Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Order of Convergence in the Direct Transcription Solution of Optimal Control Problems

Anna Engelsone Stephen L. Campbell John T. Betts

Abstract— **In the direct transcription approach to the numerical solution of optimal control problems, the optimal control problem is discretized and the resulting nonlinear programming problem is solved numerically. There has been considerable study over the last 10 years on order of convergence of cost, state, multipliers, and control. This paper discusses these questions, and the highly technical results in the literature, in the context of their implications for industrial grade optimal control packages.**

I. INTRODUCTION

A large number of applications in industry involve, at some point, the numerical solution of an optimal control problem. As a consequence there has been a large amount of research on numerical methods and the theory that lies behind them. Unfortunately, a large part of this research involves theoretical results developed under widely differing assumptions and an algorithmic approach illustrated on one or two test problems. While these results and papers are very enlightening we have found that they can create some confusion.

This paper was motivated by our ongoing work [1] and development of the Sparse Optimal Control Software (SOCS) developed at the Boeing Corporation [2]. One current question is whether the family of available discretizations within the software should be expanded. SOCS is a general purpose optimal control code and it must be able to handle a wide variety of problems, usually with numerous equality and inequality constraints. It is also used by scientists of varying numerical sophistication from several different disciplines.

Like many industrial codes, SOCS uses a direct transcription approach [3]. In order to solve the large complex nonlinear programming problems (NLP) that arise subsequent to discretization, SOCS must exploit sparse linear algebra routines that are tailored to utilize the structure of the discretizations. Decisions interior to the code, such as efficient mesh refinement [4], [5] and setting of NLP parameters, are based on extensive testing and analysis. Thus adding a new discretization is expensive in both time and dollars so the addition must be based on identified advantages.

We shall show that some of the existing results in the literature do not always have the consequences the reader may think they have and that the interpretation of these results may change depending on the intended application of the numerical solution. This is not in any way meant to suggest the results are incorrect or poorly stated. But rather, like with many types of computational results, careful interpretation is sometimes necessary when using them.

In Section II we will introduce the basic optimal control problem, what we mean by a direct transcription method, and summarize some of the important results about them. Then in Section III we carry out a discussion about how to relate and interpret these results. An academic example from [6] will be introduced and used throughout the discussion to illustrate the ideas. Finally we will summarize some of our key observations in Section IV. Our computational studies utilize the sparse optimal control code SOCS. However, the observations have general applicability to other direct transcription codes with a similar philosophy.

II. SOME PRIOR RESULTS

A. The optimal control problem

As noted in the introduction, problems arising from industrial applications frequently include a variety of constraints and in that context one also has to be careful when interpreting classical results. For a discussion of constrained problems, see [7], [8], [9]. In this paper, it suffices to consider unconstrained problems and a fixed terminal time t_f. We suppose that the optimal control problem is

$$\min_u \quad \phi(x(t_f)) + \int_0^{t_f} L(x,u,t)\,dt \tag{1a}$$
$$x' = f(x,u,t) \tag{1b}$$
$$x(t_0) = x_0 \tag{1c}$$

where L is differentiable. This problem can be written as

$$\min_u \quad \phi(x_1(t_f)) + x_2(t_f) \tag{2a}$$
$$x_1' = f(x_1,u,t) \tag{2b}$$
$$x_2' = L(x_1,u,t) \tag{2c}$$
$$x_1(0) = x_0 \tag{2d}$$
$$x_2(0) = 0 \tag{2e}$$

which, after letting $x = (x_1, x_2)$ and $\tilde{x}_0 = (x_0, 0)$, is in the Mayer form

$$\min_u \quad \psi(x(t_f)) \tag{3a}$$
$$x' = g(x,u,t) \tag{3b}$$
$$x(t_0) = \tilde{x}_0 \tag{3c}$$

Anna Engelsone is with the Operations Research Program, North Carolina State University, Raleigh, NC 27695-7913. aengels@ncsu.edu

Steve Campbell is with the Department of Mathematics, North Carolina State University, Raleigh, NC 27695-8205, USA. slc@math.ncsu.edu

John Betts is with Mathematics and Engineering Analysis, The Boeing Company, P.O. Box 3707 MS 7L-21, Seattle, Washington 98124-2207. John.T.Betts@boeing.com

0-7803-9568-9/05/$20.00 ©2005 IEEE

The Mayer form is the one used by SOCS. This means that the mesh refinement used to meet ODE tolerances meets the same tolerances in estimating the cost. Thus in talking about order of convergence we do not need to worry separately about how the cost is being approximated.

B. Direct Transcription

In a direct transcription approach the dynamics are discretized by a numerical integrator. If the cost is in the form of (1a), some type of collocation must be used. If the same problem is put into Mayer form, then the discretization used for the dynamics implicitly also provides a discretization of the cost (1a).

Many optimal control studies have used explicit Runge Kutta methods. SOCS has a fourth order Runge Kutta (RK4) available, but the preferred methods are the trapezoid method (TR) and the Hermite Simpson (HS). Since TR and HS have proved the most useful, and the RK4 has not been as extensively tuned and optimized, we shall discuss TR and HS in this paper. The TR and HS methods are based on spline interpolates which makes the output convenient for a number of applications. They can be viewed as second and fourth order respectively ODE integrators. They can also be viewed as implicit Runge Kutta methods.

SOCS starts with a coarse grid and then does automatic mesh refinement until the solution meets tolerances. The resulting grids are usually highly nonuniform which is necessary to restrict problem size and yet capture key aspects of control action. The default is for SOCS to start with TR to facilitate finding a feasible solution and then switch to HS after one or two iterations. In order to discuss the results in the literature we shall override this feature and force SOCS to refine the mesh by halving and to use the same discretization throughout a particular solution of a particular problem.

The TR discretization of (1b) takes the form

$$x^{i+1} - x^i - \frac{h_i}{2}\left(f(x^i, u^i, t_i) + f(x^{i+1}, u^{i+1}, t_{i+1})\right) = \eta_i \tag{4}$$

where x_i and u_i are the estimates of $x(t_i)$ and $u(t_i)$ at grid points t_i, $h_i = t_{i+1} - t_i$, and η_i is the residual to be reduced to the proscribed level. This generates a large, but sparse, nonlinear system of constraints.

The HS discretization is available in either the compressed or the uncompressed (separated) form. In the former, the NLP variables are x_i, u_i and a midpoint value $\overline{u}_i$. The separated formulation includes additional midpoint values $\overline{x}_i$. The uncompressed version is described by the equations

$$x^{i+1} - x^i - \frac{h_i}{6}\left(f_i + 4\overline{f}_{i+1} + f_{i+1}\right) = \eta_{i1} \tag{5a}$$

$$\overline{x}^i - \frac{1}{2}(x^i + x^{i+1}) - \frac{h_i}{8}(f_i - f_{i+1}) = \eta_{i2} \tag{5b}$$

where

$$\overline{f}_{i+1} = f(\overline{x}^i, \overline{u}^i, t_i + \frac{h_i}{2}).$$

The compressed version collapses this to one equation per time step.

C. Some prior work

Because of their importance in applications there has been prior investigation of RK methods and direct transcription. Space limitations prevent a full citation of the literature. We note only [10], [11], [12], [13], [14], [15]. Our emphasis here is on examining the implications and interpretation of some of the order results. The order of the method describes the rate at which the error approaches zero as a function of the (constant) mesh size. Note that there are several types of order.

To facilitate the discussion we suppose that there is an optimal solution x^*, u^* and an optimal cost c^*. Let λ^* be the continuous multipliers from the necessary conditions. λ^* is also referred to as the adjoint variable. Given N equally spaced mesh points t_i between 0 and t_f with $t_0 = 0, t_N = t_f$, the direct transcription method computes a value of the state x_N, control u_N, and cost c_N. There are also discrete multipliers which are available from the solution of the NLP. We denote these λ_N. Here $x_N = [x_N^0, \ldots, x_N^{N-1}]$ so that x_N^i is the approximation of $x(t_i)$ with N grid points. Similar notation is used with u_N. How to interpret λ_N will be discussed later. We sometimes ommit the N as in (4) and (5a).

In 1976, W. Hager proved second order convergence of u_N obtained by numerical methods using certain explicit Runge-Kutta discretizations for a class of unconstrained problems [16]. A. Dontchev was the first to prove a convergence result for constrained problems, in this case linear problems with decoupled control and state inequality constraints discretized using Euler's scheme [17]. In [6], Hager proved high order convergence of states and multipliers in problems with generic control constraints, obtained using any high order Runge-Kutta discretization with $\beta_i > 0$ for all i. In [18], Dontchev, Hager and Veliov proved second order convergence of controls for a specialized class of second-order Runge-Kutta discretizations.

Both [6] and [18] give numerical results demonstrating that for many other methods, and even very simple linear problems, the controls are often found to a much lower accuracy than the states and multipliers. However, Hager suggests that this could be remedied by post-computing to get an estimate u_{PN} of u. That is, u_{PN} is computed by minimizing, at each mesh point t_i, the Hamiltonian

$$H(x_N^i, \lambda_N^i, u_{PN}^i) = \lambda_N^i g(x_N^i, u_{PN}^i, t_i) \tag{6}$$

The estimate u_{PN} is proved by Hager to be the same order of accuracy as x_N and λ_N. This idea can be useful as a post-computing method for some problems but requires good multiplier estimates. For general high dimensional problems with many constraints it is often not practical. One big advantage of a direct transcription approach is that formulation of the necessary conditions and the Hamiltonian is not necessary and good multiplier estimates are not required. In problems with inequality constraints on state and control the discrete multipliers coming from the discretizations may be poor estimates of the adjoint variables or in some cases

not even convergent [8]. In the discussion that follows we will compute u_{PN} since it helps in understanding what is happening.

Convergence of the cost c_N is considered, for example, in [19] where it is shown that if a Runge-Kutta discretization is used which is locally at least of third order, and if the dynamics are affine in u, then the cost is at least second order. This, of course, includes both TR and HS. It also includes the example (7).

There are two impressions that stand out from this and related work. One is that there are more theoretical results for explicit methods than for implicit methods. The second is that with some methods we get reduced order of convergence in u_N. This would seem to be not a good thing. But if we have a problem in Mayer form, then the cost, since it is a part of the state, should be to the same order as the state. What does it mean to have the cost to higher order than the control that gives the cost? In this regard, it should be noted that with many software packages even if the user thinks of the problem as being in the form (1a), the cost functions are input separately, and the actual problem passed on to software is in Mayer form. SOCS is such a code.

III. Discussion & Computational Example

To make our discussion more concrete we shall use the following example from [6] where we omit the factor 0.5 in the cost. The optimization problem is

$$\min \quad \int_0^1 u(t)^2 + x(t)u(t) + \frac{5}{4}x(t)^2 dt \tag{7a}$$

$$x'(t) = 0.5x(t) + u(t) \tag{7b}$$

$$x(0) = 1 \tag{7c}$$

The exact optimal solution to this problem is given by

$$x^*(t) = \frac{\cosh(1-t)}{\cosh(1)} \tag{8a}$$

$$u^*(t) = -\frac{(\tanh(1-t)+0.5)\cosh(1-t)}{\cosh(1)} \tag{8b}$$

$$\lambda^*(t) = \frac{2\cosh(1-t)\tanh(1-t)}{\cosh(1)} \tag{8c}$$

The value of the optimal cost can be computed using Maple to be 0.7615941557. This example is of interest since it is known to show order reduction in the solution for the optimal control.

The functions x^*, u^*, and λ^* are graphed in Figure 1.

A. Using TR

Suppose that we solve (7) using TR on uniform grids overriding the usual mesh refinement strategy and compare the results on each iteration with the true solution (8). The error, as is typical in numerical analysis, is measured in the sup norm denoted here by $\|\cdot\|$. The results are in Table I. Note that unlike [6], here N is the number of grid points so that there are $N-1$ intervals.

Consistent with the theoretical results mentioned earlier in [6] we see that there is second order convergence in x and λ since the error is reduced by about $1/4$, equivalently the $-\log_2$ of the norm increases by 2, each time the mesh is halved. Note that we are showing first order convergence in the control. A post-computed control would be second order.

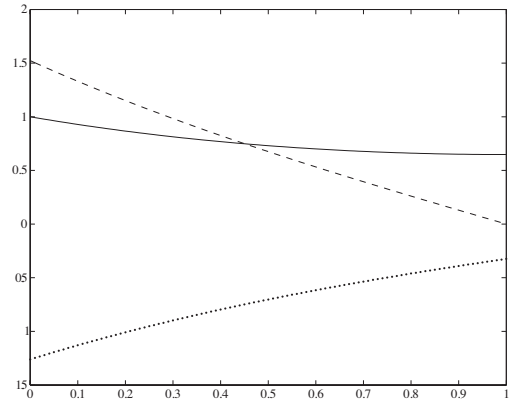

Fig. 1. Graph of x^* (solid line), u^* (dotted line), and λ^* (dashed line).

TABLE I
$-\log_2$ of L_∞ error in outputs using TR.

N	$\|x^* - x_N\|$	$\|u^* - u_N\|$	$\|\lambda^* - \lambda_N\|$	$\|c^* - c_N\|$
11	8.8010	4.3833	8.7633	8.9078
21	10.7249	5.3535	10.6731	10.8503
41	12.6850	6.3380	12.6283	12.8218
81	14.6646	7.3300	14.6053	14.8075
Order	2	1	2	2

Suppose that we take a differential equation $x' = f(x,u,t)$ and perform an $O(h)$ perturbation of u. Then we expect there to be an $O(h)$ perturbation in x and hence also the cost c. This is illustrated by the simple example of $x' = u$ where u is a constant and we perturb it to the new constant $u+h$. There is a perturbation of x by the function th which is $O(h)$.

The order results are not telling the full story in at least two ways. One is for this specific example where an $O(h)$ perturbation of u is producing an $O(h^2)$ perturbation of x. There is also the more fundamental question of what are the implications of computing the cost to higher accuracy than the control. We address this second point later. To see what is happening with this example, let us look at $u_N - u^*$ for some different values of N. Figures 2 - 3 graph $u_N - u^*$ for $N = 11, 21$ The plots for $41, 81$ are similar in that they have a single spike one grid interval wide at each end.

What we see is that while u_N may be differing from u^* by $O(h)$ in the sup norm, it is actually much closer in terms of say the L^1 norm since the spikes at either end of $u^* - u_N$ are decreasing in both height and base length. In fact, if we remove the endpoint values from u_N, the result, call it u_{EN}, is order 2 (see Tables II and III) and there is an interesting relationship between u_{EN} and the post-computed control u_{PN}.

For more complicated problems, minimizing the Hamiltonian (6) is a numerical optimization problem that may be

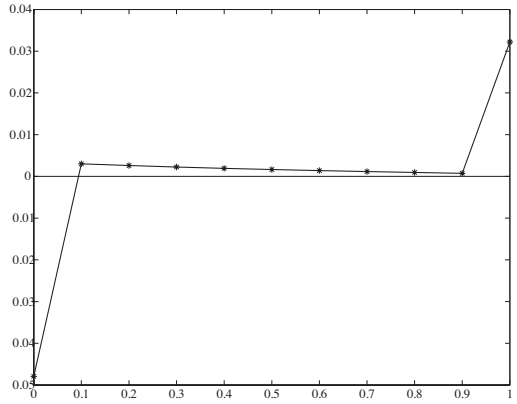

Fig. 2. Graph of $u^* - u_N$ for $N = 11$ using TR.

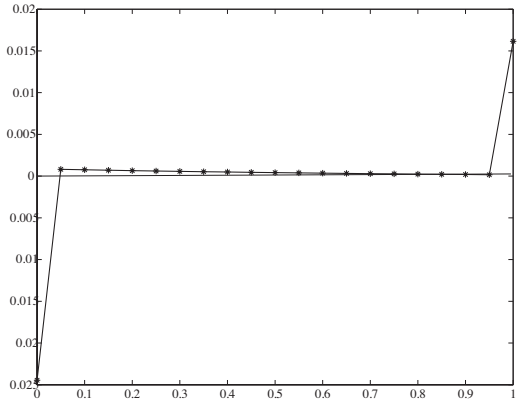

Fig. 3. Graph of $u^* - u_N$ for $N = 21$ using TR.

too time-consuming to be feasible, but in this example there is a simple analytical solution, $u^i_{PN} = -0.5(x^i_N + \lambda^i_N)$. As follows from [20] and our own computational experiments, the λ_N is a better approximation to λ at midpoints than at gridpoints. We have verified computationally for TR that this gives second order at the midpoints but would only be first order at the grid points. This means that the estimates of λ and x are not on the same grid. There are several interpolation techniques that can be used to estimate λ at the mesh points.

It turns out that when we use linear interpolation, the computed values of u_{PN} are almost exactly the same as u_N everywhere but at the endpoints. As shown in Table II u_{EN} and u_{PN} differ at the level of the NLP solver tolerances. When cubic interpolation is used, the resulting u_{PN} is more accurate but still has the same shape as u_{EN}. Figures 4 - 5 show errors in u_{EN}, u_{PN} computed by linear interpolation and u_{PN} computed by cubic interpolation on the same axis. Tables II and III give the L_∞ norm, and the $-\log_2$ of the L_∞ norm of the errors in u_N, u_{EN}, u^{linear}_{PN} and u^{cubic}_{PN}. Table II demonstrates how close u_{EN} and u^{linear}_{PN} are and Table IV confirms that while u_N is first order, u_{EN}, u^{linear}_{PN} and u^{cubic}_{PN} are all second order. The xxx in the the third column of Table II indicates the digits where the two quantities were different.

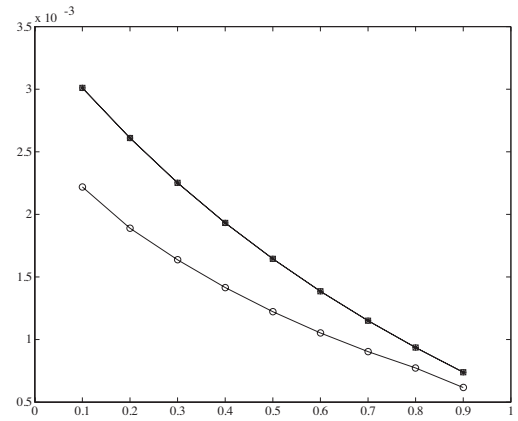

Fig. 4. Graph of $u^* - u_{EN}$ (stars), $u^* - u^{\text{linear}}_{PN}$ (squares) and $u^* - u^{\text{cubic}}_{PN}$ (circles) for $N = 11$ using TR.

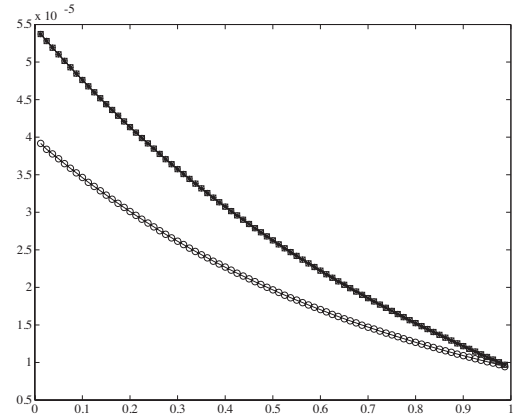

Fig. 5. Graph of $u^* - u_{EN}$ (stars), $u^* - u^{linear}_{PN}$ (squares) and $u^* - u^{cubic}_{PN}$ (circles) for $N = 81$ using TR.

TABLE II

L_∞ NORM ERROR IN POST-PROCESSED CONTROLS USING TR.

N	$\|u^* - u_N\|$	$\|u^* - u_{EN}\|$ or $\|u^* - u^{\text{linear}}_{PN}\|$	$\|u^* - u^{\text{cubic}}_{PN}\|$
11	0.047917966	0.00301066060755	0.00221954389195
21	0.024458336	0.000812xx688249	0.00059471066152
41	0.012361876	0.00021086822385	0.00015397311367
81	0.006215127	0.00005372233xxx	0.00003916937314

TABLE III

$-\log_2$ OF L_∞ ERROR IN POST-PROCESSED CONTROLS USING TR.

N	$\|u^* - u_N\|$	$\|u^* - u_{EN}\|$ or $\|u^* - u^{\text{linear}}_{PN}\|$	$\|u^* - u^{\text{cubic}}_{PN}\|$
11	4.3833	8.3757	8.8155
21	5.3535	10.2661	10.7155
41	6.3380	12.2113	12.6650
81	7.3300	14.1841	14.6399
Order	1	2	2

TABLE IV

$-\log_2$ OF ERROR IN OUTPUTS USING HS.

N	$\|x^* - x_N\|$	$\|u^* - u_N\|$	$\|\lambda^* - \lambda_N\|$
6	17.6312	8.7187	8.8156
11	21.5459	10.6696	10.7175
21	25.4973	12.6455	12.6697
Order	4	2	2

TABLE V

$-\log_2$ OF ERROR IN POST-COMPUTED u USING HS.

N	$\|u^* - u_{PN}^{linear}\|$	$\|u^* - u_{PN}^{cubic}\|$	$\|c^* - c_N\|$
6	8.0237	9.1889	18.7036
11	9.8163	11.6085	22.6947
21	11.7177	13.6156	25.8706
Order	2	2	3–4

B. Using HS

The other discretization commonly used by SOCS is HS. As an ODE integrator HS is fourth order. It satisfies the conditions in Table 1 of [6] for fourth order convergence in state, multiplier and post-computed control. Since the method is fourth order as an integrator, and since the interval $[0, 1]$ is short, we cannot take too fine a mesh or the NLP tolerances will be larger than the ODE error.

Tables IV and V show the $-\log_2$ of the infinity norm of the errors for HS when $N = 6, 11, 21$. We see fourth order convergence in state, second order in control, second order in the multipliers, and third or fourth order in the cost. This appears to contradict Hager's result which states fourth order convergence for multipliers. Since λ_N is order 2, u_{PN} is also only second order, not fourth order as Hager states. In fact, u_{PN}^{linear} is slightly less accurate than u_N because of the error introduced by interpolation, while u_{PN}^{cubic} is slightly better.

Figures 6 and 7 show errors in $u_N, u_{PN}^{\text{linear}}$ and u_{PN}^{cubic} on the same axis. Notice that, unlike when using TR, the functions $u^* - u_N$ vary smoothly and the sup norm correctly captures the effect of the control on the dynamics. Also notice that neither of the post-computed controls matches u_N as in the TR case.

Comparing the results for TR and HS we see that for TR at $N = 11$, the $-\log_2$ of the error in u_{PN} is around 8 while for HS the $-\log_2$ of the error in u_N is around 10. Similarly, for $N = 21$ we see that the $-\log_2$ of the TR post processed control error is 10.26 while for HS the $-\log_2$ of the control error is 12.65. Thus on this example it is the case that both the post processed control estimate from TR and the control estimate from HS are converging with order 2 but the error with HS is smaller.

C. Implications

The question then is how should one interpret the order results either in choosing a method or understanding the results. This requires considering a number of factors including how the answer is going to be used. Suppose that we take a computed control and try to use it. Then in a simulation

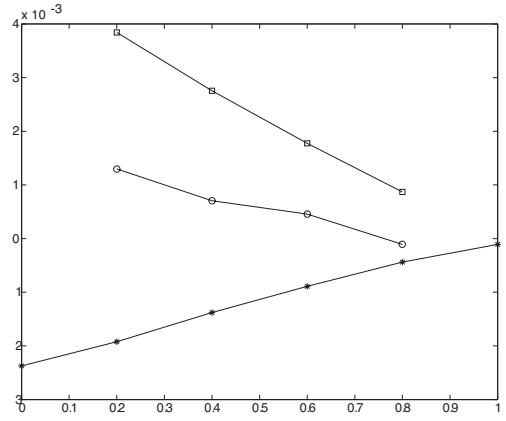

Fig. 6. Graph of $u^* - u_N$ (stars), $u^* - u_{PN}^{\text{linear}}$ (squares) and $u^* - u_{PN}^{\text{cubic}}$ (circles) for $N = 6$ using HS.

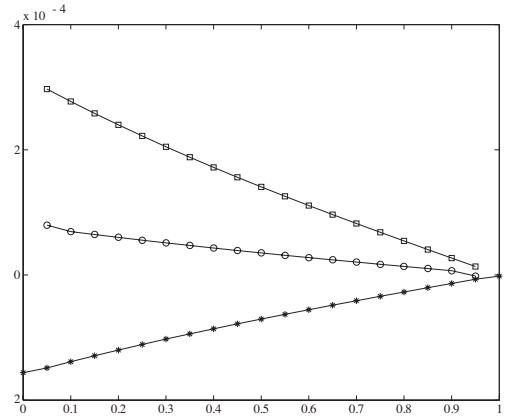

Fig. 7. Graph of $u^* - u_N$ (stars), $u^* - u_{PN}^{\text{linear}}$ (squares) and $u^* - u_{PN}^{\text{cubic}}$ (circles) for $N = 21$ using HS.

using the same step size as used in the mesh selection, we would get the the computed sate trajectory.

But the order results are based on assumptions about the smoothness of the right hand side of the differential equation defining the dynamics. Consider the simple example where we have exact values of a control $u(t) = t$ on a uniform grid of $[0, 1]$ and we have the dynamical system $x' = u$. If we apply u as a piecewise constant function, such as is sometimes done in practice, then we get an $O(h)$ error in $x(1)$. On the other hand, if we apply $u(t)$ as a piecewise linear spline, then we get the x exactly. Similar comments can be made about higher order approximations. Thus we must consider how the control will be implemented. If two methods produce a control on the same grid, and one has higher order, whether that higher order is meaningful can depend very much on how the control is implemented between the grid points. A doctor administering medication during several office visits is a very different problem from that of a cutting machine that can follow higher order splines [21].

A related, but different point, concerns control parameterization. Suppose that we assume that the control is piecewise constant and changes a finite number of times. If

the computational mesh is much finer than the grid where the control changes value, then the order as an ODE integrator may be more important than the theoretical order to which it solves optimal control problems since the function evaluation error that the NLP solver sees is dependent more on the order as an ODE integrator.

These comments presuppose that the important thing is the cost, and the control is just a means to an end. There are scenarios, however, where the control has great importance in its own right. For example, the control might be the shape of an intake to a jet engine or the shape of a part to cut out. In this case high precision may be needed and even essential.

IV. CONCLUSIONS

This paper has examined the order of approximation of the control for the Hermite Simpson and trapezoid discretizations in relation to both the existing theory and what some of the implications of that theory are for practice in the design of general purpose optimal control software.

For the trapezoid method we have seen that it is possible to know the optimal cost and the states and multipliers to $O(h^2)$ yet the values of the control may differ in some places by $O(h)$. In some situations a better solution might be desirable. In others the lower order control would give comparable performance. For example, if the control computed by TR in this example was implemented as a piecewise constant control, with the sampling finer then the computational grid, then we will get second order convergence to the cost.

We have seen that post-processing the control, when practical, may be simply equivalent to removing a few points where the error is significantly larger than on the rest of the interval. In other cases, the multiplier has the same order as the control so post-processing does not improve the control at all. This raises the possibility of modifying the original TR in order to get second order without postprocessing. Hager has described a class of second order methods in [18] but they are not spline based as is preferred in SOCS. These, and other second order methods, will be investigated.

We have also seen that while software can provide estimates of the adjoint variables, relating these different estimates may be delicate. Further investigation is needed of the best way to get adjoint estimates on the computational grid within the compressed HS framework.

V. ACKNOWLEDGMENTS

Research supported in part by NSF Grants DMS-0101802, DMS-020695, DMS-0404842, and ECS-0114095.

REFERENCES

[1] A. Engelsone, S. L. Campbell, and J. T. Betts, "Direct transcription solution of higher index optimal control problems," in *Proc. 2005 IMACS World Congress on Scientific Computation*, Paris, France, 2005.

[2] J. T. Betts and W. P. Huffman, "Sparse Optimal Control Software socs," Mathematics and Engineering Analysis Technical Document MEA-LR-085, Boeing Information and Support Services, Tech. Rep., July 1997.

[3] J. T. Betts, *Practical Methods for Optimal Control using Nonlinear Programming*. Philadelphia: SIAM, 2001.

[4] ——, "Mesh Refinement in Direct Transcription Methods for Optimal Control," *Optimal Control Applications and Methods*, vol. 19, pp. 1–21, 1998.

[5] J. T. Betts, N. Biehn, S. L. Campbell, and W. P. Huffman, "Compensating for Order Variation in Mesh Refinement for Direct Transcription Methods," *Journal of Computational and Applied Mathematics*, vol. 125, pp. 147–158, 2000.

[6] W. Hager, "Runge-kutta methods in optimal control and the transformed adjoint system," *Numer. Math.*, vol. 87, pp. 247–282, 2000.

[7] J. T. Betts, S. L. Campbell, and A. Engelsone, "Direct transcription solution of inequality constrained optimal control problems," in *Proc. 2004 American Control Conference*, Boston, MA, 2004.

[8] J. T. Betts, N. Biehn, and S. L. Campbell, "Convergence of Nonconvergent IRK Discretizations of Optimal Control Problems with State Inequality Constraints," *SIAM Journal on Scientific Computing*, vol. 23, no. 6, pp. 1982–2008, 2002.

[9] S. L. Campbell, N. Biehn, L. Jay, and T. Westbrook, "Some Comments on DAE Theory for IRK Methods and Trajectory Optimization," *Journal of Computational and Applied Mathematics*, vol. 120, no. 1–2, pp. 109–131, Aug. 2000.

[10] J. Bonnans and J. Laurent-Varin, "Computation of order conditions for symplectic partitioned runge-kutta schemes with application to optimal control," INRIA-Rocquencourt, France, Tech. Rep. Report RR5398, 2004.

[11] P. J. Enright and B. A. Conway, "Discrete Approximations to Optimal Trajectories Using Direct Transcription and Nonlinear Programming," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 15, no. 4, pp. 994–1002, July–Aug. 1992.

[12] K. Malanowski, C. Buskens, and H. Maurer, *Convergence of Approximations to Nonlinear Optimal Control Problems*, ser. Mathematical Programming with Data Perturbations, Lecure Notes in Pure and Appl. Math. New York: Dekker, 1998.

[13] A. Schwartz and E. Polak, "Consistent approximations for optimal control problems based on runge-kutta integration," *SIAM Journal of Control and Optimization*, pp. 1235–1269, July 1996.

[14] O. von Stryk, "Numerical Solution of Optimal Control Problems by Direct Collocation," in *Optimal Control*, ser. International Series of Numerical Mathematics, R. Bulirsch, A. Miele, J. Stoer, and K. H. Well, Eds., vol. 111. Basel: Birkhäuser Verlag, 1993, pp. 129–143.

[15] O. von Stryk and R. Bulirsch, "Direct and Indirect Methods for Trajectory Optimization," *Annals of Operations Research*, vol. 37, pp. 357–373, 1992.

[16] W. Hager, "Rates of convergence for discrete approximations to unconstrained control problems," *SIAM J Numer Anal*, vol. 13, pp. 321–338, 1976.

[17] A. L. Dontchev, "Error estimates for a discrete approximation to constrained control problems," *SIAM J Numer Anal*, vol. 18, pp. 500–514, 1981.

[18] A. L. Dontchev, W. Hager, and V. Veliov, "Second-order runge-kutta approximations in control constrained optimal control," *SIAM J Numer Anal.*, vol. 38, pp. 202–226, 2000.

[19] V. Veliov, "On the time discretization of control systems," *SIAM J. Control Optim.*, vol. 35, no. 5, pp. 1479–1486, 1997.

[20] J. T. Betts and W. F. Huffman, "Estimating adjoint variables from a direct transcription solution," Mathematics and Engineering Analysis Technical Document MEA-LR-085, Boeing Information and Support Services, Tech. Rep., April 2003.

[21] J. T. Betts, "Parametric tool path trajectory optimization," Boeing Information and Support Services, The Boeing Company, PO Box 3707, Seattle, WA 98124-2207, Boeing Tech Report SSGTECH-98-006, 1998.