EXAMINATION OF SOLVING OPTIMAL CONTROL PROBLEMS WITH DELAYS USING GPOPS-II

JOHN T. BETTS
Applied Mathematical Analysis
2478 SE Mirromont Pl.
Issaquah, WA, 98027, USA

STEPHEN CAMPBELL AND CLAIRE DIGIROLAMO
Department of Mathematics
North Carolina State University
Raleigh, NC, 27695-8205, USA

(Communicated by Kok Lay Teo)

Abstract. There are a limited number of user-friendly, publicly available optimal control software packages that are designed to accommodate problems with delays. GPOPS-II is a well developed MATLAB based optimal control code that was not originally designed to accommodate problems with delays. The use of GPOPS-II on optimal control problems with delays is examined for the first time. The use of various formulations of delayed optimal control problems is also discussed. It is seen that GPOPS-II finds a suboptimal solution when used as a direct transcription delayed optimal control problem solver but that it is often able to produce a good solution of the optimal control problem when used as a delayed boundary value solver of the necessary conditions.

1. Introduction. Delays occur naturally in a variety of dynamical systems and finding the solution of optimal control problems (OCP) with dynamics that have delays (OCPD) is an important task. For simple problems one can sometimes use the method of steps (MOS) or control parameterization to get a solution. But for more complex problems, with constraints and nonlinearities, solution can be more challenging. MOS, for example, creates very large problems if the time interval is long with respect to the delay or if there are several delays. Control parameterization may require development of sophisticated grid refinement strategies [4, 8]. MOS and control parameterization are explained later.

There are many types of optimization problems concerning delayed systems. One important class of problems is when the optimization is over a finite set of parameters which include system or model parameters and switching times [17, 18, 19, 22]. We do not consider that type of problem here. Another common approach is some variant of control parameterization where the optimization is over a set of parameters defining the control. One common choice is piecewise linear which may be combined

2010 Mathematics Subject Classification. Primary: 49M37; Secondary: 34K28, 49M25.
Key words and phrases. Optimal control, delayed dynamics, waveform relaxation, direct transcription, control parameterization, SOS, GPOPS-II.

* Corresponding author: Stephen Campbell.
with user provided gradients [15, 20, 21]. While effective on specific problems it can require more expertise of users on large scale industrial problems.

When solving optimal control problems a direct transcription approach can be very desirable since these types of codes can often solve very complex industrial problems [3, 27]. In direct transcription the control is parameterized on the same grid as the dynamics and the control that results from the parametrization is higher order than piecewise linear. The resulting nonlinear programming problem (NLP) is solved by an optimizer. The solutions are iteratively refined. This grid refinement in industrial strength direct transcription codes is very sophisticated and makes their use highly attractive on more complex problems.

The direct transcription code SOSD, which is part of the Sparse Optimization Suite (SOS), is written in Fortran and designed to accommodate multiple state and control delays [5, 6]. SOSD has its own custom interior point and SQP methods to solve the NLP problem on each iteration. The direct transcription MATLAB code GPOPS-II has been used to successfully solve a number of challenging OCP [11, 25, 27]. GPOPS-II uses pseudospectral methods rather than the Hermite-Simpson discretization of SOSD. Pseudospectral methods have been used successfully on OCPD, we note only [23, 26, 30]. GPOPS-II was not designed with delays in mind. However, it turns out that because of the way that problems are formulated in GPOPS-II using MATLAB notation, it is possible to formulate systems with delays in a way that GPOPS-II will accept them. This paper will examine how well that incorporation works, explain why it has trouble when it has trouble, and also provide a discussion of some of the issues in solving OCPD. GPOPS-II can be interfaced with different NLP solvers but comes with the interior point method IPOPT [29] so we will use IPOPT.

Delayed dynamical systems can come in many forms. Delays may be varying in length and there can be multiple delays. The dynamics can be nonlinear. It is also possible to have advances and delays involving integrals. There is nothing intrinsically linear in the ideas we discuss here but in this paper it suffices to primarily consider linear examples of the form:

\begin{align}
\dot{x}(t) &= Ax(t) + C x(t - \tau) + Bu(t) + D u(t - s), \quad 0 \leq t \leq T \\
x(t) &= \phi(t), \quad -\tau < t < 0 \\
u(t) &= \psi(t), \quad -s < t < 0
\end{align}

(1a) (1b) (1c)

to illustrate our points although two nonlinear problems are given later. Here \(\tau, s > 0\) are constant delays with \(\tau, s < T\) and \(A, B, C, D\) constant matrices. \(x(t)\) is \(n\) dimensional. \(u\) are piecewise smooth functions. \(u\) will later be the control. \(\phi, \psi\) are referred to as the prehistory or initial condition. Delays can occur in both the state and the control, however, in this study we shall consider problems only with state delays or only with control delays in order to more clearly see the different numerical effects.

When we consider examples of optimal control problems in later sections, we will look at costs of the form

\[ J(u) = \int_0^T (x(t) - r(t))^T Q(x(t) - r(t)) + u(t)^T Ru(t)dt. \]

(2)

Here \(R > 0\) and \(Q \geq 0\). \(r\) in a given problem will be known, thus the problem can be considered a tracking problem. Having \(r\) is useful in producing examples where the optimal solutions have state solutions that do not tend to zero. Of course, more
general costs and dynamics are important but this class of problems suffices for illustrating the points that we wish to make in this paper.

We note that the necessary conditions for a general problem of the form
\[
\min J(u, x) = g(x(T)) + \int_0^T f_0(t, x(t), x(t-\tau), u(t), u(t-s)) dt \quad (3a)
\]
\[
\frac{dx}{dt} = f(t, x(t), x(t-\tau), u(t), u(t-s)), \text{ a.e. } t \in [0, T] \quad (3b)
\]
\[
x(t) = \phi(t), t \in [-\tau, 0] \quad (3c)
\]
\[
u(t) = \psi(t), t \in [-s, 0) \quad (3d)
\]
\[
w(x(T)) = 0, \quad (3e)
\]
where a.e. means almost everywhere in case of discontinuous controls can be expressed as follows. Let
\[
H(t, x, y, u, v, \lambda, \mu) = \lambda f_0(t, x, y, u, v) + \lambda f(t, x, y, u, v).
\]
Then the necessary conditions are
\[
\frac{dx}{dt} = f(t, x(t), x(t-\tau), u(t), u(t-s)), \text{ a.e. } t \in [0, T] \quad (4a)
\]
\[
\dot{\lambda} = -H_x(t) - \chi_{[0, T-\tau]}(t) H_y(t+\tau) \quad (4b)
\]
\[
\lambda(T) = g_x(x(T)) + \nu w_x(x(T)) \quad (4c)
\]
\[
0 = H_u(t) + \chi_{[0, T-s]}(t) H_v(t+s) \quad (4d)
\]
plus any initial conditions. (4) defines a boundary value problem with delays (DB-VP). The control \(u\) is not bounded here. An example with bounded control is given later. We will focus on state delays first.

2. Solving a state delay equation. Before turning to an OCP, we first need to consider how GPOPS-II does as an integrator of systems with delays with no optimization involved. Used this way GPOPS-II becomes a DBVP solver.

There are a number of methods of integrating delayed systems. MATLAB has dde23 [28]. SOSD can solve the delayed problem by not including a cost function and unknown control. Using the method of steps (MOS), a delayed system may be written as an undelayed boundary value problem (BVP), see (10). This can be solved with a BVP solver such as bvp4c in MATLAB, SOS, or GPOPS-II. However, there are few publicly available delay DBVP solvers.

Another approach that can be used with delayed problems is to rewrite them as an iteration where on each step an ODE is solved by an ODE or BVP solver. For example, equation (1a) can be written as
\[
\dot{x}_n(t) = Ax_n(t) + Bu(t) + Cx_{n-1}(t-\tau). \quad (5)
\]
This will be referred to as waveform integration as opposed to waveform optimization which is looked at later [1, 2, 24]. Since (5), for a given \(x_{n-1}\) is an ODE in \(x_n\) at each iteration, this calculation can be done by GPOPS-II on each iteration.

In GPOPS-II the problem formulation gives the user direct access to the entire state and control history and not just at one time step as with most BVP integrators and many OCP software interfaces. This makes it possible, with some care, to incorporate delays in the “continuous” file which describes the dynamics.

For the kind of problems considered here, when integrating a linear state delayed system, SOSD, SOS and GPOPS-II using MOS, or dde23 all quickly produced comparable answers.

In order to give GPOPS-II a state delayed problem like \(x' = ax + c x(t-\tau) + bu(t)\) with a prehistory identically 1, we proceed as follows. When running GPOPS-II
there are three files: main, continuous, and events. Main is a driver. Continuous is a function where the dynamics and cost equations are described. Events is a function for boundary conditions, cost, and linking phases. Assuming that \(a, b, c, \tau\) were defined in the main file, we would place the following code within continuous to describe the dynamics. auxdata is used to pass parameters between the files and functions.

```matlab
tau=input.auxdata.tau; % the delay
a=input.auxdata.a;
b=input.auxdata.b;
c=input.auxdata.c;
t=input.phase.time(:,1); % time grid of all collocation points
x=input.phase.state(:,1); % full state history on t
u=input.phase.control(:,1); % full control history on t
iter=size(t);
NN=iter(1,1);
if NN < 4
    dx = a*x +u; % dx with no delay
else
    ts=t-tau;
    tsn=find(ts<=0);
    tsp=find(ts >0);
    xd1=ones(size(tsn)); % prehistory is 1 for this example
    xd2 = interp1(t,x,ts(tsp),'linear'); % could use pchip or spline for interp1
    xd=[xd1;xd2];
    dx = a*x + c*xd + b*u;
end
phaseout.dynamics = dx;
```

Given the problem description there is an initial setup GPOPS-II does using this information before doing the mesh refinement and solution on the first grid. GPOPS-II often has trouble with the delay term during the compilation of this setup. The 3 lines beginning with “if \(NN < 4\)” trick GPOPS-II to first utilize the data without the delay and only use the delay once the grid has a few points in it. We found that taking \(NN < 4\) worked well but one could use a different value like 5 or 6. To help distinguish the different ways GPOPS-II is used we shall use GPOPS-IIIm to mean the delayed system is implemented as above for either an integration, or a BVP, or as attempted solution of an OCPD.

In what follows we will only present a small sampling of the extensive experimentation we have done. In this section we compare the ability of GPOPS-IIIm to solve a basic state delay differential equation (DDE) to that of the MATLAB software package dde23. We treat the solution from dde23 as the “true” solution to the DDE. We have chosen four examples; one asymptotically stable (6a), one less asymptotically stable (6b), one unstable (6c), and one less unstable (6d). \(\lambda\) is the eigenvalue with largest real part for the given delay equation. All other eigenvalues have a negative real part.

\[
x' = -2x + x(t - 2), \quad \lambda = -0.2731 \quad (6a)
\]
\[
x' = -3x + 2.8x(t - .3), \quad \lambda = -0.1084 \quad (6b)
\]
\[
x' = -3x + 3.4x(t - .3), \quad \lambda = 0.2011 \quad (6c)
\]
\[
x' = -3x + 3.2x(t - .3), \quad \lambda = 0.1029. \quad (6d)
\]
To find the roots of our delay differential equations we used the root finding package developed for MATLAB [32, 33]. The software gives both an approximate and corrected eigenvalue. We use approximate eigenvalues. In all cases, we set GPOPS-IIm to a maximum of three iterations. Additional iterations did not significantly change the results. We initially set all our tolerances in both packages to $10^{-7}$. We initially solved the equations on a shorter interval, $[0, 4]$ and later repeated our comparison on longer intervals.

The solutions of GPOPS-IIm and dde23 agree very well in all cases for the short interval $[0, 4]$. For stable systems they differed on the order $10^{-5}$. For unstable systems the difference was slowly growing and order of $10^{-4}$.

We also considered these problems on a longer interval. We began with $[0, T] = [0, 20]$ and gradually increased the length of the interval to observe the change in behavior. The two software solutions agreed very well for the stable system until we reached a final time of $T = 23.95$. Attempting to take a longer time interval was too computationally expensive for GPOPS-IIm to find a solution. We will comment on that later. GPOPS-IIm even did well with the approximation of the slightly unstable example in (6d). However, GPOPS-IIm was unable to successfully approximate the very unstable example (6c) on $[0, 20]$. On the stable solutions we again got an error of $10^{-5}$. When we tightened the tolerances in GPOPS-IIm to $10^{-8}$, we were able to get an approximation with absolute error of $10^{-3}$ for the very unstable system on $[0, 20]$.

2.1. Waveform iteration on a state delayed system. If a delay integrator is not available, then as noted one option is waveform relaxation. There is extensive literature on using waveform methods as integrators for ODEs and PDEs. Since GPOPS-II is a high quality solver of ODEs as expected this often works very well. For example, consider

$$\dot{x} = ax + \sigma x(t - \tau), \quad (7)$$

where $a$ and $\sigma$ are constants and $\tau$ is the delay. We solved this problem using waveform relaxation, calling GPOPS-II recursively. This will be referred to as GPOPS-IIw. We then compared our results to MATLAB’s built in dde23 solver, as well as to waveform relaxation using the MATLAB ode solver ode45. Using ode45 in a waveform manner will be called ode45w.

**Example 1.** Take $a = -0.5, \sigma = -1.2$, and $\tau = 1.0$ in (7) to obtain the problem

$$\dot{x} = -0.5x - 1.2x(t - 1.0). \quad (8)$$

For both GPOPS-IIw and ode45w we took five iterations. A solution with $\tau = 0$ was the starting value $x_0(t)$. We allowed GPOPS-IIw one mesh iteration on the first function call and four mesh iterations during subsequent function calls. We set our tolerances to $10^{-6}$ for the first iteration and $10^{-3}$ for the remaining four. Further iterations did not improve the size of the residual and were therefore omitted. While GPOPS-IIw and ode45w were able to solve the problem to the same order of accuracy, $10^{-4}$, ode45w was slightly more accurate. Figure 1 demonstrates that GPOPS-IIw converged almost immediately, while ode45w took a full three iterations before it was able to converge. Residuals for (8) from GPOPS-IIw and dde23 and residuals for (8) from ode45w and dde23 with $\sigma = -1.2$ and $\tau = 1.0$. were of size $10^{-4}$.

We see then that GPOPS-IIm is able to solve some delay problems although with longer intervals and larger delays it took considerably longer than the other options.
examined here. It was comparable when used as a BVP solver on method of steps. We also see that GPOPS-IIw solved state delay problems quickly when used as a waveform integrator. We now look at OCPD.

3. Solving a state delayed optimal control problem. The theory for delay optimal control has been studied by several authors, we note only [9, 12, 13, 14]. It was not always convenient to use SOSD for comparison purposes and to verify which solutions were correct. We wrote a basic control parameterization code for solving state delayed OCPD. The control is parameterized on a uniform grid of $N$ points with the cubic splines function of MATLAB, the integration is done by dde23, and the optimization of the cost over the parameterization is done by fmincon.

Having seen that we are getting acceptable answers for the dynamics using GPOPS-IIIm we now turn to see how the new idea of GPOPS-IIIm works on solving optimal control problems. The first example is a tracking problem with state delays.

Example 2. Minimize

$$\int_0^5 10(x(t) - 2)^2 + u^2(t)dt,$$

subject on $0 \leq t \leq 5$ to

$$\dot{x}(t) = -0.5x(t) + \sigma x(t - \tau) + u(t)$$

$$x(t) = 1, \quad -1 \leq t < 0$$

$$x(0) = 1.$$  

For $\sigma = 1.2, \tau = 1$, this delay equation has one positive eigenvalue of 0.3476 and $Re(\lambda) \leq -1.3451$ for the other eigenvalues so the system is not stable without the control. It is stable with $\sigma = -1.2, \tau = 1$.

With a requested mesh error of $10^{-7}$, GPOPS-IIIm produces the reasonable appearing control and state solutions in Figure 2. The computed cost was 44.6641. Linear interpolation produced essentially the same cost.
To test the accuracy of this answer we solved the problem again with GPOPS-II but with a Method of Steps (MOS) formulation. The MOS formulation of (2) is

\[
\min_u \sum_{i=1}^5 \int_0^1 10(x_i(t) - 2)^2 + u_i^2(t)\,dt
\] \hspace{1cm} (10a)

Subject to the following dynamics on $0 \leq t \leq 1$.

\[
\dot{x}_1(t) = -0.5x_1(t) + 1.2 + u_1(t), \hspace{1cm} (10b)
\]

\[
\dot{x}_i(t) = -0.5x_i(t) + 1.2x_{i-1}(t) + u_i(t), \quad i = 2, 3, 4, 5 \hspace{1cm} (10c)
\]

\[
x_1(0) = 1 \hspace{1cm} (10d)
\]

\[
x_i(0) = x_{i-1}(1), \quad i = 2, 3, 4, 5. \hspace{1cm} (10e)
\]

When we run GPOPS-II on the MOS problem we get a cost of 44.4214 and the results in Figure 3. The costs differed by 2.6%. This same problem was solved using SOSD with the bisection method for mesh refinement and also with our control parametrization code. These computations all resulted in Figure 3 and essentially the same optimal cost. The SOSD solution was visually identical to the control parameterization solution so it is not labeled in the Figure. This validates the MOS and control parameterization results and shows that the result of Figure 2 is close but incorrect and has a higher cost. Since GPOPS-IIIm resolves the dynamics the
computed control will give the computed cost and trajectory. Thus the computed control is suboptimal. In particular, note in Figure 3 that there is a small corner around \( t = 4 \) in both the MOS and SOSD optimal control solutions but that corner is missing in the GPOPS-IIIm solution in Figure 2(a).

In general when solving these problems we see that GPOPS-IIIm is taking a much longer time and the solutions are suboptimal. The slowness ran across all examples as did the suboptimality. The amount of suboptimality decreased as the delay decreased.

4. Analysis of what is happening. The question is what is happening? When performing the optimization, GPOPS-II uses specialized sparse linear algebra. Needed Jacobians and gradients are computed under the assumption that state variables appear in specific places. Thus it appears possible that what is happening is that the delayed terms appear in function evaluations but are missing from the Jacobians and this becomes more important as the delay increases with optimization as opposed to simulation. It is possible to supply gradients to GPOPS-II but they are assumed to have the same structure as the difference gradients and thus this is not helpful in improving the solution of OCPD.

To test this idea about gradients, we consider what we call waveform optimization since the delayed terms are treated as known on each iteration and will not affect Jacobians in the optimization. The GPOPS-IIIm gradient should be similar to the GPOPS-II gradient when GPOPS-II is doing waveform optimization.

4.1. Waveform optimization on a state delayed OCP. As noted GPOPS-II was able to successfully solve the state delayed dynamics equations using waveform integration. This raises the question of whether it can solve a OCPD using what we call waveform optimization. That is, it iteratively solves an OCP without delays. Consider the problem

\[
\begin{align*}
\min_u J &= \int_0^T 10(x - 2)^2 + u^2 \, dt, \quad (11a) \\
\dot{x} &= ax + \sigma x(t - \tau) + u, \quad x(0) = 1, \quad (11b)
\end{align*}
\]

where \( a \) and \( \sigma \) are constants and \( \tau \) is the value of the delay. Waveform optimization would recursively solve the OCP

\[
\begin{align*}
\min_u J &= \int_0^T 10(x_n - 2)^2 + u^2 \, dt, \quad (12a) \\
\dot{x}_n &= ax_n + \sigma x_{n-1}(t - \tau) + u, \quad x_n(0) = 1. \quad (12b)
\end{align*}
\]

When GPOPS-II is used this way we will refer to it as GPOPS-IIow. We ran GPOPS-IIow for five iterations. For our first iteration we set our tolerances at \( 10^{-6} \) and allowed one mesh iteration. For the four subsequent iterations we set our tolerances at \( 10^{-9} \) and allowed four mesh iterations. Our purpose in setting a lower tolerance and number of mesh iterations in the first run was to decrease the computational time. GPOPS-IIow ran much faster than GPOPS-IIIm since it is no longer solving the delay problem itself.

We begin with (9) with \( \sigma = -1.2, \tau = 1.0 \). We can see from Figures 4 and 5 that GPOPS-IIow converges after five iterations. The new state in Figure 4(a) is a much better approximation than the old solution in Figure 4(b). Unfortunately it does not converge to the completely correct OCPD solution obtained by SOSD and
control parameterization. To see why this occurs we need to look at using necessary conditions (4).

![Figure 4.](image1.png)

**Figure 4.** Left: Iterative states of GPOPS-IIow for (9) and Right: states obtained by SOSD, GPOPS-IIim, and control parameterization with $\sigma = -1.2$ and $\tau = 1.0$.

![Figure 5.](image2.png)

**Figure 5.** Left: Iterative controls of GPOPS-IIow for (9) and Right: controls obtained by SOSD, GPOPS-IIim, and control parameterization with $\sigma = -1.2$ and $\tau = 1.0$.

4.2. **Basic theory of state delayed OCP.** Suppose that we have

\[
\begin{aligned}
\min_u J &= \int_0^T (x - r)^T Q (x - r) + u^T R u \, dt. \\
\dot{x} &= Ax + Cx(t - \tau) + Bu(t).
\end{aligned}
\]  

(13a)  

(13b)

Then from (4) the necessary conditions for the state delayed OCPD are

\[
\begin{aligned}
\dot{x}(t) &= Ax(t) + Cx(t - \tau) + Bu, \\
\dot{\lambda}(t) &= -(2Q(x(t) - r(t)) + A^T \lambda(t)) - \chi_{[0,T-\tau]}(t)C^T \lambda(t + \tau) \\
\lambda(T) &= 0 \\
0 &= 2Ru(t) + B^T \lambda(t) \\
x(t) &= \phi(t), \ t \in [-\tau, 0],
\end{aligned}
\]  

(14a)  

(14b)  

(14c)  

(14d)  

(14e)
which results in the DBVP necessary conditions

\[
\begin{align*}
\dot{x}(t) &= Ax(t) + Cx(t - \tau) - \frac{1}{2}BR^{-1}BT\lambda, \quad (15a) \\
\dot{\lambda}(t) &= -(2Q(x(t) - r(t)) + A^T\lambda(t)) - \chi_{[0,T-\tau]}(t)C^T\lambda(t + \tau) \quad (15b) \\
\lambda(T) &= 0 \quad (15c) \\
x(t) &= \phi(t), \ t \in [-\tau, 0], \quad (15d)
\end{align*}
\]

where \( \chi_I \) is the usual characteristic function of a set \( I \) so \( \chi_I \) is 1 if \( t \in I \) and 0 otherwise.

If we do GPOPS-IIow to convergence, then the necessary conditions for the final optimization problem treat \( x(t - \tau) \) as an exogenous function in a problem without delays so the necessary conditions result in the BVP necessary conditions

\[
\begin{align*}
\dot{x}(t) &= Ax(t) + Cx(t - \tau) - \frac{1}{2}BR^{-1}BT\lambda, \quad (16a) \\
\dot{\lambda}(t) &= -(2Q(x(t) - r(t)) + A^T\lambda(t)) \quad (16b) \\
\lambda(T) &= 0 \quad (16c) \\
x(t) &= \phi(t), \ t \in [-\tau, 0], \quad (16d)
\end{align*}
\]

which are not the same as (15).

It is worthwhile noting that in (15b) the \( \dot{\lambda}(t) \) equation is an \( O(\tau) \) perturbation of the \( \tau = 0 \) necessary conditions if the adjoint variables are piecewise smooth since

\[
\begin{align*}
\chi_{[0,T-\tau]}(t)C^T\lambda(t + \tau) &= \chi_{[0,T-\tau]}(t)C^T\lambda(t) + \|\lambda'\|_\infty \|C\|O(\tau) \quad (17) \\
&= \chi_{[0,T-\tau]}(t)C^T\lambda(t) + (\|\lambda'\|_\infty + \|\lambda\|_\infty) \|C\|O(\tau) \quad (18) \\
&= C^T\lambda(t) + O(\tau). \quad (19)
\end{align*}
\]

Thus for small \( \tau \) it is relatively easy to get a good approximation of the true solution with GPOPS-IIIm and this is born out by all of our computations using GPOPS-IIIm. Note that the perturbation with GPOPS-IIow is not \( O(\tau) \) since there is no \( C \) term in (16b). This is also born out in the computational examples.

5. **State delays as PDEs using MOL.** Given a delayed differential equation (DDE), it is possible to reformulate the DDE as a system of partial differential equations (PDEs). The system of PDEs can then be solved numerically using the method of lines (MOL) [16]. In this section, we use MOL to reformulate a OCPD. We shall see that this approach often allows GPOPS-II to get good solutions of an OCPD. That this works is not surprising since the OCPD has been reformulated without delays and is one GPOPS-II is expected to solve. What is interesting is that MOL approximations can sometimes give good solutions of the OCPD using very coarse \( x \) grids. Consider

\[
\begin{align*}
J &= \int_0^T (x - r)^TQ(x - r) + u^TRu \quad (20a) \\
\dot{x} &= Ax + C(x - \tau) + Bu \quad (20b)
\end{align*}
\]

with delay \( \tau \). We can reformulate (20) as a system of PDEs with respect to time and a delayed variable, \( z \). Let \( x(t - \tau) = x(t + z) = U(t, z) \) where \(-\tau \leq z \leq 0\). Then we have
with the nonlinear delayed dynamics which are a modified Van der Pol oscillator
discretize the interval \([\phi \tau]\) where \(U\) problem considered here. Thus our boundary condition becomes
Note that some applications will use shadow points instead but this suffices for the
and we have the system
\[
J_t = (U(t,0) - r)^T Q(U(t,0) - r) + u^T R u, \quad t \geq 0, \tag{21d}
\]
where \(\phi(\tau)\) is the prehistory. MOL is used to get an ODE from the PDE. We
discretize the interval \([-\tau 0]\) with stepsize \(h\). Let \(U(t,\tau_k) \approx U_k(t), \quad k = 0...N\). For \(U\tau\) we will use centered differences and on the boundary we use
\[
U'(t) = 1/2h (-U(t + 2h) + 4U(t + h) - 3U(t)) + \tilde{O}(h^3). \tag{22}
\]
Note that some applications will use shadow points instead but this suffices for the problem considered here. Thus our boundary condition becomes
\[
U'_0(t) = 1/2h (-U_2(t) + 4U_1(t) - 3U_0(t)) + \tilde{O}(h^3) \tag{23}
\]
and we have the system
\[
\begin{align*}
U'_0(t) & = 1/2h (-U_2(t) + 4U_1(t) - 3U_0(t)) + \tilde{O}(h^3), \quad k = 0 \tag{24a} \\
U'_k & = 1/2h (U_{k+1}(t) - U_{k-1}(t)) + \mathcal{O}(h^3), \quad k = 1...N \tag{24b} \\
U'_N & = AU_N(t) + Cu(t), \quad k = N \tag{24c} \\
J'_N & = (U_N - r)^T Q(U_N - r) + u^T R u, \quad k = N. \tag{24d}
\end{align*}
\]
We used the MOL approach given above to solve (25) with \(N = 5\) using GPOPS-II and plotted our results in Figure 6.
\[
y' = -5y - 1.2y(t - 1) + u \tag{25a} \\
J = \int_0^T 10(y - 2)^2 + u^2. \tag{25b}
\]
For our purposes, we took the solution obtained by control parameterization as “truth”. We set our tolerances at \(10^{-7}\) and allowed three mesh iterations. For this problem, GPOPS-II obtained a cost of \(J = 43.4161601\). For reference, control parameterization obtained a cost of \(J = 43.4214094\), while using GPOPS-II without
MOL obtained a cost of \(J = 44.6641508\). Using MOL to eliminate the delay from the dynamics enabled GPOPS-II to compute the cost much more accurately.

It should be noted that we are using interior point methods to solve the NLP
and the solutions can sometimes be sensitive to initial conditions [10]. For the
problems considered in this paper this sensitivity does not affect the general form
of the solutions but, given the tolerances used, can affect the third or fourth digit
in the costs when using the delay formulation of a problem and GPOPS-IIIm. It is
interesting to note that a fine \(\tau\) grid was not needed to get an accurate answer to
this problem.

5.1. Control bounds. Our approach can also be used on some nonlinear problems
with control bounds. We consider Example 5 from [23] which has control bounds and initial and terminal conditions. The cost is
\[
J = \frac{1}{2} \int_0^5 x_1^2 + x_2 + u^2 \, dt \tag{26a}
\]
with the nonlinear delayed dynamics which are a modified Van der Pol oscillator.
Figure 6. State (left) and control (right) obtained for (25) using GPOPS-II and MOL.

\[
\begin{align*}
\dot{x}_1(t) &= x_2(t) & (26b) \\
\dot{x}_2(t) &= -x_1(t) + (1 - x_1(t))^2 x_2(t - 1) + u(t) & (26c) \\
|u(t)| &\leq 0.75 & (26d) \\
x_1(t) &= 1 - 1 \leq t \leq 0 & (26e) \\
x_2(t) &= 0 - 1 \leq t \leq 0 & (26f) \\
x_1(5) &= -1 & (26g) \\
x_2(5) &= 0 & (26h)
\end{align*}
\]

When we solved this problem in GPOPS-IIIm we got a cost of 2.0237941 in comparison to 1.987116 in [23]. Our computed control and state are in Figure 7 which are quite similar to that reported in Figure 10 of [23] as shown in Figure 8.

Figure 7. State (left) and control (right) for (31) solving (26) using GPOPS-IIIm.
features: iterative-adaptive pseudospectral methods developed in this paper. We mention the following

8. Discussion of results

The computational results of Section 7 demonstrate several key features of the adaptive and... 5. Points are discrete approximations.

M. Maleki, I. Hashim / Journal of the Franklin Institute 351 (2014) 811–839

6. Control delays. We also consider GPOPS-II on OCPD with control delays such as

\[ x'(t) = ax(t) + bu(t) + du(t - \tau), \]  
\[ x(0) = 10, \]  
\[ u(t) = \phi(t), \quad -\tau \leq t < 0 \]

with cost

\[ \int_0^T 10(x - 4)^2 + u^2 dt, \]  

using the same type of coding.

The first example of (27) we take is

\[ T = 4, a = -1.14, b = 2.0, d = 0.3, x(0) = 10, \phi = -2.3. \]  

We had an initial grid of \( N = 30 \) intervals. The optimal state and control are graphed in Figure 9. Retraining the original problem (28) in SOSD gave Figure 10. Note that while GPOPS-IIIm gave a smaller solution, the difference is within the tolerances being used.

When initially working with SOSD [7], there was an issue with OCPD in the control. But in that case SOSD could easily solve OCPD with state delays. So
a reformulation was developed called EIC that moved the delay to the state with a small penalty and enabled a better answer to be found. It is interesting that a similar idea is helpful here even though the motivation is different. For example, we could change the previous problem by adding another dynamical equation $u' = z$ and then adding $\alpha \|z\|^2$ to the cost for a small $\alpha$ so that the cost becomes

$$J = \int_0^5 10\|x - 4\|^2 + \|u\|^2 + \alpha\|z\|^2 dt.$$ (29)

For example, if we take $\alpha = 0.01$ we get Figure 11 where the solution is plotted along with the SOSD solution for the original problem. As Figure 11 shows the solutions are very close.

Then we considered

$$T = 4, a = -0.2, b = 2.0, d = -0.4, x(0) = 10, \phi = -2.3,$$ (30)

with 2 iterations and $10^{-4}$ mesh tolerance. The program ran for 20 minutes and produced a result graphed in Figure 12. The results using SOSD are graphed in Figure 13.
Figure 12. State (left) and control (right) for (30) using GPOPS-IIIm. Computed cost was 52.8417171.

Figure 13. State (left) and control (right) for (30) using SOSD. Computed cost was 56.187.

In many OCP the solution is in two parts. The second part is the optimal solution. The first part is moving into position to follow the optimal strategy. This suggests that it can sometimes be appropriate to consider the precontrol $\phi$ as part of the control strategy. We returned to (28) and (1) except that we treated $\phi$ as a constant control prehistory and another optimization parameter. We took the initial guess for $\phi$ to be -2.3 and allowed it to vary between -5 and 3 during the optimization. For (28) we found the optimal $\phi$ to be -2.209 so there was little change and the optimal solution appeared the same as Figure 3. For (14) the optimal $\phi$ was $-3.0$ and we got the solution in Figure 14 which is much improved over that of Figure 12 although there was still a small spike.

7. Solution of necessary conditions: BVP with delays. The necessary conditions for both state and control delayed problems are a DBVP. There exist very few readily available user friendly DBPV solvers. GPOPS-II, like most direct transcription codes, has a boundary value philosophy and can be applied as a BVP solver. We have seen that even though it was not designed for delayed systems GPOPS-IIIm can sometimes simulate the dynamics. Since one can formulate delays in a way that GPOPS-IIIm can accept, it is of great interest to see how it performs
on the necessary conditions. We shall see that it can often solve these DBVP even though it was not designed to do that.

7.1. State delays. Solving a previous example with state delays we see in Figure 15 that solving the necessary conditions with GPOPS-IIIm can be very successful.

For the earlier discussion of what approaches might work linear examples sufficed and simplified the presentation. But it is important to note that those approaches successful on the linear problems are also very successful on many nonlinear problems. Our first example is from [23] and has been considered by several authors. The problem is to minimize

$$\frac{1}{2} x(2)^2 + \frac{1}{2} \int_0^2 x^2(t) + u^2(t) dt$$  \hspace{1cm} (31a)$$
subject to

$$\dot{x}(t) = x(t) \sin(x(t)) + x(t-1) + u(t),$$  \hspace{1cm} (31b)$$
with initial condition $x(t) = 10$ for $-1 \leq t \leq 0$. 

Figure 14. State (left) and control (right) for (1) using GPOPS-IIIm with prehistory a control variable. Computed cost was 52.8417171.

Figure 15. State (left) and control (right) for (9) with $\sigma = -1.2, \tau = 1$, found by solving the necessary conditions using GPOPS-IIIm.
The necessary conditions from (4) are

\[
\begin{align*}
\dot{x}(t) &= x(t) \sin(x(t)) + x(t - 1) - \lambda(t) \\
\dot{\lambda}(t) &= -x(t) + (\sin(x(t)) + x(t) \cos(t))\lambda(t) - \chi_{[0,1]}\lambda(t + 1) \\
\lambda(2) &= x(2) \\
x(t) &= 10, \ t \leq 0.
\end{align*}
\] (32a)  
(32b)  
(32c)  
(32d)

**Figure 16.** State (left) and control (right) for (31) solving the necessary conditions (32) with GPOPS-IIIm.

In [23] they report an optimal cost of 162.0xx with the xx depending on discretization parameters but usually close to 162.03. We computed 161.9809 with GPOPS-IIIm and tolerance of $10^{-4}$.

With nonlinear problems there is always the issue of multiple minima and finding a correct initial guess. In GPOPS-II there is the option of just specifying the endpoints and then it uses a linear initial guess between the given endpoints. For this problem we just used a vector of 10s as the starting and ending points of the state variable. When we tried a constant 1 we quickly computed a solution that was slightly too large. This solution was robust to small perturbations of starting data. Looking at the problem suggested that the control should be negative. Taking the initial guess for the control of the form $[-2; -10]$ (Matlab notation) produced the answer in Figure 16.

7.2. Control delays. Suppose the optimal control problem is

\[
\begin{align*}
\dot{x}(t) &= f(t, x(t), u(t), u(t - s)), \ a.e. t \in [0, T] \\
x(0) &= x_0 \\
u(t) &= \psi(t), t \in [-s, 0) \\
J(u, x) &= g(x(b)) + \int_0^T f_0(t, x(t), u(s), u(t - s))dt.
\end{align*}
\] (33a)  
(33b)  
(33c)  
(33d)
From (4) the necessary conditions for a control delay are
\[
\dot{x}(t) = f(t, x(t), u(t), u(t-s)), \text{ a.e. } t \in [0, T] \\
\dot{\lambda} = -H_x(t) \\
\lambda(b) = g_x(x(T)) \\
0 = H_u(t) + \chi_{[0,T-s]}(t)H_v(t+s).
\] (34a) (34b) (34c) (34d)

For our initial test problem we get
\[
\dot{x}(t) = ax(t) + bu(t) + du(t-\tau) \\
\dot{\lambda} = -20(x(t) - 4)^2 - a\lambda(t) \\
\lambda(b) = 0, \quad x(0) = 10, \\
0 = 2u(t) + \lambda(t)b + \chi_{[0,T-s]}(t)d\lambda(t+s). 
\] (35a) (35b) (35c) (35d)

From (35d) we get that \( u(t) \) in (35a) can be replaced by
\[
u(t) = -\frac{1}{2} \left( \lambda(t)b + \chi_{[0,T-s]}(t)d\lambda(t+s) \right)
\]
and the \( u(t-1) \) in (35a) replaced by
\[
u(t-1) = \begin{cases} 
\psi(t-1) & \text{if } 0 \leq t < 1 \\
-\frac{1}{2} \left( \lambda(t-1)b + \chi_{[1,T]}d\lambda(t) \right) & \text{if } 1 < t < 4.
\end{cases}
\]

This results in a DBVP with delays in the \( \lambda \) variable. Solving with this DBVP with GPOPS-II we get a cost of 53.2708304 which agrees with the SOSD results.

The state and control for both SOSD and GPOPS-II are graphed in Figure 17 and appear identical.

![Figure 17. State (left) and control (right) for (28) solving the necessary conditions with GPOPS-II and also using SOSD on the original problem, \( \tau = 1, a = -1.14 \).](image)

The key again to implementing this delay boundary value problem is GPOPS-II is working with the full state history. To illustrate, here is the part of the continuous file that provides the dynamics. Note that we have three state variables \( x, \lambda, j \). \( j \) is to provide the cost of the solution of the necessary conditions.

```plaintext
a = input.auxdata.a;
b = input.auxdata.b;
d = input.auxdata.d;
```
r = input.auxdata.r;
tau = input.auxdata.tau; % delay
x = input.phase.state(:,1); % state x
l = input.phase.state(:,2); % lambda or costate
j = input.phase.state(:,3); % for cost
tt = input.phase.time;
iter= size(tt);
NN = iter(1,1); % initialize without delay

if NN<4 % only used to get through initialization
dx = a*x + b*(-.5*b*l) + d*(-0.5*b*l);
dl = -20*(x-r) -a*l;
dj = 10*(x-r).^2 + (-.5*b*l).^2;
else
  keeptt = tt(tt<=3);
  zerott = tt(tt>3);
  newtime = keeptt + tau;
  lnew1 = interp1(tt,l,newtime);
  lnew2 = 0*zerott;
  dell = [lnew1;lnew2]; % delayed lambda
  keep1 = l(tt>1);
  ttmtau = tt-tau;
  tsn = ttmtau(ttmtau<=0); % delay times <=
  tsp = ttmtau(ttmtau>0); % delay times > 0
  lmtau = interp1(tt,l,tsp);
  lm_size= size(lmtau); % usub is u(t-1) in dynamics
  upos = -.5*(b*lmtau + d*keep1);
  uneg = -2.3*ones(size(tsn));
  usub = [uneg;upos];
dx = a*x + b*-.5*(b*l + d*dell) + d*usub;
dl = -20*(x-r) - a*l;
dj = 10*(x-r).^2 + (-.5*(b*l + d*dell)).^2;
end

phaseout.dynamics = [dx, dl, dj];

We also solved (30) using the necessary conditions with GPOPS-IIIm. This problem seemed to be more delicate as to starting guess. We frequently got a solution that was close but had trouble near \( t = 1 \). However, if we used the incorrect solution to motivate the starting conditions we did much better. For example using,
tGuess = [0;0.8;1;1.5; 3; 4];
xGuess = [x0,17,0;4,0.1,2; 0.38,0.1,4; 4.2,0,6; 4,0,8 ; 4,0,10];
we got the states and controls in Figure 18 which should be compared to Figure 13.

We have seen that GPOPS-IIIm does a much better job of solving delayed dynamics or DBVP than it does an OCPD. The reason is related to the different role that gradients play in the two situations. When solving an optimal control problem there are parameters, such as values of the controls, that have to be varied in order
Figure 18. State (left) and control (right) for (30) solving the necessary conditions with GPOPS-IIIm.

The inner loop proceeds as follows. Given values of $x_k, \lambda_k, z_k$ which are the approximation of $x$, the Lagrange multipliers for the constraints, and the multipliers for the inequality constraints, a damped Newton’s method is applied. The search directions $d^x_k, d^\lambda_k, d^z_k$ are obtained from (38) which is (9) in [29]:

$$
\begin{bmatrix}
W_k & A_k & -I \\
A_k^T & 0 & 0 \\
Z_k & 0 & X_k
\end{bmatrix}
\begin{bmatrix}
d^x_k \\
d^\lambda_k \\
d^z_k
\end{bmatrix} =
\begin{bmatrix}
\nabla f + A_k \lambda_k - z_k \\
c(x_k) \\
X_k Z_k e - \mu_j e
\end{bmatrix}, \quad (38)
$$

where $A_k = \nabla c(x_k)$. Given the search direction, the iterates are updated by

$$
\begin{align*}
x_{k+1} &= x_k + \alpha_k d^x_k \\
\lambda_{k+1} &= \lambda_k + \alpha_k d^\lambda_k \\
z_{k+1} &= z_k + \alpha_k d^z_k.
\end{align*} \quad (39)
$$

However, when performing a simulation, or solving a DBVP, the unknown state values are completely determined by $c(x) = 0$ and (39) with GPOPS-IIIm becomes
the damped quasi-Newton method

\[ \dot{c}'(x_k)d_k^T = -c(x_k) \quad (40a) \]
\[ x_{k+1} = x_k + \alpha_k d_k^T, \quad (40b) \]

where \( \dot{c}'(x_k) \) is an approximation of \( c'(x_k) \) gotten by zeroing those entries which come from the delayed parts of the equations. Note that because of the structure of the discretized dynamics equations \( \dot{c}'(x_k) \) and \( c'(x_k) \) are both nonsingular for \( x_k \) close to the solution of \( c(x) = 0 \).

There is a large body of literature exploiting the fact that damped quasi-Newton methods can also sometimes converge using simplified approximations of Jacobians. When effective the cost is typically restricted to have to take more iterations and sometimes to lose a little accuracy.

Looking at (39) and (40) we can establish some conditions that guarantee the damped quasi-Newton method will converge for good enough initial guesses. To simplify the notation we will suppress the \( k \) subscript and look at the change in \( \|c(x)\|^2 \) in one iteration using a search direction \( \hat{\delta} \) gotten from (40) and the fact that \( c(x + \hat{\delta}) = c(x) + c'(x)\hat{\delta} + O(\|\hat{\delta}\|^2) \). We have

\[
(c(x + \hat{\delta})^Tc(x + \hat{\delta}) - c(x)^Tc(x)) = (c(x) + c'(x)\hat{\delta} + O(\|\hat{\delta}\|^2))^T(c(x) + c'(x)\hat{\delta} + O(\|\hat{\delta}\|^2)) - c(x)^Tc(x) \quad (41a)
\]

\[ = \hat{\delta}^Tc'(x)^Tc(x) + c(x)^Tc'(x)\hat{\delta} + O(\|\hat{\delta}\|^2). \quad (41b) \]

Dropping the \( O \) term, and substituting back in \( \hat{\delta} = -\dot{c}'(x)^{-1}c(x) \), we have

\[ -(\dot{c}'(x)^{-1}c(x))^Tc'(x)^Tc(x) - c(x)^Tc'(x)\dot{c}'(x)^{-1}c(x) \quad (42a) \]
\[ = -c(x)^T(\dot{c}'(x)^{-T}c'(x)^T + c'(x)\dot{c}'(x)^{-1})c(x). \quad (42b) \]

Now \( c'(x) \) can be written as \( E + F \) and \( \dot{c}'(x) = E \) where the nonzero entries of \( F \) appear in places where \( E = \dot{c}'(x) \) has zero entries. Thus (42b) is

\[ -c^T(E^{-T}(E + F)^T + (E + F)E^{-1})c \quad (43a) \]
\[ = -c^T((I + FE^{-1})^T + (I + FE^{-1}))c. \quad (43b) \]

We want (43b) to be negative along the search direction \( \hat{\delta} \) given we know it is negative along the direction \( \hat{\delta} \). A sufficient condition is that the numerical radius of \( FE^{-1} \) is less than one which is a weaker condition than \( FE^{-1} \) has norm less than one. This is not a necessary condition and the iterations will sometimes converge when it does not hold. Note that this sufficient condition is not a small delay condition like we mentioned earlier. The delay can be large. It is a statement about the magnitude of the delay terms.

8. Conclusion & Comments. The numerical solution of OCPD which include state or control delays using the numerical software GPOPS-II which runs under MATLAB is examined. A modification incorporating delays, called GPOPS-IIim is introduced. It is seen that for small delays it can provide a good approximation of the optimal control. For larger state delays the execution time slows down and the computed control is suboptimal. Similar discretizations to those of GPOPS-II have been used by others, such a Biegler’s group at Carnegie Mellon with success on OCPD, so the problem is most likely that the sparse linear algebra in GPOPS-II is producing inaccurate Jacobians and gradients.
An alternative is introduced which we call GPOPS-IIow. This method converges much faster to an answer and computes a suboptimal control. However, the amount of suboptimality does not go to zero as the delay goes to zero so the usefulness of this approach will be highly problem dependent.

GPOPS-II, like most direct transcription software, has a boundary value philosophy behind it. Thus the approach we used to describe delay problems in GPOPS-IIIm could also be used with advances or problems with delays and advances to produce suboptimal solutions. A boundary value solver does not have the same dependence on direction of information flow that an initial value solver does.

In our examples we treated the initial condition \( \phi \) as known. But GPOPS-II allows for including model parameters in the optimization. Thus if the initial condition is not known, it can be parameterized and those parameters added to the optimization problem in GPOPS-IIIm. Again suboptimal solutions would be expected.

Stronger results came with using the necessary conditions. There are very few pieces of user friendly software for solving DBVP. However, the way GPOPS-II accepts formulations allows us to formulate the necessary conditions as a DBVP that GPOPS-II accepts. GPOPS-IIIm turned out to be successful in solving a number of these problems as we have illustrated. This success has been analyzed and explained.

REFERENCES

[1] Z. Bartoszewski and M. Kwapisz, On the convergence of waveform relaxation methods for differential-functional systems of equations, *J. Math. Anal. Appl.*, 225 (1999), 478–496.

[2] Z. Bartoszewski and M. Kwapisz, On error estimates for waveform relaxation methods for delay-differential equations, *SIAM J. Numerical Analysis*, 38 (2011), 639–659.

[3] J. T. Betts, *Methods for Optimal Control and Estimation using Nonlinear Programming*, SIAM, Philadelphia, 2010.

[4] J. T. Betts, N. Biehn, S. L. Campbell and W. Huffman, Compensating for order variation in mesh refinement for direct transcription methods II: computational experience, *J. Comp. Appl. Math.*, 143 (2002), 237–261.

[5] J. T. Betts, S. L. Campbell and K. Thompson, Optimal control of delay partial differential equations, in *Control and Optimization with Differential-Algebraic Constraints*, SIAM, (2012), 213–231.

[6] J. T. Betts, S. L. Campbell and K. Thompson, Optimal control software for constrained nonlinear systems with delays, *Proc. IEEE Multi-Conference on Systems and Control (2011 MSC)*, Denver, (2011), 444–449.

[7] J. T. Betts, S. L. Campbell and K. Thompson, Solving optimal control problems with control delays using direct transcription, *Applied Numerical Mathematics*, 108 (2016), 185–203.

[8] N. Biehn, J. T. Betts, S. L. Campbell and W. Huffman, Compensating for order variation in mesh refinement for direct transcription methods, *J. Comp. Appl. Math.*, 125 (2000), 147–158.

[9] G. V. Bokov, Pontryagin’s maximum principle of optimal control problems with time-delay, *J. Mathematical Sciences*, 172 (2011), 623–634. (Russian version: Fundam. Prikl. Mat., 15 (2009), Issue 5, 3–19.)

[10] S. L. Campbell, J. T. Betts and C. Digirolamo, Comments on initial guess sensitivity when solving optimal control problems using interior point methods, *Numerical Algebra, Control, and Optimization*, 10 (2020), 39–41.

[11] C. L. Darby, W. W. Hager and A. V. Rao, An hp-adaptive pseudospectral method for solving optimal control problems, *Optimal Control Applications and Methods*, 32 (2011), 476–502.

[12] J. F. Frankena, Optimal control problems with delay, the maximum principle and necessary conditions, *J. Engineering Mathematics*, 9 (1975), 53–64.

[13] L. Göllmann, D. Kern and H. Maurer, Optimal control problems with delays in state and control subject to mixed state control-state constraints, *Optimal Control Applications and Methods*, 30 (2009), 341–365.
[14] L. Göllmann and H. Maurer, Theory and application of optimal control problems with multiple delays, *J. Industrial and Management Optimization*, 10 (2014), 413–441.

[15] Z. H. Gong, C. Y. Liu and Y. J. Wang, Optimal control of switched systems with multiple time-delays and a cost on changing control, *Journal of Industrial and Management Optimization*, 14 (2018), 183–198.

[16] T. Koto, Method of lines approximation of delay differential equations, *Computers & Mathematics with Applications*, 48 (2004), 45–59.

[17] C. Y. Liu, R. Loxton, Q. Lin and K. L. Teo, Dynamic optimization for switched time-delay systems with state-dependent switching conditions, *SIAM Journal on Control and Optimization*, 56 (2018), 3499–3523.

[18] C. Y. Liu, Z. Gong, H. W. Lee and K. L. Teo, Robust bi-objective optimal control of 1,3-propanediol microbial batch production process, *Journal of Process Control*, 78 (2019), 170–182.

[19] C. Liu, Z. Gong, K. L. Teo, R. Loxton and E. Feng, Bi-objective dynamic optimization of a nonlinear time-delay system in microbial batch process, *Optimization Letters*, 12 (2018), 1249–1264.

[20] C. Y. Liu, R. Loxton, and K. L. Teo, A computational method for solving time-delay optimal control problems with free terminal time, *Systems & Control Letters*, 72 (2014), 53–60.

[21] C. Y. Liu, R. Loxton and K. L. Teo, Optimal parameter selection for nonlinear multistage systems with time-delays, *Computational Optimization and Applications*, 59 (2014), 285–306.

[22] C. Y. Liu, R. Loxton and K. L. Teo, Switching time and parameter optimization in nonlinear switched systems with multiple time-delays, *Journal of Optimization Theory and Applications*, 63 (2014), 957–988.

[23] M. Maleki and I Hashim, Adaptive pseudospectral methods for solving constrained linear and nonlinear time-delay optimal control problems, *J. Franklin Institute*, 351 (2014), 811–839.

[24] J. Mead and B. Zubik-Kowal, An iterated pseudospectral method for delay partial differential equations, *Applied Numerical Mathematics*, 55 (2005), 227–250.

[25] M. A. Patterson and A. V. Rao, GPOPS II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming, *ACM Trans. Math. Software*, 41 (2014), 1–37.

[26] H. Peng, X. Wang, S. Zhang and B. Chen, An iterative symplectic pseudospectral method to solve nonlinear state-delayed optimal control problems, *Commun. Nonlinear. Sci. Numer. Simulat.*, 48 (2017), 95–114.

[27] A. V. Rao, D. A. Benson, C. Darby, M. A. Patterson, C. Francolin, I. Sanders and G. T. Huntington, Algorithm 902: Gpops, a MATLAB software for solving multiple-phase optimal control problems using the Gauss pseudospectral method, *ACM Transactions Mathematical Software*, 37 (2010), 1–39.

[28] L. F. Shampine and S. Thompson, Solving DDEs in Matlab, *Applied Numerical Mathematics*, 37 (2001), 441–458.

[29] A. Wächter and L.T. Biegler, On the implementation of interior-point filter line-search algorithm for large-scale nonlinear programming, *Math. Prog.*, 106 (2006), 25–57.

[30] Z. Wang and L. Wang, A Legendre-Gauss collocation method for nonlinear delay differential equations, *Discrete and Continuous Dynamical Systems, Series B*, 13 (2010), 685–708.

[31] D. Wu, Y. Bai and C. Yu, A new computational approach for optimal control problems with multiple time-delay, *Automatica*, 101 (2019), 388–395.

[32] Z. Wu and W. Michiels, Reliably computing all characteristic roots of delay equations in a given right half plane using a spectral method, *J. Comp. and Appl. Math.*, 236 (2012), 2499–2514.

[33] Z. Wu and W. Michiels, Reliably computing all characteristic roots of delay differential equations in a given right half plane using a spectral method, Internal report TW 596, Department of Computer Science, K. U. Leuven, May, 2011. Available by download from http://twr.cs.kuleuven.be/research/software/delay-control/roots/+.

Received April 2019; 1st revision January 2020; Final revision February 2020.

E-mail address: john.t.betts@comcast.net
E-mail address: slc@ncsu.edu
E-mail address: mdigiro@ncsu.edu