A Self-Stabilizing Pantoja-like Indirect Algorithm for Optimal Control

Bruce Christianson

Numerical Optimisation Centre, University of Hertfordshire
Hatfield, England, Europe

Abstract. In 1983 Pantoja described a computationally efficient stage-wise construction of the Newton direction for the discrete time optimal control problem. Automatic Differentiation can be used to implement Pantoja's algorithm, and calculate the Newton direction without truncation error, and without extensive manual re-writing of target function code to form derivatives.
Pantoja's algorithm is *direct*, in that the independent variables are the control vectors at each timestep. In this paper we formulate an *indirect* analogue of Pantoja's algorithm, in which the only independent variables are the components of a costate vector corresponding to the initial timestep. This reformulated algorithm gives exactly the Newton step for the initial costate with respect to a terminal transversality condition: at each timestep we solve implicit equations for the current controls and successor costates. A remarkable feature of the indirect algorithm is that it is straightforward to compensate for the effect of non-zero residuals in the implicit costate equations. The indirect reformulation of Pantoja's algorithm set out in this paper is a suitable basis for verified optimization using interval methods.

1 Introduction

Consider a discrete-time optimal control problem in the following *direct* form: choose $u_i \in R^p$ for $0 \leq i < N$ so as to minimize

$$z = F(x_N)$$

where x_0 is some fixed constant and the state equation is

$$x_{i+1} = f_i(x_i, u_i) \quad \text{for } 0 \leq i < N.$$

Here each f_i is a smooth map from $R^q \times R^p \to R^q$ and F is a smooth map from R^q to R. The dimension of u_i may depend upon the timestep i, but for notational convenience we omit this refinement.

We lose nothing by restricting attention to target functions of this form: the more usual formulation where z has the form $z = \sum_{i=0}^{N-1} F_i(x_i, u_i) + F_N(x_N)$, can be reduced to the form $z = F(x_N)$ by augmenting each state x_i with a new component $v_i \in R$ defined by $v_0 = 0; v_{i+1} = v_i + F_i(x_i, u_i)$ and defining $F(x_N, v_N) = v_N + F_N(x_N)$.

An obvious approach to solving an optimal control problem is to apply Newton's method. In the direct formulation the Np independent variables are the components of the control vectors $u_i : 0 \leq i < N$. In 1983, Pantoja presented a a stagewise construction of the Newton direction for the direct formulation [14, 15]. Pantoja's algorithm is a modification of the Differential Dynamic Programming (DDP) approach [13, 11] and so incurs amazingly low computational costs: order $p + q$ times the computational cost of evaluating z, independent of N, together with order $(p + q)^3$ multiply-and-add operations per time step, less if there is structural sparsity. Checkpointing [5, §5] can be used to reduce the total space requirement of Pantoja's algorithm for a typical problem to the same order as the storage required for u. Automatic Differentiation (AD) techniques [10] can be used to implement Pantoja's algorithm, allowing the Newton direction for the u_i to be calculated accurately without incurring any truncation error and without requiring extensive manual re-writing of target function code to form first or second derivative expressions explicitly [5, 6, 3].

In this paper we show that an algorithm of analogous form to Pantoja's algorithm, and which enjoys the same implementation advantages, can be derived in an *indirect* formulation, where the only independent variables are the components of a *costate* vector $\tilde{x}_0$ corresponding to the initial timestep. Furthermore, the indirect algorithm can be made self-stabilizing, in the sense that the construction compensates for the first-order effect of non-zero residuals in the implicit equations which are solved for the control values at each timestep.

This paper is organized as follows. In the next section we characterize the Newton step for both direct and indirect formulations of the discrete time optimal control problem. In Section 3 we set out Pantoja's direct algorithm to construct the Newton step, and review its properties together with the application of AD. Section 4 contains the necessary tool-kit of technical machinery for establishing properties of algorithms of this general form in either formulation. In

Section 5 we introduce a new, indirect, reformulation of Pantoja's algorithm, and show that it provides the Newton step for the initial costate. We then show how to enhance this indirect algorithm so as to compensate for the effect of non-zero residuals in the adjoint state-control equations. In the final section we discuss the significance of these results and set out a manifesto for future research.

2 Direct and Indirect Formulations

It is convenient to characterize the Newton step for the *direct* formulation of the discrete-time optimal control problem in terms of the *adjoint* equations corresponding to the state equations for the original problem. Define adjoint state and adjoint control variables, $\bar{x}_i \in R^q$ and $\bar{u}_i \in R^p$ respectively, by the adjoint state and adjoint control equations:

$$\bar{x}_N = F'(x_N); \qquad \bar{x}_i = [f'_{x,i}]^T \bar{x}_{i+1}; \qquad \bar{u}_i = [f'_{u,i}]^T \bar{x}_{i+1} \quad \text{for } 0 \leq i < N$$

where $\cdot^T$ denotes transpose, and $f'_{x,i}$ and $f'_{u,i}$ are the Jacobians of f_i with respect to x_i and u_i respectively, evaluated at (x_i, u_i). Then $\bar{u}_i = \partial z / \partial u_i$ by the chain rule, and a necessary condition for optimality is that $\bar{u}_i = 0$ for $0 \leq i < N$.

In this direct formulation, the Np independent variables are the are the components of the control vectors $u_i : 0 \leq i < N$. The variables $\bar{u}_i, \bar{x}_i$ and x_i are all dependent. We remark in passing that this direct formulation differs from a *Hamiltonian* formulation in which Lagrange multipliers λ_i corresponding to the state evolutions f_i are also independent variables, although at the optimum the dependent variables $\bar{x}_i$ in the direct formulation have the same values as the λ_i have in the Hamiltonian formulation.

Suppose that we linearize both the state and adjoint state equations with respect to a perturbation $t_i = \Delta u_i$ of the control variables $(u_i : 0 \leq i < N)$. Then the Newton direction is the value of t for which the linearizations of the adjoint controls $\bar{u}_i$ all vanish, ie such that $\bar{u} + t \cdot \nabla_u \, \bar{u} = 0$ or

$$\bar{u}_i + \sum_{j=0}^{N-1} \frac{\partial \bar{u}_i}{\partial u_j} t_j = 0 \quad \text{for } 0 \leq i < N.$$

In the early stages of optimization, the $\bar{u}_i$ may be quite large. Pantoja's algorithm constructs, for arbitrary starting values u, the Newton direction t.

In the *indirect* formulation, the only independent variables are the q components of a *costate* vector $\tilde{x}_0$ corresponding to the initial timestep. At each timestep i, the current control vector u_i and the successor costate $\tilde{x}_{i+1}$ are implicitly defined, in terms of the current state x_i and current costate $\tilde{x}_i$, by the costate equations

$$\tilde{x}_i - [f'_{x,i}(x_i, u_i)]^T \tilde{x}_{i+1} = 0$$

and the Pontryagin equations

$$[f'_{u,i}(x_i, u_i)]^T \tilde{x}_{i+1} = 0.$$

These implicit equations are generally non-linear in u_i, and so must be solved by some iterative method. Once values are available for the current control u_i and the successor costate $\tilde{x}_{i+1}$, the state equation $x_{i+1} = f_i(x_i, u_i)$ then gives x_{i+1} in terms of x_i and u_i.

In the indirect formulation, the requirement for the path to be optimal is that $\tilde{x}_N - F'(x_N) = 0$ which we observe can be regarded as a form of the transversality condition. The residual value

$$r = \tilde{x}_N - F'(x_N)$$

of the transversality equation thus gives a measure of how far the initial costate value $\tilde{x}_0$ differs from that for the optimum path. The outer optimization iteration must then adjust the value of the initial costate $\tilde{x}_0$ so as to reduce the size of r. See [12, Chapter 3] for a very early AD implementation of an approach in this spirit.

The remarkable fact which we demonstrate in this paper is that Pantoja's algorithm can be transformed into an algorithm which gives exactly the Newton step a_0 for the initial costate $\tilde{x}_0$ with respect to the transversality condition, ie such that $r + a_0 \cdot \partial r / \partial \tilde{x}_0 = 0$ or

$$\tilde{x}_N - F'(x_N) + a_0 \cdot \frac{\partial}{\partial \tilde{x}_0} \left(\tilde{x}_N - F'(x_N) \right) = 0.$$

Particularly in the early stages of optimization, the transversality residual r may be significantly non-zero, and consequently the costate variable values $\tilde{x}_i$ do not correspond directly to the adjoint states $\bar{x}_i = \partial z / \partial x_i$ for non-optimal paths.

An important advantage of the indirect approach is the reduction in the number of independent variables, from Np to q. A number of objections may nevertheless be made to the indirect method, and we consider these briefly here. (i) The costate equations may be unstable in the forward direction. Of course, this depends upon the problem dynamics: it is not the case for spacecraft trajectory optimization, for example, where the dynamics is given by the inverse square law. (ii) The formation of the adjoint state equations and the transversality equations can be time consuming, and is problem dependent. The need for explicit equation formation can be avoided by the use of AD. Of course, the equations must still be solved for the controls and costates, but AD can again be used to generate the necessary second derivative coefficients. (iii) It is not in general possible to solve the implicit costate and Pontryagin equations exactly: once their residuals are of the same order as the residual r of the transversality equation, no further progress can be made using the calculated Newton step for the initial costate. We shall see following Proposition 2 below how the indirect algorithm can be modified so as to take account of small non-zero residuals. Consequently, Newton's method remains viable in the end-game for the indirect formulation. (iv) The costate variables do not have a physical meaning, and so it can be hard to make good initial estimates of $\tilde{x}_0$. At the optimum, the costate variable value $\tilde{x}_0$ is equal to the value of the adjoint state $\bar{x}_0$: both represent the sensitivity of the optimal value of the target function z to a small perturbation

of the initial state x_0 for fixed control values. In many cases, prior experience with problems of the same general type provides insight into appropriate initial values. Alternatively an adjoint-control transformation [9] can be used to replace guesses for the initial costate by guesses for the initial controls u_0 and certain of their time-derivatives $\dot{u}_0$. Finally, this objection loses much of its force in the context of rigorous global optimization, where the objective is to search the entire independent variable space.

3 The Direct Algorithm

Given a starting position u_i and arbitrary values b_i for $0 \leq i < N$, the following algorithm obtains values for t_i such that $t \cdot \nabla_u \ \bar{u} = b$, ie

$$\sum_{j=0}^{N-1} \frac{\partial^2 z}{\partial u_i \partial u_j} \ t_j = \sum_{j=0}^{N-1} \frac{\partial \bar{u}_i}{\partial u_j} \ t_j = b_i \quad \text{for } 0 \leq i < N.$$

Algorithm 1

Step 1. For i from 1 up to N calculate x_i by $x_{i+1} = f_i(x_i, u_i)$ where x_0 is a fixed constant. Set $z = F(x_N)$.

Step 2. Define $\bar{x}_N, a_N \in R^q; D_N \in R^{q \times q}$ by

$$\bar{x}_N = F'(x_N); \qquad D_N = F''(x_N); \qquad a_N = 0.$$

Step 3. For i from $N-1$ down to 0 calculate $\bar{x}_i, a_i \in R^q; \bar{u}_i, c_i \in R^p; A_i, D_i \in R^{q \times q}; B_i \in R^{p \times q}; C_i \in R^{p \times p}$ by

$$\bar{x}_i = \left[f'_{x,i}\right]^T \bar{x}_{i+1}; \qquad \bar{u}_i = \left[f'_{u,i}\right]^T \bar{x}_{i+1}$$

$$A_i = \left[f'_{x,i}\right]^T D_{i+1} \left[f'_{x,i}\right] + (\bar{x}_{i+1})^T \left[f''_{xx,i}\right]$$

$$B_i = \left[f'_{u,i}\right]^T D_{i+1} \left[f'_{x,i}\right] + (\bar{x}_{i+1})^T \left[f''_{ux,i}\right]$$

$$C_i = \left[f'_{u,i}\right]^T D_{i+1} \left[f'_{u,i}\right] + (\bar{x}_{i+1})^T \left[f''_{uu,i}\right]$$

where $[.]$ denotes evaluation at (x_i, u_i), and we write (for example)

$$\left(\left[f'_{u,i}\right]^T D_{i+1} \left[f'_{x,i}\right]\right)_{j,k} \text{ for } \sum_{l=1}^{q} \sum_{m=1}^{q} \left[\frac{\partial (x_{i+1})_l}{\partial (u_i)_j}\right] (D_{i+1})_{l,m} \left[\frac{\partial (x_{i+1})_m}{\partial (x_i)_k}\right] \text{ etc.}$$

If C_i is singular then the algorithm fails, otherwise set

$$D_i = A_i - B_i^T C_i^{-1} B_i$$

$$c_i = \left[f'_{u,i}\right]^T a_{i+1} - b_i; \qquad a_i = \left[f'_{x,i}\right]^T a_{i+1} - B_i^T C_i^{-1} c_i$$

Step 4. For i from 0 up to $N-1$ calculate $t_i \in R^p, s_{i+1} \in R^q$ by

$$t_i = -C_i^{-1}(B_i s_i + c_i); \qquad s_{i+1} = \left[f'_{x,i}\right] s_i + \left[f'_{u,i}\right] t_i$$

where s_0 is the fixed constant 0.

STOP

Notation.

Let H be the block matrix with (i,j)-th block given by

$$H_{ij} = \left[\frac{\partial \bar{u}_i}{\partial u_j}\right]$$

For given values of $t_i, b_i \in R^p : 0 \le i < N$ we write $Ht = b$ to denote

$$\sum_{j=0}^{N-1} H_{ij} t_j = b_i \quad \text{for } 0 \le i < N.$$

Proposition 1

Either Algorithm 1 fails because some C_i is singular, or else it terminates with t_i which satisfy $Ht = b$. Consequently if all the C_i defined in Step 3 of Algorithm 1 are invertible, then so is H.

If all the C_i are positive definite, then so is H. Conversely, if H is positive definite then all the C_i are positive definite (and hence are invertible). In this case all eigenvalues of every C_i are bounded below by the smallest eigenvalue λ_0 of H. □

The assertion that positive definiteness for all the C_i implies the same property for H was proved by Pantoja [15]. The converse property was first proved much later by Coleman and Liao [7]. It follows that we can establish whether or not H is positive definite at a point of interest by running the algorithm and checking the N $p \times p$ matrices C_i for positive definiteness. The assertion about the eigenvalues is proved in Corollary 6 of §4 below.

In the particular case where $b = -\bar{u}$, Algorithm 1 is equivalent to the following modified form: in Step 1 replace the definition of a_N by $a_N = \bar{x}_N$; in Step 2 simplify the calculation for c_i to $c_i = [f'_{u,i}]^T a_{i+1}$. Actually this modified algorithm just calculates $y_i = \bar{x}_i + a_i$ in place of a_i, since if $b_i = -\bar{u}_i$ then y_i and c_i satisfy the recurrence relations:

$$c_i = \left[f'_{u,i}\right]^T y_{i+1}; \qquad y_i = \left[f'_{x,i}\right]^T y_{i+1} - B_i^T C_i^{-1} c_i.$$

This modified form is the original algorithm given by Pantoja in [15]. Clearly in this case t is the Newton direction. The generalization to an arbitrary right

hand side b given by Coleman and Liao [*op cit*] applies the original algorithm to a modified target function. Algorithm 1 in the form given here is more efficient in terms of operation count: see [5] for details. The correctness of Algorithm 1 follows from the material set out in the next section.

Near a second-order minimum, it follows by Proposition 1 that all the C_i are invertible, so this algorithm will successfully produce the Newton direction in the end-game. Further, we can verify that we are at a minimum by checking for each i that $\bar{u}_i = 0$ and that C_i is positive definite. This result is often useful even if we did not use Newton to find the optimal point.

The account given thus far assumes that all control variables are active. The value of $\bar{u}$ will indicate when a bound control value should be released, and in the meantime rows and columns corresponding to bound control variables can simply be deleted. If the whole of u_i is bound then $D_i = A_i$.

It is also assumed that state-space constraints have been incorporated into the target variable by the use of barrier or penalty methods. A vast literature exists on the incorporation of constraints into Newton's method. The fact that Pantoja's algorithm, in contrast with other methods such as DDP [11, 13], produces the exact Newton step and not an approximation to it, even far from the optimum, means that we can use all the available theory and tools directly. See [16] for a particularly interesting approach in the context of Pantoja's algorithm.

In case one of the C_i fails to be positive definite in Algorithm 1, indicating that the Hessian H fails to be positive definite, then we can modify the calculation to ensure termination of the algorithm and generation of a descent direction, by changing the recurrence relation for C_i to

$$C_i = \left[f'_{u,i}\right]^T D_{i+1} \left[f'_{u,i}\right] + (\bar{x}_{i+1})^T \left[f''_{uu,i}\right] + \Lambda_i$$

where Λ_i is a symmetric $p \times p$ matrix chosen such that $C_i + \Lambda_i$ is positive definite.

However, if some C_i is not positive definite then it is probably more effective in practice to deploy an alternative strategy rather than damping. A (conceptual) description of one such strategy follows: Rotate u_i in R^p so that the coordinate directions are eigenvectors of C_i. In calculating the Newton step using Algorithm 1, treat as inactive (bound) those components of u_i corresponding to non-positive eigenvalues (ie those less than some positive threshold λ_0). For each significantly negative eigenvalue (less than $-\lambda_0$), a corresponding concave control regime u_j for $i \leq j < N$ can be constructed using Corollary 6 below, and a control update Δu can then be formed by combining these regimes (or a descent regime in the case of eigenvalues close to zero) together with the Newton direction, using techniques such as those suggested in [6, §4].

Implementation using Automatic Differentiation

Algorithm 1 requires accurate (truncation-free) second derivative values, because these values occur in the recurrence relations which generate the linear equations to be solved at each time stage. Using Automatic Differentiation [10], existing

code to calculate the target function z does not require extensive (and error-prone) manual re-writing in order to evaluate these derivatives.

The forward accumulation technique of AD associates with each program variable v a vector $\dot{v}$, which contains numerical values for the partial derivatives of v with respect to each of the r independent variables. The combined structure $V = (v, \dot{v})$ is called a *doublet*.

In the reverse accumulation technique of AD, a floating point *adjoint* variable $\bar{v}$ (initially zero) is associated with each program variable v. The adjoint variables $\bar{v}$ are updated, in the reverse order to the forward computation, so that at each stage they contain the numerical value of the partial derivative of the dependent variable with respect to the value which was contained in v at the corresponding point in the forward computation.

The forward and reverse techniques can be combined to calculate Hessians. We embed doublet arithmetic into an implementation of reverse AD: each program variable value is a doublet rather than a real, and so is each corresponding adjoint variable value. After initializing $[\dot{u}]$ we calculate $Y = f(U)$ giving $\dot{y} = [f'(u)]\dot{u}$. We then initialize $\bar{Y} = (\bar{y}, \dot{\bar{y}})$ and perform the reverse pass in doublet arithmetic, following which we have

$$[\bar{u}] = [f'(u)]^T \bar{y} \quad \text{and} \quad [\dot{\bar{u}}] = \bar{y}^T [f''(u)]\dot{u} + [f'(u)]^T \dot{\bar{y}}$$

The values required by Pantoja's algorithm can be evaluated by choosing suitable initial values for $\dot{u}$ and $\bar{Y}$: for details see [5, 3].

Checkpointing can be used to reduce the total storage requirement for Algorithm 1 to the order of $4p$ floating point stores per time step. Details of this and of the corresponding time and space bounds are reported in [5].

4 Technical Lemmata

In this section we establish a number of technical constructions and results, within the framework of Algorithm 1. For expository purposes, this section is set out in the form of an elementary direct proof of Proposition 1. However our ulterior purpose is to develop a toolkit which can be used to fabricate and analyse the properties of variant algorithms, including the indirect reformulation of Pantoja's algorithm in the following section. The result in Corollary 6 is new, and is needed to justify some of the constructions proposed in [6].

Notation

Given values of $t_i : 0 \leq i < N$ define the directional derivative

$$\frac{\partial}{\partial t} \equiv t \cdot \nabla_u = \sum_{i=0}^{N-1} t_i \cdot \frac{\partial}{\partial u_i}$$

Set

$$s_i = \left[\frac{\partial x_i}{\partial t}\right], \quad \bar{s}_i = \left[\frac{\partial \bar{x}_i}{\partial t}\right], \quad \bar{t}_i = \left[\frac{\partial \bar{u}_i}{\partial t}\right].$$

Note that $\bar{t} = Ht$, ie given values for $t_i : 0 \leq i < N$ we have $\bar{t}_i = \sum_{j=0}^{N-1} H_{ij} t_j$.

Lemma 1

The $t_i, s_i, \bar{s}_i, \bar{t}_i$ satisfy the following recurrence relations:

$$s_0 = 0; \qquad s_{i+1} = \left[f'_{x,i}\right] s_i + \left[f'_{u,i}\right] t_i \quad \text{for } 0 \le i < N$$

$$\bar{s}_N = \left[F''(x_N)\right] s_N$$

$$\bar{s}_i = \left[f'_{x,i}\right]^T \bar{s}_{i+1} + \left(\bar{x}_{i+1}\right)^T \left[f''_{xx,i}\right] s_i + \left(\bar{x}_{i+1}\right)^T \left[f''_{xu,i}\right] t_i \quad \text{for } 0 < i < N$$

$$\bar{t}_i = \left[f'_{u,i}\right]^T \bar{s}_{i+1} + \left(\bar{x}_{i+1}\right)^T \left[f''_{ux,i}\right] s_i + \left(\bar{x}_{i+1}\right)^T \left[f''_{uu,i}\right] t_i \quad \text{for } 0 \le i < N$$

where we write (for example)

$$\left(\left(\bar{x}_{i+1}\right)^T \left[f''_{xu,i}\right] t_i\right)_j \text{ for } \sum_{k=1}^{q} \sum_{l=1}^{p} \left(\bar{x}_{i+1}\right)_k \left[\frac{\partial (x_{i+1})_k}{\partial (x_i)_j \partial (u_i)_l}\right] (t_i)_l \text{ etc.}$$

Proof: Apply the differential operator $\partial/\partial t$ to the equations $x_{i+1} = f_i(x_i, u_i)$; $\bar{x}_N = F'(x_N)$; $\bar{x}_i = [f'_{x,i}]^T \bar{x}_{i+1}$ and $\bar{u}_i = [f'_{u,i}]^T \bar{x}_{i+1}$. □

Lemma 2

Fix $i : 1 \le i \le N$. Suppose that Algorithm 1 has been successfully carried out as far as the inversion of C_i in Step 3. Suppose that we are given an arbitrary value for s_i. Pick

$$t_j = -C_j^{-1} B_j s_j \quad \text{for } j \text{ from } i \text{ up to } N-1.$$

Then $\bar{t}_j = 0$ for $i \le j < N$ and $\bar{s}_j = D_j s_j$ for $i \le j \le N$.

Proof: In case $i = N$ there are no values for $\bar{t}_j$ to be considered, and the last assertion is trivially satisfied since $D_N = F''(x_N)$. For $i < N$ assume as induction hypothesis that the required assertions are true for $i+1$. Set an arbitrary value for t_i. Then $s_{i+1} = \left[f'_{x,i}\right] s_i + \left[f'_{u,i}\right] t_i$. By the induction hypothesis we can construct, for this s_{i+1}, values of $t_j : j > i$ such that $\bar{t}_j = 0$ for $j > i$, and for this choice of t_j we have $\bar{s}_{i+1} = D_{i+1} s_{i+1}$. Hence expanding the recurrence relations for $\bar{t}_i$ and $\bar{s}_i$ in Lemma 1 we have

$$\bar{s}_i = A_i s_i + B_i^T t_i; \qquad \bar{t}_i = B_i s_i + C_i t_i.$$

We are assuming that C_i is non-singular, so setting $t_i = -C_i^{-1} B_i s_i$ will ensure that $\bar{t}_i = 0$, and since $D_i = A_i - B_i^T C_i^{-1} B_i$ this choice of t_i gives $\bar{s}_i = D_i s_i$ which establishes the induction step. □

Lemma 3

Fix $i : 1 \leq i \leq N$, and suppose that we are given values $b_j : i \leq j < N$. Suppose that Algorithm 1 has been successfully carried out as far as the inversion of C_i in Step 3. Suppose that we are given an arbitrary value for s_i. Pick

$$t_j = -C_j^{-1}(B_j s_j + c_j) \quad \text{for } j \text{ from } i \text{ up to } N-1$$

Then $\bar{t}_j = b_j$ for $i \leq j < N$ and $\bar{s}_i = D_i s_i + a_i$ for $i \leq j \leq N$.

Proof: In case $i = N$ there are no values for b_j to be considered, and $a_N = 0$ so the last assertion is trivially satisfied by Lemma 1. For $i < N$ assume as induction hypothesis that the required assertions are true for $i+1$. Set an arbitrary value for t_i and let the values for $t_j : j > i$ be constructed from the resulting value of s_{i+1} by the induction hypothesis. Since $\bar{s}_{i+1} = D_{i+1}s_{i+1} + a_{i+1}$ by hypothesis, the recurrence relations in Lemma 1 expand to give

$$\bar{s}_i = A_i s_i + B_i^T t_i + \left[f'_{x,i}\right]^T a_{i+1}; \qquad \bar{t}_i = B_i s_i + C_i t_i + \left[f'_{u,i}\right]^T a_{i+1}$$

We are assuming that C_i is non-singular, so setting $t_i = -C_i^{-1}(B_i s_i + c_i)$ will ensure that $\bar{t}_i = b_i$, and this choice of t_i gives $\bar{s}_i = D_i s_i + a_i$ which establishes the induction step. □

Putting $i = 0$ in Lemma 3 gives the first part of Proposition 1. Since b is arbitrary and the C_i do not depend on the choice of b_i it follows that H is invertible if all the C_i are, although the converse may fail.

Lemma 4

If all the matrices C_i are positive definite, then so is H.

Proof: Choose arbitrary t_j for all $0 \leq j < N$ and set

$$q(t) = \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} t_j^T H_{jk} t_k = \sum_{j=i}^{N-1} t_j \cdot \bar{t}_j$$

It is required to show that $q(t) > 0$ provided $t_j \neq 0$ for some j. Let i be the least index with $t_i \neq 0$, so $t_j = 0$ for $j < i$.

Define t_j^o by $t_j^o = 0$ for $j < i$, $t_i^o = t_i$, and for $j > i$ choose t_j^o by Lemma 2 so $t_j^o : j > i$ are chosen by Lemma 2 so as to make $\bar{t}_j^o = 0$ for $j > i$. Define $t_j^* = t_j - t_j^o$ for $0 \leq j < N$. Note that $t_j^* = 0$ for $j \leq i$. Now

$$q(t) = \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} t_j^{*T} H_{jk} t_k^* + 2\sum_{j=0}^{N-1}\sum_{k=0}^{N-1} t_j^{*T} H_{jk} t_k^o + \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} t_j^{oT} H_{jk} t_k^o$$

$$= q(t^*) + 2\sum_{j=0}^{N-1} t_j^* \cdot \bar{t}_j^o + \sum_{j=0}^{N-1} t_j^o \cdot \bar{t}_j^o = q(t^*) + t_i^o \cdot \bar{t}_i^o = q(t^*) + t_i^T C_i t_i$$

since $\bar{t}_i^o = C_i t_i^o$ by Lemma 2 and $t_i^o = t_i \neq 0$. Now C_i is positive definite whence $q(t) > q(t^*)$. If $t^* = 0$ then $q(t^*) = 0$, otherwise since $t_j^* = 0$ for $j \leq i$ by construction, applying the argument recursively to t^* gives $q(t^*) > 0$. Hence $q(t) > 0$ in any case. □

Lemma 5

Suppose that H is positive definite. Then so are all the C_i.

Proof: Set $t_j = 0$ for $j < i$ and pick an arbitrary $t_i \neq 0$. If $i < N-1$ then suppose by induction that C_j is positive definite for all $j > i$, and use Lemma 2 applied to $i+1$ to construct $t_j : j > i$ so that $\bar{t}_j = 0$ for $j > i$.

Since H is positive definite we have that $q(t) > 0$ where

$$q(t) = \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} t_j^T H_{jk} t_k = \sum_{j=i}^{N-1} t_j \cdot \bar{t}_j = t_i \cdot \bar{t}_i = t_i^T C_i t_i$$

since $s_i = 0$ so $\bar{t}_i = C_i t_i$. Since t_i is arbitrary we have that C_i is positive definite. □

Corollary 6

Suppose that H is positive definite, with smallest eigenvalue $\lambda_0 > 0$, choose arbitrary i and let λ_i be any eigenvalue of C_i. Then $\lambda_i \geq \lambda_0$.

Proof: We have $\lambda_i > 0$ by Lemma 5. Let t_i be any non-zero eigenvector of C_i with eigenvalue λ_i. Use this t_i in Lemma 5 to construct t_j such that $t_j = 0$ for $j < i$ and $\bar{t}_j = 0$ for $j > i$. Then

$$\lambda_0 ||t_i||^2 \leq \lambda_0 \sum_{j=i}^{N-1} ||t_j||^2 \leq \sum_{j=0}^{N-1}\sum_{k=0}^{N-1} t_j^T H_{jk} t_k = q(t) = t_i^T C_i t_i = \lambda_i ||t_i||^2$$

whence $\lambda_0 \leq \lambda_i$. □

5 The Indirect Algorithm

In the direct formulation we begin with a trial value for the control values u_i at each timestep. In the indirect formulation, in contrast, we begin with a trial value for the initial costate $\tilde{x}_0$ and use the implicit costate and Pontryagin equations to calculate the controls u_i and successor costates $\tilde{x}_{i+1}$ for each timestep $0 < i \leq N$.

We seek the Newton direction a_0 for the initial costate which will force the linearized transversality condition $r + a_0 \cdot \partial r / \partial \tilde{x}_0 = 0$ to hold, ie a_0 such that

$$\tilde{x}_N - F'(x_N) + a_0 \cdot \frac{\partial}{\partial \tilde{x}_0} \left(\tilde{x}_N - F'(x_N) \right) = 0.$$

The following modified formulation of Pantoja's algorithm obtains this required value for a_0.

Algorithm 2

Step 1. Given the fixed initial value for x_0, set a trial initial value for $\tilde{x}_0$. For i from 0 up to $N-1$ calculate $u_i \in R^p; \tilde{x}_{i+1}, x_{i+1} \in R^q$ by solving the implicit costate and Pontryagin equations, respectively

$$\tilde{x}_i - \left[f'_{x,i} \right]^T \tilde{x}_{i+1} = 0; \qquad \tilde{u}_i = \left[f'_{u,i} \right]^T \tilde{x}_{i+1} = 0,$$

for u_i and $\tilde{x}_{i+1}$ and setting $x_{i+1} = f_i(x_i, u_i)$.

Step 2. Set $z = F(x_N)$, and define $a_N \in R^q, D_N \in R^{q \times q}$ by

$$D_N = F''(x_N); \qquad a_N = -r \quad \text{where } r = \tilde{x}_N - F'(x_N).$$

Step 3. For i from $N-1$ down to 0 calculate $a_i \in R^q; A_i, D_i \in R^{q \times q}; B_i \in R^{p \times q}; C_i \in R^{p \times p}$ by

$$A_i = \left[f'_{x,i} \right]^T D_{i+1} \left[f'_{x,i} \right] + (\tilde{x}_{i+1})^T \left[f''_{xx,i} \right]$$

$$B_i = \left[f'_{u,i} \right]^T D_{i+1} \left[f'_{x,i} \right] + (\tilde{x}_{i+1})^T \left[f''_{ux,i} \right]$$

$$C_i = \left[f'_{u,i} \right]^T D_{i+1} \left[f'_{u,i} \right] + (\tilde{x}_{i+1})^T \left[f''_{uu,i} \right]$$

where $[.]$ denotes evaluation at (x_i, u_i), and we write (for example)

$$\left(\left[f'_{u,i} \right]^T D_{i+1} \left[f'_{x,i} \right] \right)_{j,k} \text{ for } \sum_{l=1}^{q} \sum_{m=1}^{q} \left[\frac{\partial (x_{i+1})_l}{\partial (u_i)_j} \right] (D_{i+1})_{l,m} \left[\frac{\partial (x_{i+1})_m}{\partial (x_i)_k} \right] \text{ etc.}$$

If C_i is singular then the algorithm fails, otherwise set

$$D_i = A_i - B_i^T C_i^{-1} B_i$$

$$a_i = \left[f'_{x,i} \right]^T a_{i+1} - B_i^T C_i^{-1} \left[f'_{u,i} \right]^T a_{i+1}$$

STOP

Proposition 2

Either Algorithm 2 fails to terminate, or else at the end a_0 satisfies

$$\tilde{x}_N - F'(x_N) + a_0 \cdot \frac{\partial}{\partial \tilde{x}_0} \left(\tilde{x}_N - F'(x_N) \right) = 0.$$

Proof: Let $a \in R^q$ be an arbitrary direction and for $0 \leq i < N$ define the directional derivative

$$\frac{\partial}{\partial a} \equiv a \cdot \frac{\partial}{\partial \tilde{x}_0}$$

Set

$$t_i = \left[\frac{\partial u_i}{\partial a}\right], \quad s_i = \left[\frac{\partial x_i}{\partial a}\right], \quad \tilde{s}_i = \left[\frac{\partial \tilde{x}_i}{\partial a}\right].$$

Modify Algorithm 2 to initialize $a_N = \partial r/\partial a$. We show by reverse induction that for $0 \leq i \leq N$ we have $\tilde{s}_i = D_i s_i + a_i$. The case $i = N$ is trivial since by the choice of a_N

$$a_N = \frac{\partial r}{\partial a} = \frac{\partial}{\partial a}\left(\tilde{x}_N - F'(x_N)\right) = \tilde{s}_N - F''(x_N)s_N = \tilde{s}_N - D_N s_N.$$

Now assume case $i+1$. Differentiating the state equation with respect to a gives $s_{i+1} = \left[f'_{x,i}\right] s_i + \left[f'_{u,i}\right] t_i$. Differentiating the Pontryagin equation with respect to a and substituting $\tilde{s}_{i+1} = D_{i+1}s_{i+1} + a_{i+1}$ gives

$$B_i s_i + C_i^T t_i + \left[f'_{u,i}\right]^T a_{i+1} = 0, \text{ whence } t_i = -C_i^{-1}\left(B_i^T s_i + \left[f'_{u,i}\right]^T a_{i+1}\right).$$

Differentiating the costate equation and substituting now gives

$$\tilde{s}_i = A_i s_i + B_i^T t_i + \left[f'_{x,i}\right]^T a_{i+1} = D_i s_i + a_i$$

as required for the induction step. The case $i = 0$ gives $a = \partial \tilde{x}_0/\partial a = \tilde{s}_0 = D_0 s_0 + a_0 = a_0$ since $s_0 = 0$. This holds for any value of $a \in R^q$. In particular, if $\partial r/\partial a = 0$ then $a = 0$, thus the operator which maps a to $\partial r/\partial a$ is rank-preserving and hence is onto. It follows that $\partial r/\partial a = -r$ for some value of a, and this value is the required Newton step. Consequently if we initialize $a_N = -r$ in Algorithm 2 then a_0 is the Newton direction as asserted. □

As for Algorithm 1, in the region of an optimum path we have that all the C_i are positive definite.

Self-Stabilization

In Algorithm 2 we are solving implicit equations for the controls and costate at each timestep, and these equations may have small but non-zero residuals for the accepted solutions. Close to a solution, when the transversality residual r also becomes small, the non-zero residuals for the implicit control and costate equations may have a significant impact upon the effectiveness of the calculated Newton step in reducing the transversality residual. It is straightforward to generalize Algorithm 2 so as to compensate to first order for the effect of these residuals in the calculation of the Newton direction.

Suppose the implicit equations actually solved in Step 1 of Algorithm 2 are:

$$\tilde{x}_i - \left[f'_{x,i}\right]^T \tilde{x}_{i+1} = r_i; \qquad \left[f'_{u,i}\right]^T \tilde{x}_{i+1} = -b_i.$$

Modify Step 3 of Algorithm 2 to compute:

$$c_i = \left[f'_{u,i}\right]^T a_{i+1} - b_i; \qquad a_i = \left[f'_{x,i}\right]^T a_{i+1} - r_i - B_i^T C_i^{-1} c_i.$$

To see that this modification has the desired effect, observe that setting $\tilde{x}_0 + a_0$ in place of $\tilde{x}_0$, linearizing the state equations, and solving the (linearized) costate and Pontryagin implicit equations exactly now gives x_N and $\tilde{x}_N$ with $F'(x_N) = \tilde{x}_N$ exactly for quadratic F.

Indeed, we can go further and incorporate the first-order effects of non-zero residuals δx_i in the state equations, by perturbing a_i by $D_i \delta x_i$. Such residuals might arise because the state equation is given in implicit form, or because a multigrid method is being used resulting in a number of different discretizations. A solution which is exact with respect to one discretization may have non-trivial residuals with respect to another.

Once again, the account given here does not consider control or state constraints. Where the Pontryagin equations have no interior solutions for u_i we may bind the relevant controls to an extreme value as discussed following Proposition 1 above. More general state and interior control constraints can be incorporated into the indirect formulation using the penalty approach of [1].

6 Discussion

AD can be used to implement Algorithm 2, in the same way as for Algorithm 1. AD can also be used to calculate derivative values for rapid iterative solution of the implicit equations at each time step.

Of course, in the indirect formulation, these byproducts of AD could instead be used to differentiate forward [2] or backward [4] through the implicit equations to obtain explicitly the Jacobian of the transversality condition, and hence to determine the Newton step, without any reference to a Pantoja-like construction.

However, Algorithm 2 possesses certain additional advantages. (i) Algorithm 2 can be made self-stabilizing. (ii) Algorithm 2 permits good estimates to be made for new trial controls u_i and costates $\tilde{x}_{i+1}$ prior to implicit equation solution at each timestep of the new solution point: $\delta u_i = -C_i^{-1}(B_i \delta x_i + c_i)$, $\delta \tilde{x}_{i+1} = D_{i+1} \delta x_{i+1}$. (iii) Algorithm 2 allows inexpensive post-hoc verification that a second-order minimum has indeed been reached.

The alternative formulations of Pantoja's algorithm set out in this paper also provide a unified framework for consideration of hybrid direct-indirect approaches to discrete time optimal control, and this is identified as a promising area for future research.

For example, a composite of Algorithms 1 and 2 allows simple determination of whether the Hessian of z is positive definite with respect to the controls u_i for any given trial path, and also allows damping to be applied to the Newton step for the initial costate if not. This means that the outer optimization process can be started by choosing a control regime, perhaps through an interactive simulation, and then a self-stabilizing step can be used to modify the initial adjoint state values into sensible trial values for the initial costates.

A similar approach can be taken to post-hoc verification of the smallest eigenvalue of H and hence of the inclusions required for fixed point interval constructions. The number q of independent variables in Algorithm 2 is small compared to the number Np of independent variables in Algorithm 1, which makes the indirect form of Pantoja's algorithm suitable as a basis for verified optimization using interval methods. This is also identified as a challenge for future development.

A final challenge for the future is the incorporation of third and higher order derivative information into solution strategies for optimal control problems, for example allowing curvilinear searches in the style of Conforti and Mancini [8]. This is likely to be of particular importance when state constraints are handled by penalty or barrier methods.

Dedication

At various different times, and amongst many other things, Laurence Dixon has been responsible for introducing me to Automatic Differentiation, to the indirect approach to Optimal Control, and to Pantoja's original algorithm. This paper is dedicated to him, with thanks, for his many years of quiet discernment.

References

1. Michael Bartholomew-Biggs, 1995, A Penalty Method for Point and Path State Constraints in Trajectory Optimization, Optimal Control Applications and Methods, **16**, 291–297.
2. Michael Bartholomew-Biggs, 1998, Automatic Differentiation of Implicit Functions using Forward Accumulation, Computational Optimization and Applications, **9**, 65–84.
3. Michael Bartholomew-Biggs, Steven Brown, Bruce Christianson *and* Laurence Dixon, 2000, Automatic Differentiation of Algorithms, Journal of Computational and Applied Mathematics, **124**, 171–190.
4. Bruce Christianson, 1998, Reverse Accumulation and Implicit Functions, Optimization Methods and Software, **9**, 307–322.
5. Bruce Christianson, 1999, Cheap Newton Steps for Optimal Control Problems: Automatic Differentiation and Pantoja's Algorithm, Optimization Methods and Software, **10**, 729–743.
6. Bruce Christianson *and* Michael Bartholomew-Biggs, 2001, Globalization of Pantoja's Optimal Control Algorithm, *in* From Simulation to Optimization: 3rd International Conference on Automatic Differentiation, Springer LNCS, *to appear*.
7. Thomas Coleman *and* Aiping Liao, 1995, An Efficient Trust Region Method for Unconstrained Discrete-Time Optimal Control Problems, Computational Optimization and Applications **4**, 47–66.
8. D. Conforti *and* M. Mancini, 2001, A Curvilinear Search Algorithm for Unconstrained Optimization, Optimization Methods and Software, *to appear*.
9. Laurence Dixon *and* Michael Bartholomew-Biggs, 1981, Adjoint-Control Transformations for Solving Practical Optimal Control Problems, Optimal Control Applications and Methods, **2**, 365–381.

10. Andreas Griewank, 2000, Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Frontiers in Applied Mathematics **19**, SIAM, Philadelphia.
11. D.H. Jacobson *and* D.Q. Mayne, 1970, Differential Dynamic Programming, *in* Modern Analytic and Computational Methods in Science and Mathematics **24**, ed. R. Bellman, American Elsevier, New York.
12. Harriet Kagiwada *et al*, 1986, Numerical Derivatives and Nonlinear Analysis, Plenum New York.
13. D.M. Murray *and* S.J. Yakowitz, 1984, Differential Dynamic Programming and Newton's Method for Discrete Optimal Control Problems, JOTA **43**(3), 395–414.
14. J.F.A.De O. Pantoja, 1983, Algorithms for Constrained Optimization Problems, PhD thesis, Imperial College of Science and Technology, University of London.
15. J.F.A.De O. Pantoja, 1988, Differential Dynamic Programming and Newton's Method, Int J Control **47**(5), 1539–1553.
16. J.F.A.De O. Pantoja *and* D.Q. Mayne, 1991, Sequential Quadratic Programming Algorithm for Discrete Optimal Control Problems with Control Inequality Constraints, Int J Control **53**(4), 823–836.