptible plants do not have the disease but could contract the disease if infected with the virus through an infected vector. The infected plants have the virus but cannot directly transmit the virus to susceptible plants. Infected plants can either die from the disease or recover since plants have defense mechanisms. Additionally, since the infected plants can die from the viral infection their death rate is higher than that of susceptible plants. For simplicity we also assume that as soon as a plant dies either from the infection or from a natural death, it is immediately replaced with a new susceptible plant by a farm worker. This assumption means that the total plant population remains constant with a value denoted by *K* . This simplification has the modeling advantage that

* = *S*(*t*)+*I*(*t*)+*R*(*t*) can be used to eliminate the recovered population and its corresponding conservation equation from the system of equations. The susceptible insects do not have the virus but can obtain the virus if they come in contact with a infected plant by biting it. Infected insects can transmit the virus to susceptible plants upon contact by bite. We

|  |  |
| --- | --- |
| *B.M. Chen-Charpentier and M. Jackson / Journal of Computational and Applied Mathematics 380 (2020) 112983* | 5 |

**Table 1**

Gives a list of the parameters used in the model, their description and their value or range. The parameters were used from the ranges given in [[24](#page10)].

|  |  |  |
| --- | --- | --- |
| Parameter | Description | Value |
|  |  |  |
| *K* | Total plant host population | 63 |
| β | Biting rate of plants due to vectors | 0.01 |
| β1 | Biting rate of vectors due to plants | 0.01 |
| α | Saturation constant of plants due to vectors | 0.2 |
| α1 | Saturation constant of vectors due to plants | 0.1 |
| µ | Natural death rate of plants | .01 |
| *m* | Natural death rate of vectors | .2974 |
| γ | Recovery rate of plants | 0.01 |
| Λ | Replenishing rate of vectors | 10 |
| *d* | Death rate of infected plants due to the disease | 0.2 |
| *c*1 | Contact rate between predators and healthy insects | 0.05 |
| *c*2 | Contact rate between predators and infected insects | 0.05 |
| δ | Natural death rate of predators | 0.05 |
| ϵ | Competition constant between predators | 0.01 |
| α3 | Saturation of predators due to insects | 0.01 |
| α4 | Conversion rate of predators due to insects | 0.01 |
| *din* | Death rate due to insecticide | 0–0.9 |
| Λ*p* | Rate at which predators are added | 0-10 |
| *G* | Cost of Insecticide | 0–1000 |
| *A* | Cost of Infected Plant | 0–100 |
| *F* | Cost of Predators | 0–1000 |
|  |  |  |

assume no vertical transmission of the virus with neither plants nor vectors. Moreover, we assume that the virus does not harm the vector and thus the vector does not need defence against the virus and it retains the virus for the rest of its life. So there are recovered vectors and the death rate of both susceptible and infected vectors is the same, including death by insecticide or predator. A predator does not get sick with the virus even it consumes an infected vector and it does not propagate the virus. The model includes two delays given by the time it takes the virus to spread in a plant or in a vector. Additionally, the contact rates between insect and plant change depending on the season and are higher in warmer months than cooler ones. Thus, we also assume the contact rates to be yearly periodic. We do the same for the vector growth rates. We will also include competition between predators for the insects. The interaction between vector and plant as well as that of predator and vector are assumed to have a limitation of the form of predator–prey Holling type 2.

The mathematical model is given by the following systems of delay differential equations:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *dS* | |  | µ | |  |  |  |  |  |  |  |  | β(*t*)*Y* (*t* − τ1) | | | | | | | | | | | | |  |  |  | τ |  |  |  |  |
|  |  |  | = |  |  | (*K* − *S*) + *dI* − | | | | | | | |  |  |  | | | | |  | |  | | | *S*(*t* − | | | |  | 1) | |  |  |
|  | *dt* | |  |  | 1 + α*Y* (*t* − τ1) | | | | | | | | | | | |  |  |  |
|  |  | *dI* |  |  | β(*t*)*Y* (*t* − τ1) | | | | | | | |  |  |  |  |  |  | µ |  |  |  |  | γ |  |  |  |  |  |  |  |  |  |  |
|  |  |  | = | |  | | | |  |  |  | *S* | − (*d* + | | | | | |  | + | | | |  | )*I* | |  |  |  |  |  |  |  |  |
|  |  | *dt* | 1 + α*Y* (*t* − τ1) | | | | | |  |  |  |  |  |  |  |  |  |  |
| *dX* | | |  | Λ | | |  | β1(*t*)*I*(*t* − τ2) | | | | | | | |  |  |  |  | τ | |  |  |  |  |  |  | *c*1*X* | | |  |  | (2) |  |
| *dt* | | | = | − | 1 + α1*I*(*t* − τ2) *X* (*t* − | | | | | | | | | | | | 2) − | | | | | 1+α3*X P* − *mX* − *dinX* | | | | | |  |
|  |  |  |  |  |  |
| *dY* | | |  |  | β1(*t*)*I*(*t* − τ2) | | | | | | | |  |  |  |  | τ |  |  |  |  |  |  | *c*2*Y* | | | | | |  |  |  |  |  |
|  | | | = | |  | | | |  |  | | *X* (*t* − | | | | |  | 2) − | | | | |  | | | | | | *P* − *mY* − *dinY* | | | |  |  |
| *dt* | | | 1 + α1*I*(*t* − τ2) | | | | | | |  | 1+α3*Y* | | | | | |  |  |
| *dP* | | |  |  |  |  |  |  | α4*c*1*X* | | | |  |  |  | α4*c*2*Y* | | | |  | *P*−δ*P*−ϵ*P*2 | | | | | | | | | | | |  |  |
|  |  | | = Λ*p* + | | | | | |  | *P* + | | | | |  | | | | | |  |  |
|  | *dt* | | 1+α3*X* | 1+α3*Y* | | | | | |  |  |

where

* ( ))

2π*t*

β(*t*) = β 1 + *h* cos

365

* ( ))

2π*t*

β1(*t*) = β1 1 + *h* cos .

365

We used *h* = 0 for no seasonality and *h* = .5 when including seasonality.

[Table 1](#page5) gives a list of the parameters used in the model, their description and their value or range. The parameters were used from the ranges given in [[24](#page10)].

The objective function is given in Lagrange form so the optimal control problem is

* *T*

min *AI*(*t*)2+ *Gdin*(*t*)2+ *F* Λ*p*(*t*)2*dt*, (3)

*din*,Λ*p* 0

subject to the dynamical constraints. Here *A*, *G* and *F* are the relative costs of death plants, insecticide and predators.

* *B.M. Chen-Charpentier and M. Jackson / Journal of Computational and Applied Mathematics 380 (2020) 112983*

**4. Numerical simulations**

In order to test and compare the indirect and direct methods consisting of expression [(3)](#page5) subject to the constraints given by system [(2)](#page5) we will do two cases. The first one consists of only one control, the amount of insecticide *u*(*t*) = *din*, and Λ*p* is constant. Furthermore the model has no delays, τ = 0, τ1 = 0, and no seasonal effects, that is, β(*t*) and β1(*t*) both constant (*h* = 0). The second case will have the two controls, *u*(*t*) = *din* and v(*t*) = Λ*p*, and for the third case the model will also include delays and seasonality.

*4.1. Case I*

We start with the indirect method based on Pontryagin Maximum Principle. The Hamiltonian for the system is

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *H* =*AI*(*t*)2 | | | | + *Gdin*(*t*)2 + λ1(µ(*K* − *S*) + *dI* − | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |  | β*Y* | | | |  |  |  |  |  |  |  |  | β*Y* | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |
|  | |  | | | *S*)+λ2( | | | | | |  |  |  | |  |  |  |  | *S* −(*d* +µ+γ)*I*)+ | | | | | | | | | | |  |  |
| 1+α*Y* | | | | | 1+α*Y* | | | | | | | |  |  |
|  |  |  |  |  |  |  | β1*I* | | | | | | | |  |  |  | *c*1*X* | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | β1*I* | | | | | | | |  |  |  |  | *c*2*Y* | | | | | |  |  |
|  | λ3(Λ − | | | |  |  |  |  | *X* − | | | | | |  |  |  |  |  |  |  | *P* − *mX* − *din*(*t*)*X* )+λ4( | | | | | | | | | | | | | | | | | | | | | |  | |  |  | |  |  |  | *X* − | | | |  |  |  | |  | *P* − *mY* − *din*(*t*)*Y* )+ | | |  |  |
|  | 1+α1*I* | | | | 1+α3*X* | | | | | | | 1+α1*I* | | | | | | | 1+α3*Y* | | | | |  |  |
|  |  |  |  |  |  |  | α4*c*1*X* | | | | | | | |  |  |  | α4*c*2*Y* | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | λ5(Λ*p* + | | | | | |  |  | |  | *P* + | | | | | | |  |  |  |  | | *P*−δ*P*−ϵ*P*2) | | | | | | | | | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  | 1+α3*X* | | | | 1+α3*Y* | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| The adjoint equations are | | | | | | | | | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| λ1′ |  | ∂*H* | |  |  |  |  |  |  |  |  |  |  | β*Y* | | | | | | | | | |  |  |  | β*Y* | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| = |  |  | = λ1(−µ − | | | | | | | |  |  |  | |  |  | |  | )+λ2 | | | | | |  | | | | | | |  | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| ∂*S* | 1+α*Y* | | | | | | | | 1+α*Y* | | | | | | |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| λ2′ | = | ∂*H* | | = 2*AI* + λ1*d* − λ2(*d* + µ + γ ) − λ3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |  |  | β1*X* | |  |  | − λ4 | | | | |  |  |  | β1*X* | | | |  |  |  |  |  |  |  |  |  |  |  |  |
| ∂*I* |  | (1 + α1*I*)2 | | | | | | (1 + α1*I*)2 | | | | | | | | | |  |  |  |  |  |  |  |  |  |
| λ3′ | = | ∂*H* | | = λ3(− | | | | β1*I* | | | | | | |  | − | | |  |  | *cP* | | |  |  |  |  |  | − *m* − *din*) + λ4 | | | | | | | | | | | | | |  | β1*I* | | | |  |  | + λ5 | | | | |  |  |  | *c*α4*P* | | | | | | (4) |  |
| ∂*X* |  | 1+α1*I* | | | | | | |  | (1 + α3*X*)2 | | | | | | | | | | 1+α1*I* | | | | | | | (1 + α3*X*)2 | | | | | | |  |  |  |
| λ′ |  | ∂*H* |  |  |  |  | −λ1β*S* | | | | | |  |  |  |  |  | λ2β*S* | | | | | |  |  |  |  | λ | |  |  |  |  |  |  |  |  |  |  |  |  |  |  | *cP* | | | |  |  |  |  |  |  | λ | | |  |  |  | *c*α4*P* | | |  |  |  |
| = ∂*Y* = (1+α*Y*)2 + (1+α*Y*)2 + | | | | | | | | | | | | | | | | | | | | | | | | | | | 4(−*m* − *din* | | | | | | | | | | − (1+α3*Y*)2)+ | | | | | | | | | | | | | | 5((1 + α3*Y*)2 ) | | | | | | |  |  |
| 4 |  |  |  |  |  |  |  |
| λ′ |  | ∂*H* |  |  |  | −λ3*cX* | | | |  | | | |  |  | λ4*cY* | | | | |  | | | λ | | |  |  |  |  | δ |  |  |  |  |  | ϵ |  |  | α4*cX* | | | | | |  | |  |  | α4*cY* | | | | | | |  |  | . | |  |  |  |  |  |
| = ∂*P* | | | = 1+α3*X* − 1+α3*Y* + | | | | | | | | | | | | | | | | | | | | 5(− | | | | − | | | 2 | | *P*+ 1+α3*X* + 1+α3*Y*) | | | | | | | | | | | | | | | | | | | | |  |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |  |  |  |

We must satisfy the optimality condition

* *H*

= 2*dinG* − λ3*X* − λ4*Y* = 0.

∂*din*

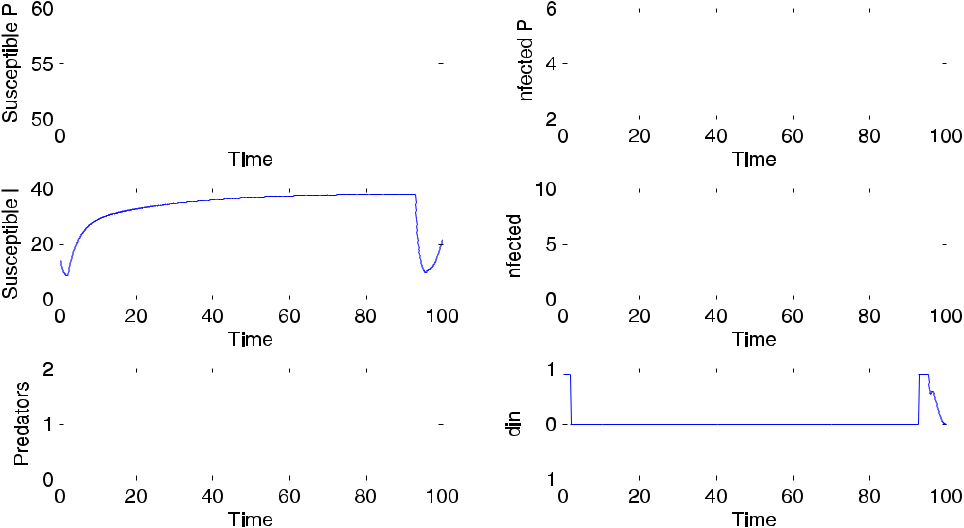
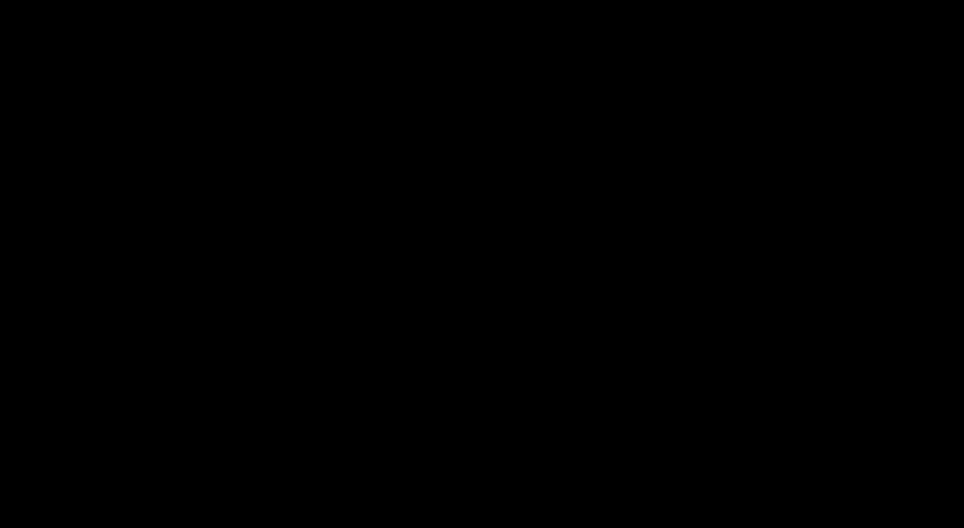
The forward–backward sweep method for several variables used to solve the previous differential equations [(2)](#page5) and

[(4)](#page6). We used both Euler and Runge–Kutta method of order 4 for the time integration. An initial guess the control variable is made and then all states *xi* are simultaneously solved forward in time. Afterwards all adjoints λ*i* are simultaneously solved backward in time. The control *u*∗*i* is updated subject to its individual characterization using a Picard-like iteration. The process is repeated until convergence occurs. The simulation used Λ*p* = 1. [Fig. 1](#page7) shows the numerical results using the above method, for the case when the relative cost of a death plant is the same as the cost of the insecticide to kill an insect (*A* = *F* = 1).

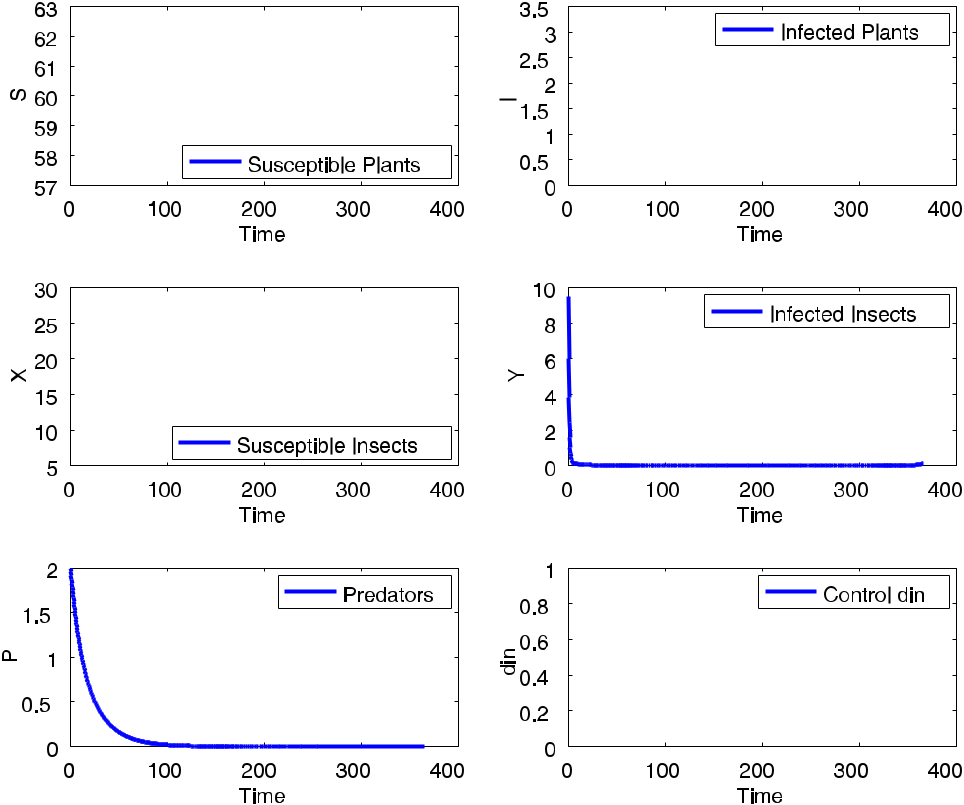
Notice that since insecticide is present, both infected populations are smaller than without the insecticide, as expected. However, the insecticide fluctuates drastically at the end of the time interval. The method had convergence difficulties to the global minimum and reached a local minimum instead. This illustrates one of the problems using an indirect method. Changing the initial guess for the control did not improve the convergence. Furthermore for times greater than 100 days the method stopped converging. Another problem with using this method is that the numerical method to solve the system of constraints and adjoints is highly sensitive to parameters and time interval. We had difficulties getting the indirect method to converge for times larger than 100 days. The indirect method based on the forward–backward method involves two iterations: the iteration to solve the equation for the control, and the iteration between determining the state variables, the adjoint variables and the control. Solving for the state variables, the adjoint variables and the control with more accuracy does not improve the convergence. The difficulty is that the convergence depends on the initial guess for the control. As the time interval increases, it gets more difficult to find a good starting value. In this case we could not get a good starting guess for times greater than 100 days. But for times shorter than 100 days the method converged well and as expected it was easy to program and could be made very accurate since the solution of the differential equations is inexpensive.

Of the three open source packages, ACADO is the easiest to use following the examples in the manual even for multiple controls. But it does not have an integrator for delay differential equations. BOCOP is the one that includes examples with both multiple controls and delay differential equations. The necessary routines can be written by easily modifying the examples. One possible difficulty is that it is better to run it in linux since it runs in C++, but it also runs in windows.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *B.M. Chen-Charpentier and M. Jackson / Journal of Computational and Applied Mathematics 380 (2020) 112983* | | | | | | | | | | 7 |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |



**Fig. 1.** The above graphs show the dynamics of the system as well as the optimal function for the insecticide*di**n*in the interval [0,0.9]. The timeinterval is from 0 to 100 days using Pontryagin Maximum Principle.



**Fig. 2.** The above graphs show the dynamics of the system as well as the optimal function for the insecticide*di**n*in the interval [0,0.9] and timebetween 0 and 400 days using BOCOP.

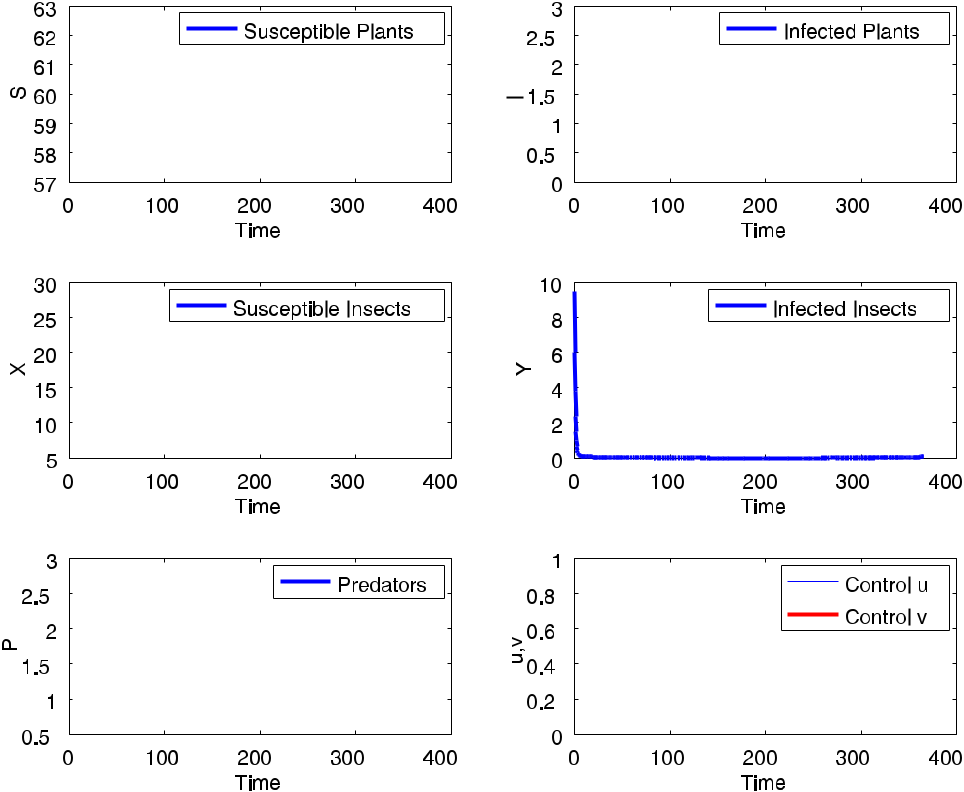
BOCOP only uses an objective form in Mayer form. Ours is a Lagrange form since we minimize an objective function of the form

* *t*1

*f* (*t*, *x*(*t*), *u*)(*t*)*dt*.

*t*0

* *B.M. Chen-Charpentier and M. Jackson / Journal of Computational and Applied Mathematics 380 (2020) 112983*



**Fig. 3.** The above graphs show the dynamics of the system as well as the optimal functions for the insecticide*di**n*and the amount of predatorsv(*t*) in the interval [0,0.9] and time between 0 and 400 days using BOCOP.

But by defining a new ‘state’ variable

*x*′*n*+1= *f* (*t*, *x*(*t*), *u*(*t*))

*xn*+1(*t*0)=0,

where *n* is the number of original state variables, we obtain the new state equation

* *t*

*xn*+1(*t*)= *f* (*t*, *x*(*t*), *u*(*t*))*dt* + *xn*+1(*t*0).

*t*0

So minimizing *J* = *xn*+1(*T* ) gives the Mayer form of the functional to be minimized.

All three packages produce the same optimal solution with small variations in accuracy since they use the same NLP solver, IPOPT. Therefore we chose BOCOP and its implementation of the direct method as the method to use.

Using BOCOP, to discretize our problem, we chose an implicit Gaussian integrator of order 4 with 1000 subintervals between time 0 and time 365 (in days). The discretization was then passed to IPOPT optimizer, and [Fig. 2](#page7) shows the numerical results. Notice that there is no jump around 100 days.

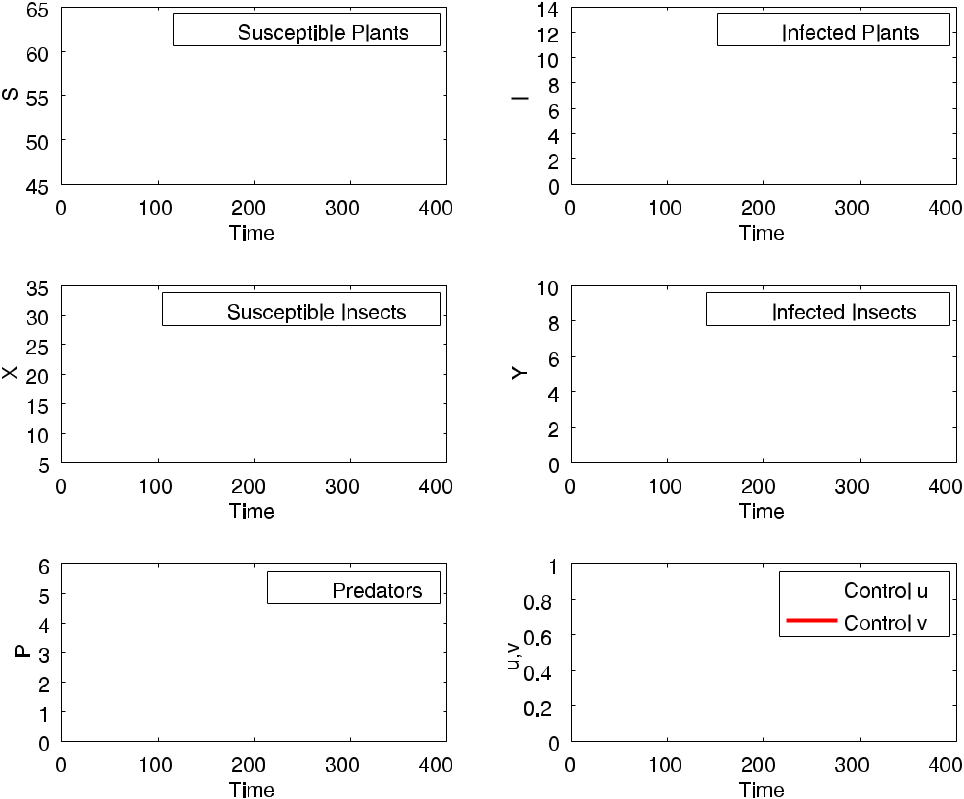
*4.2. Case II*

As a second case we consider the model with no delays, τ = 0, τ1 = 0, or seasonality, β(*t*) and β1(*t*) both constant (*h* = 0), but with two controls. The first control is the amount of insecticide *u*(*t*) = *din*, and the second control is the amount of predators introduced v(*t*) = Λ*p*. We consider the case when all three costs have the same value (*A* = *F* = *G* = 1). Since the OCP is more complicated than in Case I, we will only use BOCOP as the direct solver. [Fig. 3](#page8) shows the plots for all populations and for the two controls. The use of the amount of introduced predators as a second optimal control reduces the number of infected plants and insects faster than for Case I.

*4.3. Case III*

Now we consider the full mathematical model including delays and seasonality, and the two controls, *u*(*t*) = *din* and v(*t*) = Λ*p*. This is a more realistic model and offers greater flexibility by using two controls. Since the direct methods

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *B.M. Chen-Charpentier and M. Jackson / Journal of Computational and Applied Mathematics 380 (2020) 112983* | | | | | | | | | | | | | 9 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |



**Fig. 4.** The above graphs show the dynamics of the system as well as the optimal function for the insecticide*di**n*in the interval [0,0.9] and timebetween 0 and 400 days using BOCOP.

converged with no difficulty, we only used BOCOP for this case, using the same integrator as number of points as in Case I. We consider the case when all three costs have the same value (*A* = *F* = *G* = 1) and take τ = 24 and τ1 = 1, both delays in days. The numerical results are given in [Fig. 4](#page9). Note that the number of infected plants and insects decreases faster than in Case I. The amount of the insecticide affects directly the number of insects. The oscillations in this control are reflected in oscillations in the number of susceptible and infected insects. The magnitude of these oscillations is small compared to the corresponding population values. Since by the time the oscillations appear, the number of infected insects is very small, the effect of the oscillations on the plant populations is very small and cannot be seen in the graph. The same is true for the oscillations of the infected insect population. Again BOCOP did not have convergence issues even though the number of optimization variables is very large. Lack of convergence to the global optimal point is generally due to the use of a not very good initial guess. The larger the dimension of the search space the harder it is to get this good initial guess. So, if there were difficulties in the convergence, we could have reduced the number of time subintervals and, therefore the number of unknown coefficients in the expansion of controls and of the state variables would be decreased. Also the order of the Lagrange polynomials used could be reduced, especially for the controls since in practice the controls would very likely be varied as piecewise constants with a small number of jumps. With less unknowns it is easier to get good starting values. And, if necessary, we could do this reduction again until BOCOP converged and then used an interpolation procedure to obtain a good initial guess for the desired number of subintervals and order of the expansions for the state variables and the controls.

**5. Conclusions**

In the field of mathematical biology most optimal control problems are approximately solved using indirect methods based on Pontryagin Maximum Principle. The reasons are that they are widely known, easy to implement and very accurate. But they may have convergence problems that may be solved by changing the integrator and by trying different initial guesses for the controls. But it may be time consuming to do so and success is not guaranteed. Also the theory is well developed for problems where the dynamical system giving the constraints are ordinary differential equations but not so for other types of equations. Also path constraints, if present, are not trivial to impose. Direct methods are harder to implement even if using an available NLP solver. But existing open source packages such as BOCOP make it relatively easy. The main strength of the direct methods is their convergence properties based on the robustness of the IPOPT method. For

1. *B.M. Chen-Charpentier and M. Jackson / Journal of Computational and Applied Mathematics 380 (2020) 112983*

the plant virus propagation model, indirect methods work well for short time intervals but have convergence problems for times of 100 days or larger. The direct methods have no convergence problems for our time interval of interest of one year. So, in summary, we conclude that for an optimal control problem start with and indirect method but if convergence problems arise switch to an indirect method, maybe using a package such as BOCOP.

**References**

1. [S. Lenhart, J.T. Workman, Optimal Control Applied to Biological Models, Crc Press, 2007.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb1)
2. [D.G. Hull, Optimal Control Theory for Applications, Springer Science & Business Media, 2013.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb2)
3. [T.A. Weber, et al., Optimal control theory with applications in economics, in: MIT Press Books, vol. 1, The MIT Press, 2011.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb3)
4. [L. Pontryagin, V. Boltyanskii, R. Gamkrelidze, E. Mishchenko, The Mathematical Theory of Optimal Processes, Ser. Translated from the Russian](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb4) [by KN Trirogoff; edited by LW Neustadt, Interscience Publishers John Wiley & Sons, Inc. New York-London, 1962.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb4)
5. I.S. Team Commands, BOCOP: an open source toolbox for optimal control, 2017, <http://bocop.org>.
6. [V. Nicaise, Crop immunity against viruses: outcomes and future challenges, Front. Plant Sci. 5 (2014) 660.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb6)
7. [R. Bellman, On the theory of dynamic programming, Proc. Natl. Acad. Sci. 38 (8) (1952) 716–719.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb7)
8. [R. Bellman, Dynamic programming and Lagrange multipliers, Proc. Natl. Acad. Sci. 42 (10) (1956) 767–769.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb8)
9. [R.E. Bellman, S.E. Dreyfus, Applied Dynamic Programming, vol. 2050, Princeton university press, 2015.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb9)
10. [I.M. Ross, A Primer on Pontryagin’s Principle in Optimal Control, vol. 2, Collegiate publishers San Francisco, 2015.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb10)
11. [S. Anita, V. Capasso, V. Arnautu, An Introduction to Optimal Control Problems in Life Sciences and Economics: From Mathematical Models to](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb11) [Numerical Simulation with MATLAB⃝R , Springer, 2011.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb11)
12. [D. Kincaid, D.R. Kincaid, E.W. Cheney, Numerical Analysis: Mathematics of Scientific Computing, vol. 2, American Mathematical Soc., 2009.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb12)
13. [H.B. Keller, Numerical Methods for Two-Point Boundary-Value Problems, Courier Dover Publications, 2018.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb13)
14. [U.M. Ascher, L.R. Petzold, Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, vol. 61, Siam, 1998.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb14)
15. [L.T. Biegler, Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes, vol. 10, Siam, 2010.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb15)
16. [J. Nocedal, A. Wächter, R.A. Waltz, Adaptive barrier update strategies for nonlinear interior methods, SIAM J. Optim. 19 (4) (2009) 1674–1693.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb16)
17. [A. Wächter, L.T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, Math.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb17) [Program. 106 (1) (2006) 25–57.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb17)
18. [J.A. Andersson, J. Gillis, G. Horn, J.B. Rawlings, M. Diehl, CasADi: a software framework for nonlinear optimization and optimal control, Math.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb18) [Program. Comput. (2018) 1–36.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb18)
19. [B. Houska, H.J. Ferreau, M. Diehl, ACADO toolkit—An open-source framework for automatic control and dynamic optimization, Optim. Control](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb19) [Appl. Methods 32 (3) (2011) 298–312.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb19)
20. [A. Wächter, L.T. Biegler, Line search filter methods for nonlinear programming: Local convergence, SIAM J. Optim. 16 (1) (2005) 32–48.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb20)
21. [M. Jackson, B.M. Chen-Charpentier, Modeling plant virus propagation with delays, J. Comput. Appl. Math. 309 (2017) 611–621.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb21)
22. [M. Jackson, B.M. Chen-Charpentier, A model of biological control of plant virus propagation with delays, J. Comput. Appl. Math. 330 (2018)](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb22) [855–865.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb22)
23. [M. Jackson, B.M. Chen-Charpentier, Modeling plant virus propagation with seasonality, J. Comput. Appl. Math. 345 (2019) 310–319.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb23)
24. [R. Shi, H. Zhao, S. Tang, Global dynamic analysis of a vector-borne plant disease model, Adv. Difference Equ. 2014 (1) (2014) 59.](http://refhub.elsevier.com/S0377-0427(20)30274-0/sb24)