Globalization of Pantoja's Optimal Control Algorithm

Bruce Christianson
Michael Bartholomew-Biggs

ABSTRACT In 1983 Pantoja described a stagewise construction of the Newton direction for a general class of discrete time optimal control problems. His algorithm incurs amazingly low overheads: the cost (measured in target function evaluations) is independent of the number of discrete time-steps.

The algorithm can be modified to verify that the Hessian contains no eigenvalues less than a postulated quantity, and to produce an appropriate descent direction in the case where the Hessian fails to be positive definite and global convergence becomes an issue. Coleman and Liao have proposed a specific damping strategy in this context.

Here we describe how Automatic Differentiation can be used to implement Pantoja's algorithm, and we briefly consider some alternative globalization strategies, within which AD techniques can be further deployed.

1 Introduction

Consider the following optimal control problem: choose $u_i \in R^p$ for $0 \leq i < N$ to minimize

$$z = F(x_N)$$

where x_0 is some fixed constant, and $x_{i+1} = f_i(x_i, u_i)$ for $0 \leq i < N$. Here each f_i is a smooth map from $R^q \times R^p \to R^q$, F is a smooth map from R^q to R, and N is the number of discrete time-steps. The dimensions p_i, q_i may depend upon i, but for notational convenience we omit these subscripts.

The more usual formulation of a discrete time optimal control problem, 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 adjoining to each state x_i a new component $v_i \in R$ defined by $v_0 = 0, v_{i+1} = v_i + F_i(x_i, u_i)$ and then defining $F(x_N, v_N) = v_N + F_N(x_N)$. Consequently, we lose nothing by restricting attention to minimizing target functions of the form $z = F(x_N)$.

An obvious approach to solving this problem is to apply Newton's method. Dealing explicitly with quadratic approximations to the f_i is extremely inconvenient when performing actual calculations, because of interactions between the very large number of cross terms. It would be much more convenient if we could calculate the Newton direction in a way which required only linear approximations to the problem dynamics. Pantoja [12, 13] has shown that this can be done, provided we also consider linear approximations to the adjoint problem dynamics.

2 The Generalized Pantoja Algorithm

Let g be the block vector with i-th block given by $g_i = \bar{u}_i = [\partial z/\partial u_i]$. Let H be the block matrix with (i,j)-th block given by $H_{ij} = [\partial \bar{u}_i/\partial u_j] = \left[\partial^2 z/\partial u_i \partial u_j\right]$. Let Λ be a symmetric block diagonal matrix. For block vectors t and b we write $Ht = b$ to signify $\sum_{j=0}^{N-1} H_{ij} t_j = b_i$ for $0 \le i < N$.

Given a starting position u_i and arbitrary values b_i for $0 \le i < N$, to obtain values for t_i such that $(H + \Lambda)t = b$ proceed as follows.

Step 1. For i from 1 up to N calculate $x_{i+1} = f_i(x_i, u_i)$ where x_0 is a fixed constant. Define $\bar{x}_N = F'(x_N), a_N = 0 \in R^q; D_N = F''(x_N) \in R^{q\times q}$.

Step 2. 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

$$\bar{x}_i = \bar{x}_{i+1}\left[f'_{x,i}\right], \qquad \bar{u}_i = \bar{x}_{i+1}\left[f'_{u,i}\right]$$

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

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

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

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

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

If C_i is singular then the algorithm fails. Otherwise set

$$D_i = A_i - B_i^T C_i^{-1} B_i, \qquad E_i = C_i^{-1} B_i$$

$$c_i = a_{i+1}\left[f'_{u,i}\right] - b_i, \qquad a_i = a_{i+1}\left[f'_{x,i}\right] - c_i E_i.$$

Step 3. For i from 0 up to $N-1$ calculate $t_i \in R^p, s_{i+1} \in R^q$ by

$$t_i = -E_i s_i - C_i^{-1} c_i^T, \quad s_{i+1} = \left[f'_{x,i}\right] s_i + \left[f'_{u,i}\right] t_i$$

where $s_0 = 0 \in R^q$. STOP.

Either this algorithm fails because some C_i is singular, or else at the end the t_i satisfy $\widehat{H}t = b$, where $\widehat{H} = H + \Lambda$. Consequently if all the C_i defined in Step 2 of the algorithm are invertible, then so is $\widehat{H}$.

If all the C_i are positive definite, then so is $\widehat{H}$. Conversely, if $\widehat{H}$ 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 $\widehat{H}$, and furthermore $(t,b) = \sum_i t_i \cdot b_i > 0$ provided $b \neq 0$.

In the particular case where $b = -g$ and $\Lambda = O$, the algorithm is equivalent to the following modified form: in Step 1, replace the definition $a_N = 0$ by $a_N = \bar{x}_N$; in Step 2, simplify the calculation for c_i to $c_i = a_{i+1}\left[f'_{u,i}\right]$. This is the original algorithm, given by Pantoja in [13].

3 Implementation using AD

Automatic Differentiation combined with checkpointing [7, 8] can be used to provide efficient implementations of the algorithm introduced in §2 [2, 3]. Accurate (truncation-free) second derivative values are required, because they occur in the recurrence relations which generate the linear equations to be solved at each time stage. Using Automatic Differentiation, existing code to calculate the target function z does not require extensive (and error-prone) re-writing 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*. When the doublets U_k corresponding to the independent variables u_k are initialized by setting $\dot{u}_k$ to be the $k-$th Cartesian unit vector, we write this (rather loosely) as $[\dot{u}] = [I_r]$.

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 y with respect to 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 $\dot{y} = f'(u)\dot{u}$ as before. We then initialize $\bar{Y}$ and perform the reverse pass in doublet arithmetic, following which we have $[\bar{u}] = \bar{y}[f'(u)]$ as before, and

$$[\dot{\bar{u}}] = \bar{y}[f''(u)]\dot{u} + \dot{\bar{y}}[f'(u)]$$

The values required by Pantoja's algorithm can be evaluated by choosing suitable initial values for $\dot{u}$ and $\bar{Y}$.

For example, in Step 1 of the Algorithm, we define doublets X_N with scalar parts x_N respectively, and vector parts (of length q) given by $\dot{x}_N = [I_q]$. Evaluate $Z = F(X_N)$ in doublets, recording a trajectory. We now have

$\dot{z} = [F'(x_N)]$. Define $\bar{Z}$ by setting $\bar{z} = 1.0, \dot{\bar{z}} = 0_q$. Reverse through the computation for Z to obtain the doublets $\bar{X}_N$. We have $\bar{x}_N = [F'(x_N)]^T, \dot{\bar{x}}_N = [F''(x_N)]$. Set $D_N = \dot{\bar{x}}_N$.

Similarly, consider Step 2 of the Algorithm. We wish to calculate $\bar{x}_i, a_i$ and D_i assuming that the corresponding quantities are available for $i+1$. Define doublets X_i and U_i with scalar parts x_i and u_i respectively, and vector parts (of length $q+p$) given by

$$\begin{bmatrix} \dot{x}_i \\ \dot{u}_i \end{bmatrix} = \begin{bmatrix} I_q & O \\ O & I_p \end{bmatrix}.$$

Evaluate $X_{i+1} = f_i(X_i, U_i)$ in doublets. Now we have $[\dot{x}_{i+1}] = [f'_{x,i} \; f'_{u,i}]$. Calculate the vectors $a_{i+1}\left[f'_{x,i}\right]$ and $c_i = a_{i+1}\left[f'_{u,i}\right] - b_i$. Define $\bar{X}_{i+1}$ by setting $\bar{x}_{i+1}$ to the supplied value and setting

$$\begin{bmatrix} \dot{\bar{x}}_{i+1} \end{bmatrix} = \begin{bmatrix} D_{i+1} f'_{x,i} & D_{i+1} f'_{u,i} \end{bmatrix}.$$

Reverse through the trace for X_{i+1} to obtain the doublets $\bar{X}_i$ and $\bar{U}_i$. Then

$$\begin{bmatrix} \dot{\bar{x}}_i \\ \dot{\bar{u}}_i \end{bmatrix} = \begin{bmatrix} A_i & B_i^T \\ B_i & C_i \end{bmatrix}.$$

Row reduction gives

$$\begin{bmatrix} A_i - B_i^T C_i^{-1} B_i & O \\ C_i^{-1} B_i & I \end{bmatrix} = \begin{bmatrix} D_i & O \\ E_i & I \end{bmatrix}.$$

Now $\bar{x}_i, a_i = a_{i+1}\left[f'_{x,i}\right] - c_i E_i$ and D_i are available for the next iteration.

An implementation of Pantoja's algorithm using AD typically requires on the order of $p+q$ times the floating point cost of an evaluation of z, independent of N, together with order $(p+q)^3$ multiply-and-add operations per time step (less if there is structural sparsity). Checkpointing can be used to reduce the total storage requirement to the order of $4p$ floating point stores per time step. Detailed time and space bounds are reported in [3].

4 Globalization Strategies

If H is positive definite, and we take $\Lambda = 0$, then all the C_i will be invertible. So, near a second order minimum, the algorithm will successfully produce the Newton direction t. We can verify that we are at such a minimum by checking for each i that $\bar{u}_i = 0$ and C_i is positive definite, even if we did not use Newton to find the optimal point. The algorithm can also be used to inform us efficiently if H fails to be positive definite at an arbitrary point of interest, or has an eigenvalue falling below a specified threshold.

However, if good initial estimates are not available for the control variables, then H may be indefinite. In this case the algorithm may fail, or may produce a t which is not a descent direction. An alternative strategy to obtain a search direction is required in order to ensure global convergence of the algorithm. A straightforward way of doing this is to choose Λ in such a way as to ensure that all the C_i are positive definite.

Coleman and Liao [4] have proposed modifications to Pantoja's original algorithm to achieve this, based upon the Nocedal-Yuan trust-region method. This approach sets $\Lambda = \lambda I$ and adjusts λ so that the resulting step falls within the trust region r. The adjustment requires the solution for t and d of the pair of equations $(H + \Lambda)t = -g, (H + \Lambda)d = t$, (since to first order $\delta t = \delta\lambda d$, whence $\delta\lambda = (r^2 - t^2)/2t \cdot d$, see [4, §2.2]).

Essentially Coleman and Liao construct a solution of $(H + \lambda I)t = b$ for given λ and b by applying Pantoja's algorithm to a modified target function. In contrast, the approach formulated in §2 constructs such solutions by applying the generalized algorithm, with $\Lambda = \lambda I$, to the original target function. As well as producing a modest reduction in the operation count [3], the generalized algorithm additionally allows the components of b and Λ to be specified dynamically rather than in advance.

Thus rather than restrict Λ to a multiple of the identity, we can choose Λ as the Algorithm proceeds to ensure that $\widehat{H} = H + \Lambda$ has no eigenvalues smaller than the positive value $\lambda_0 = \|g\|/r$. In this case the equation $Ht = -g$ is guaranteed to have a solution with $\|t\| \leq \|g\|/\lambda_0 = r$. To ensure this, we conjointly evaluate the sequence $\widehat{C}_i$ corresponding to $\widehat{\Lambda}_i = \Lambda_i - \lambda_0 I_p$ and choose Λ_i to ensure that the $\widehat{C}_i$ are positive definite at each stage.

A trust region approach is not the only established mechanism for ensuring global convergence. The requirement that Λ_i be chosen to make C_i positive definite in the algorithm amounts just to saying that if, in the course of inverting C_i, we hit a pivot which is not significantly positive, then we can replace that pivot by a positive value (say 10^{-3} scaled by the norm of the corresponding row) and carry on. This process builds up a diagonal matrix Λ with the amounts added to the pivots at stage i forming the corresponding diagonal Λ_i.

The eigenvalues of C_i are bounded below by the smallest eigenvalue of $\widehat{H}$, and the resulting value for t is guaranteed to be a descent direction, with the property that $g + Ht$ is zero everywhere except for blocks corresponding to those timesteps at which the pivots required adjustment. Furthermore the algorithm can readily be extended to produce, corresponding to each such timestep i, a direction $n^{(i)}$ of negative curvature for H, with $n_j^{(i)} = 0, j < i; (Hn^{(i)})_j = 0, j > i$. These concave directions $n^{(i)}$ can be combined with the descent direction t to establish a search direction in a number of ways. This type of combining is examined in a different context in [1].

In one alternative, suggested by Conforti and Mancini [5], a curvilinear search can be performed, selecting among steps of the form $\alpha t + \alpha^2 t'/2$, where $\widehat{H}t' + g' = 0$ and g' is the gradient $\nabla_u \, (t^T Ht)$.

Another possibility is to use AD to establish a quartic model on the subspace of directions, or to carry out a multi-dimensional search in the spirit of Miele and Cantrell [10].

These types of non-linear search, combining directions using accurate higher-order derivative information obtained by AD, offer a promising avenue for future research, particularly where state or control constraints have been incorporated into z by penalty or barrier terms.

5 Concluding Remarks

Pantoja originally introduced his algorithm as a variant of Differential Dynamic Programming (DDP) [9, 11]. Traditional DDP algorithms and other state-control feedback regimes can also be implemented using AD: for example traditional DDP just puts a_{i+1} in place of $\bar{x}_{i+1}$ in the initialization of $\bar{X}_{i+1}$. One advantage of Pantoja's algorithm relative to other state-control feedback methods is that, because it provides the exact Newton step even far from the optimal point, we can use all the theory and tools (eg. trust regions, line searches, termination criteria etc.) already available from the extensive literature on Newton's method.

Pantoja's algorithm operates on a discrete time optimal control problem. An optimal control problem may also be formulated as a continuous problem, see [6, §3]. Continuous variations of Pantoja's algorithm exist in other forms: for example the matrix Riccati method can be viewed in this light. However, continuous problems must be discretized one way or another before being solved on digital computers.

In this paper we take the view that the problem for which the exact Newton step is appropriate is the actual discretization being used. This discretization may change during the course of problem solution (for example by inserting finer time steps in regions where the control variables are changing rapidly), but verification of descent criteria such as Wolfe conditions, and the introduction of devices to enforce global convergence of Newton's method, should be applied to the numerical values actually being calculated.

Pantoja's Algorithm is extremely useful even when a Newton optimization technique is not employed: it allows inexpensive determination of whether a Hessian is positive definite at a given point. As well as being of utility in enforcing global convergence, this feature also allows post-hoc verification of the inclusions required for interval fixed point constructions.

6 References

[1] M. Bartholomew-Biggs, 2000, A Two-dimensional Search used with a Non-linear Least-squares Solver, Journal of Optimization Theory and Applications, **104**(1), 215–234.

[2] M. Bartholomew-Biggs, S. Brown, B. Christianson *and* L.C.W. Dixon, 2000, Automatic Differentiation of Algorithms, Journal of Computational and Applied Mathematics, to appear.

[3] Bruce Christianson, 1999, Cheap Newton Steps for Optimal Control Problems: Automatic Differentiation and Pantoja's Algorithm, Optimization Methods and Software, **10**, 729–743.

[4] 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.

[5] D. Conforti *and* M. Mancini, 2000, A Curvilinear Search Algorithm for Unconstrained Optimization by Automatic Differentiation, *under review.*

[6] J.C. Dunn *and* D.P. Bertsekas, 1989, Efficient Dynamic Programming Implementations of Newton's Method for Unconstrained Optimal Control Problems, JOTA **63**(1), 23–38.

[7] Andreas Griewank, 1992, Achieving Logarithmic Growth of Temporal and Spatial Complexity in Reverse Automatic Differentiation, Optimization Methods and Software, **1**, 35–54

[8] Andreas Griewank *and* Andrea Walther, 2000, Algorithm 799: Revolve: An Implementation of Checkpoint for the Reverse or Adjoint Mode of Computational Differentiation, ACM Transactions on Mathematical Software, **26**(1), 19.

[9] D.H. Jacobson *and* D.Q. Mayne, 1970, Differential Dynamic Programming, American Elsevier, New York.

[10] A. Miele *and* J.W. Cantrell, 1969, Study on a Memory Gradient Method for the Minimization of Functions, Journal of Optimization Theory and Applications, **3**, 459–470.

[11] 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.

[12] J.F.A.De O. Pantoja, 1983, Algorithms for Constrained Optimization Problems, PhD thesis, Imperial College of Science and Technology, University of London.

[13] J.F.A.De O. Pantoja, 1988, Differential Dynamic Programming and Newton's Method, Int J Control, **47**(5), 1539–1553.