A NEW COMPUTATIONAL METHOD FOR OPTIMAL CONTROL OF A CLASS OF CONSTRAINED SYSTEMS GOVERNED BY PARTIAL DIFFERENTIAL EQUATIONS

Nicolas Petit * **Mark Milam** ** **Richard Murray** **

* *Centre Automatique et Systèmes*
École des Mines de Paris
60, bd St Michel
75272 Paris, France
** *Division of Engineering and Applied Science*
Control and Dynamical Systems
California Institute of Technology
Pasadena,CA 91125, USA

Abstract: A computationally efficient technique for the numerical solution of constrained optimal control problems governed by one-dimensional partial differential equations is considered in this paper. This technique utilizes inversion to map the optimal control problem to a lower dimensional space. Results are presented using the Nonlinear Trajectory Generation software package (NTG) showing that real-time implementation may be possible.

Keywords: Optimal Control, Partial Differential Equations, Inversion, Real-time, Computational methods.

1. INTRODUCTION

In the recent years, optimal control problems with systems governed by partial differential equations subject to control and state constraints have been extensively studied. We refer for instance to (Lions, 1971; Bonnans and Casas, 1995; Bergounioux *et al.*, 1998) for necessary optimality conditions for special cases of elliptic problems and to (Maurer and Mittelmann, 2001) for numerical studies. A typical approach to solve these problems is to discretize both the control and the state and use nonlinear programming to solve the resulting optimization problem. In (Maurer and Mittelmann, 2001), this approach was proposed resulting in a large nonlinear programming prob-

In this paper, we will propose a different methodology. For optimal control of nonlinear ordinary differential equations of the form $\dot{x} = f(x)+g(x)u$, where $\mathbb{R} \ni t \mapsto x \in \mathbb{R}^n$ and $\mathbb{R} \ni t \mapsto u \in \mathbb{R}^m$, we have shown (Milam *et al.*, 2000; Petit *et al.*, 2001) that it is possible and computationally efficient to reduce the dimension of the nonlinear programming problem by using inversion to reduce the number of dynamic constraints, thus eliminating variables, in the problem. Given a particular output, it is generally possible to parameterize a part of the control and a part of the state in terms of this output and its time derivatives. The case of complete parameterization of nonlinear ordinary differential equations is called "flatness" (Fliess *et al.*, 1995; Fliess *et al.*, 1999).

The idea of reducing the dynamic constraints via

ear Trajectory Generation (NTG) software package (Milam *et al.*, 2000). The outputs of the system are approximated by B-splines and nonlinear programming is used to solve for the coefficients of the B-splines. This software can today be considered as an alternative to the well-established collocation software packages developed using methods described in (Hargraves and Paris, 1987; Seywald, 1994), and (von Stryk and Bulirsch, 1992). Other publications (Milam *et al.*, 2002) deal with the real-time implementation of NTG and thus underlines the importance of the computation-time reduction.

In this paper we propose to extend the "inversion" concept to the field of partial differential equations. In this case the outputs are parameterized by tensor-product B-splines instead of B-splines. B-spline tensor products' partial derivatives can be easily computed, combined and substituted to as many components of the states and the control as possible in both the cost functions and the constraints.

The contribution of our current work is to develop theory and a set of corresponding software tools for the real-time solution of constrained optimal control problems for a class of systems governed by partial differential equations. We think that these set of software tools would be useful in the model predictive and process control communities.

In Section 1 we detail our approach. We apply our proposed methodology to an example from the literature in Section 2. The results show that this methodology is efficient and that solutions of optimal control problems for systems governed by partial differential equations may be computed in real-time using our technique.

2. PROBLEM FORMULATION AND PROPOSED METHOD OF SOLUTION

2.1 *Optimal Control Problem*

Notationally, we use $\mathbb{N} = \{1, 2, 3, ...\}$ to represent the natural numbers and $\mathbb{R}$ to represent the reals. Let Ω be an open set in $\mathbb{R}^2$ and $\Gamma = \bar{\Omega} - \Omega$ its boundary. We denote $\Omega \ni (t, x) \mapsto \phi(t, x)$ the state of the system, $\Omega \ni (t, x) \mapsto u(t, x)$ the control, with $n = \dim \phi$, $m = \dim u$. Let ξ represent the first $(n_t + 1)(n_x + 1)$ partial

$$\begin{aligned}\xi \doteq (\phi, &\frac{\partial \phi}{\partial t}, \ldots, \frac{\partial^{n_t} \phi}{\partial t^{n_t}},\\ &\frac{\partial \phi}{\partial x}, \frac{\partial^2 \phi}{\partial t \partial x}, \ldots, \frac{\partial^{n_t+1} \phi}{\partial t^{n_t} \partial x},\\ &\ldots\\ &\frac{\partial^{n_x} \phi}{\partial^{n_x} x}, \frac{\partial^{(n_t-1)n_x+1} \phi}{\partial t \partial^{n_x} x}, \ldots, \frac{\partial^{n_t+n_x} \phi}{\partial t^{n_t} \partial^{n_x} x}).\end{aligned}$$

We consider systems that are governed by partial differential equations of the form

$$f(\xi(t, x)) = Bu(t, x) \tag{1}$$

in Ω, where $B \in \mathbb{R}^{n \times m}$ is a matrix with coefficients in $\mathbb{R}$, $f : \mathbb{R}^{n(n_t+1)(n_x+1)} \to \mathbb{R}^n$ is a nonlinear function.

We desire to find a trajectory of (1) that minimizes the cost functional

$$\min_{(\phi,u)} J(\phi, u) = \int_\Omega L(\xi(t, x), u(t, x))dx\, dt \tag{2}$$

subject to the domain constraints

$$lb_\Omega \leq S_\Omega(\xi, u) \leq ub_\Omega \tag{3}$$

on Ω and the boundary constraints

$$lb_\Gamma \leq S_\Gamma(\xi, u) \leq ub_\Gamma \tag{4}$$

on Γ, where $L : \mathbb{R}^{n(n_t+1)(n_x+1)+m} \to \mathbb{R}$, $S_\Omega : \mathbb{R}^{n(n_t+1)(n_x+1)+m} \to \mathbb{R}^{n_\Omega}$, $S_\Gamma : \mathbb{R}^{n(n_t+1)(n_x+1)+m} \to \mathbb{R}^{n_\Gamma}$ are nonlinear functions, $n_\Omega \in \mathbb{N}$, $n_\Gamma \in \mathbb{N}$.

We tacitly assume that there exists such an optimal control and refer to (Lions, 1971; Bonnans and Casas, 1995; Bergounioux *et al.*, 1998; Maurer and Mittelmann, 2001) for discussions concerning this important issue.

2.2 *Proposed Methodology of Solution*

There are three components to the methodology we propose. The first is to determine a parameterization (output) such that Equation (1) can be mapped to a lower dimensional space (output space). Once this is done the cost in Equation (2) and constraints in Equations (3) and (4) can also be mapped to the output space. The second is to parameterize each component of the output in terms of an appropriate tensor product B-spline surface. Finally, sequential quadratic programming is used to solve for the coefficients of the B-splines that minimize the cost subject to the constraints in output space.

In most cases, it is desirable to find and output $\Omega \ni (t, x) \mapsto z(t, x) \in \mathbb{R}^p$, $p \in \mathbb{N}$ and a mapping ψ of the form

$$z = \psi(\xi, u) \tag{5}$$

such that (ξ, u) (and thus ϕ) can be completely

$$(\xi, u) = \vartheta(\frac{\partial^s z}{\partial t^s}, \frac{\partial^s z}{\partial t^{s-1}\partial x}, \ldots, \frac{\partial^s z}{\partial x^s}, \frac{\partial^{s-1} z}{\partial t^{s-1}}, \ldots, \ldots, z).$$

Once the output z is chosen, we look for the optimum in a particular functional space: we parameterize each of its components in terms of tensor product B-spline basis functions defined over Ω. These tensor products are only one of many possible choices for basis functions. They are chosen for their flexibility and ease of enforcing continuity between patches of surface. A complete treatment of these functions can be found in (de Boor, 1978). A pictorial representation of one component of an output from an example optimization problem is given in Figure 1 for which $\Omega = (-2, 2) \times (-3, 2)$.

Each component z^l, $l = 1, \ldots, p$ of the output z is written in terms of a finite dimensional B-spline surface as

$$z^l(t, x) = \sum_{i=1}^{p_t} \sum_{j=1}^{p_x} B_{i,k_t}(t) B_{j,k_x}(x) C^l_{i,j} \quad (6)$$

$$p_t = l_t(k_t - m_t) + m_t \quad \text{and} \quad (7)$$

$$p_x = l_x(k_x - m_x) + m_x \quad (8)$$

where $\mathbb{R} \ni t \mapsto B_{i,k_t}(t)$ and $\mathbb{R} \ni x \mapsto B_{j,k_x}(x)$ are the B-spline basis functions given by the recursion formula in (de Boor, 1978). In this case we chose $l_t = 5$ and $l_x = 4$ knot intervals in the t and x directions, respectively. The piecewise polynomials in each of the knot intervals will be of order $k_t = 5$ and $k_x = 6$ in the t and x directions, respectively. Smoothness of the piecewise polynomials will be given by the multiplicities $m_t = 3$ and $m_x = 4$ in the t and x directions, respectively. Note that it is also possible to use different parameters k_t, k_x, m_t, m_x for each component of the output. There is a total of $p_t \times p_x = 156$ total coefficients $C^l_{i,j}$ used to define the component z^l of the output in Figure 1.

The breakpoints are a grid ($nbps_t \times nbps_x$) where the boundary and domain constraints will be enforced. There is a similar notion for the integration points. We chose 21 breakpoints in the t direction and 26 breakpoints in the x direction.

After the output has been parameterized in terms of B-spline surfaces, the coefficients $C^l_{i,j}$ of the B-spline basis functions will be found using sequential quadratic programming. This problem is stated as

$$\min_{y \in \mathbb{R}^N_c} F(y) \qquad \text{subject to} \quad lb \le c(y) \le ub \quad (9)$$

$$\text{where } y = (C^1_{1,1}, C^1_{1,2}, \ldots, C^p_{p_t,p_x})$$

$$\text{and } N_c = p_t * p_x * p.$$

$F(y)$ is the discrete approximation in output space

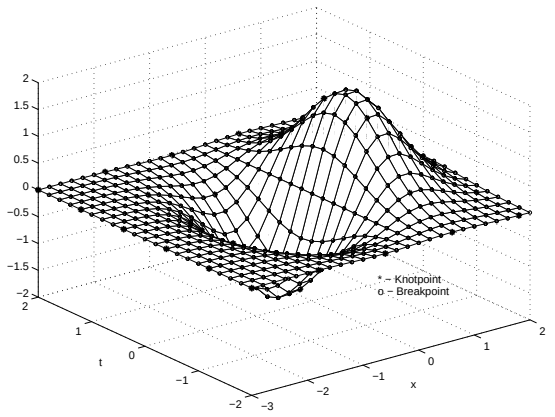

Fig. 1. B-spline Tensor Product Basis Representation

constraints is

$$M = nbps_t * nbps_x * n_\Omega + 2 * (nbps_t + nbps_x) * n_\Gamma.$$

The vector $\mathbb{R}^{N_c} \ni y \mapsto c(y) \in \mathbb{R}^M$ contains the constraints mapped to output space from Equations (3) and (4). We will use NPSOL (Gill *et al.*, 1998) as the sequential quadratic programming to solve this new problem.

3. EXAMPLE

We use here one of the example treated in (Maurer and Mittelmann, 2001). It is related to a simplified *Ginzburg-Landau* equation arising in superconductivity.

As before Ω is an open set in $\mathbb{R}^2$ and Γ its boundary. We consider the following nonlinear partial differential with homogeneous Dirichlet boundary condition ($n = \dim \phi = 1$, $m = \dim u = 1$)

$$-\Delta y - \exp(y) = u \text{ on } \Omega$$
$$y = 0 \text{ on } \Gamma.$$

We look for a control u that minimizes the following cost functional of tracking type

$$F(y, u) = \frac{1}{2} \|y - y_d\|_{L^2(\Omega)} + \frac{\alpha}{2} \|u\|_{L^2(\Omega)}$$

where $y_d(t, x) = 1 + 2\,(t(t-1) + x(x-1))$, while satisfying the constraints

$$y \le .185 \text{ on } \Omega$$
$$1.5 \le u \le 4.5 \text{ on } \Omega.$$

It is clearly possible to parameterize the control using y and its partial derivatives (in this simple case we note $z = y$). Doing so we cast the problem

$$\min \frac{1}{2} \|y - y_d\|_{L^2(\Omega)} + \frac{\alpha}{2} \|-\Delta y(x) - \exp(y)\|_{L^2(\Omega)}$$
$$\text{subject to } y = 0 \text{ on } \Gamma$$
$$y \leq .185 \text{ on } \Omega$$
$$1.5 \leq -\Delta y - \exp(y) \leq 4.5 \text{ on } \Omega.$$

3.1 *Results*

As in (Maurer and Mittelmann, 2001) we choose $\Omega = (0,1) \times (0,1)$, $\alpha = 0.001$. No analytical gradients of the cost and the constraints were provided to NPSOL. Instead, finite difference approximation is used for the gradients. In the future, a function will analytically compute gradients within the NTG software package (it is already the case for ordinary differential equations, not yet for partial differential equations). It is expected to cut down the cpu-time even further (at least by a factor of 2) and increase the accuracy as well.

A set of optimal control and states are plotted in Figure 2. The results of numerical investigations of our approach are detailed in Table 1.

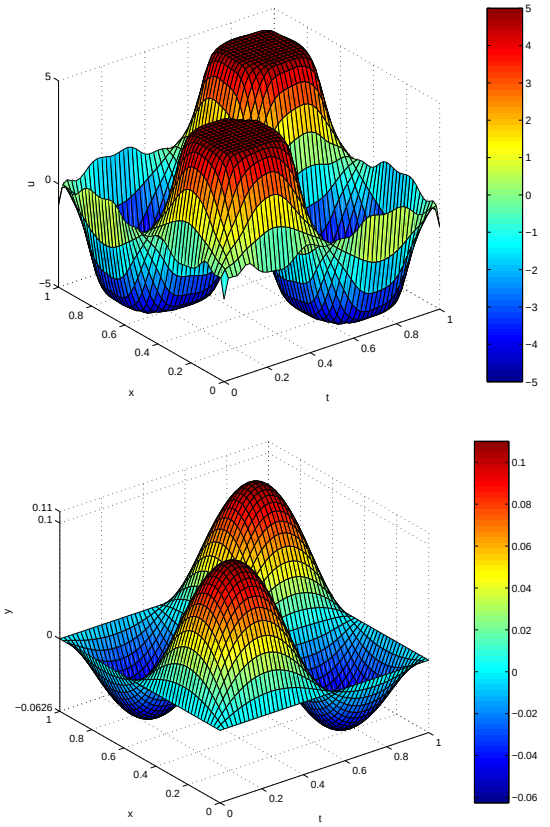

Fig. 2. Example. Optimal control (up) and state (down).

Nomenclature

- N_c: number of coefficients.
- $nbps_t$, $nbps_x$: number of breakpoints in the t and x direction respectively.
- CPU: CPU time (in seconds) on a Pentium-III 733MHZ under Linux Red Hat 6.2 .
- ig: initial guess for coefficients, where 0 means that zeros are used as an initial guess. If the solution from another run with less breakpoints was used for ig then the name of the run is specified, e.g. $ig = (20, 20)$ means the initial guess is the solution to the run with the same degrees and multiplicities but with $(20, 20)$ breakpoints.
- Objective: objective value at the optimum
- k: degree of the polynomials, $k_t = k_x = k$ in this example.
- m: multiplicity of the knotpoints, $m_t = m_x = m$ in this example.
- l: number of intervals, $m_t = m_x = m$ in this example.
- err_u: absolute violation of the constraint on the control.
- err_y: absolute violation of the constraint on the state.

The results in Table 1 show that it is possible to compute fast and with a reasonable accuracy a solution of the optimal control problem. There is also the trade off of a more precise solution for larger computation times. The choice of initial

In the numerical experiments presented above, we evaluate afterwards the cost associated with each set of solution coefficients by an adaptative Lobatto quadrature with accuracy to 8 digits. Thus the costs given here are not the costs evaluated by the nonlinear programming solver but more accurate evaluations of them. Similarly, the absolute violations of the contraints err_u and err_y are evaluated afterwards, using a very large number of breakpoints in both directions.

We reproduce in Table 2 some numerical results from (Maurer and Mittelmann, 2001). It is important to notice that the evaluation of the objective in their approach is different. The quantities y and u are evaluated only at the grid points. The cost is evaluated by the nonlinear programming solver and is not as accurate as the results in Table 1. Asymptotically the two approaches seem to converge to the same value that is not known. In terms of computation time, it is to be noted that the results by Maurer and Mittelmann (in Table 2) were obtained on a 450MHz Pentium-II with a different nonlinear programming solver

N_c	$nbps_t$,$nbps_x$	CPU	ig	Objective	k	m	l	err_u	err_y
64	(10,10)	4	0	0.1112665	5	4	4	5.6e-1	2.5e-3
64	(15,15)	7	0	0.1117173	5	4	4	9.8e-2	0
64	(20,20)	18	0	0.1117267	5	4	4	1.6e-1	0
64	(40,40)	62	0	0.1118501	5	4	4	2.2e-2	0
64	(60,60)	176	0	0.1118300	5	4	4	4.4e-3	0
64	(80,80)	312	0	0.1118253	5	4	4	1.2e-3	0
64	(100,100)	560	0	0.1118291	5	4	4	2.7e-3	0
64	(40,40)	40	(20,20)	0.1118501	5	4	4	2.2e-2	0
64	(80,80)	163	(20,20)	0.1118253	5	4	4	1.2e-3	0
64	(80,80)	334	(40,40)	0.1118254	5	4	4	1.1e-3	0
64	(100,100)	261	(20,20)	0.1118291	5	4	4	2.7e-3	0
144	(10,10)	30	0	0.1104154	6	4	4	4.7e-1	1.0e-2
144	(15,15)	61	0	0.1105299	6	4	4	1.6e-1	2.2e-3
144	(20,20)	90	0	0.1105640	6	4	4	9.2e-2	8.0e-5
144	(40,40)	464	0	0.1106407	6	4	4	1.3e-2	1.0e-5
144	(60,60)	998	0	0.1106456	6	4	4	4.0e-2	6.8e-5
144	(80,80)	1674	0	0.1106465	6	4	4	2.2e-3	1.1e-4
144	(100,100)	2670	0	0.1106481	6	4	4	1.4e-3	3.3e-5
144	(80,80)	924	(20,20)	0.1106465	6	4	4	2.2e-3	1.5e-5
400	(80,80)	15810	0	0.1102986	6	4	8	3.2e-3	0
400	(90,90)	4910	(80,80)	0.1102987	6	4	8	2.2e-3	0

Table 1. Numerical results with the NTG approach.

$gridpoints$	CPU	Objective
2401	131	0.110242
9801	2257	0.110263
39601	42644	0.110269

Table 2. Numerical results by Maurer and Mittelmann.

3.2 *Remarks*

It is important to realize that the methodology we propose produces exact solutions. Once the solution coefficients are determined, the control can be exactly evaluated at any desired point without any refinement of the grid by combinations of exact partial derivatives of the output. Some constraints may be slightly violated in between breakpoints. Asymptotically though, as the number of breakpoints increases the violation experimentally goes to zero.

4. CONCLUSION

The idea in this paper is the use of inversion to eliminate variables from the optimal control problem before using a nonlinear programming solver. To do so, partial derivatives of the output (the parameterizing quantities) are needed. In this context tensor product B-Splines are a useful representation. Numerical results suggest that this methodology is efficient and that fast resolution of such problems can be achieved. Real-time implementation on a reasonably fast process seems close at hand. One can consider for instance a tubular polymerization reactor governed by a one dimensional hyperbolic equation with reaction and heat exhange terms (see for instance (Westerterp *et al.*, 1988)), its time scale is typically 5 minutes which is long enough for receding horizon control

The methodology presented here can be used in various situations including the following problem that we detail to show the generality of our approach. In (Heinkenschloss and Sachs, 1994) the authors expose the following solid-liquid phase transitions control problem. The model consists of two non-linear parabolic equations in Ω subset of $\mathbb{R}^2$.

$$\begin{aligned} T_t + \frac{1}{2}\varphi_t &= kT_{xx} + u \\ \tau\varphi_t &= \xi^2\varphi_{xx} + g(\varphi) + 2T. \end{aligned}$$

In this model the state is $\phi = (\varphi, T) \in \mathbb{R}^2$ (phase function and temperature of the medium, $n = 2$), u is the control ($m = 1$), k, τ, ξ^2 are given parameters, and g is a given nonlinear function. A certain desired phase function φ_d and a temperature u_d are given. An interesting optimal control problem is to minimize the following objective function

$$\begin{aligned} J = &\frac{\alpha}{2}\left\|T - T_d\right\|_{L^2(\Omega)} + \frac{\beta}{2}\left\|\varphi - \varphi_d\right\|_{L^2(\Omega)} \\ &+ \frac{\gamma}{2}\left\|u\right\|_{L^2(\Omega)}. \end{aligned}$$

Inversion can be used in this problem. Both T and u express in terms of the output φ and its partial

$$
\begin{aligned}
T =& \frac{1}{2}\left(\tau\varphi_t - \xi^2\varphi_{xx} - g(\varphi)\right) \doteq h_1(\varphi, \varphi_t, \varphi_{xx}) \\
u =& \frac{1}{2}\left(\tau\varphi_{tt} - \xi^2\varphi_{txx} - \varphi_t\dot{g}(\varphi)\right) + \frac{1}{2}\varphi_t \\
& - \frac{k}{2}\left(\tau\varphi_{txx} - \xi^2\varphi_{xxxx} - \varphi_{xx}\dot{g}(\varphi) - \varphi_x^2\ddot{g}(\varphi)\right) \\
& \doteq h_2(\varphi, \varphi_t, \varphi_{tt}, \varphi_{txx}, \varphi_x, \varphi_{xx}, \varphi_{xxxx}).
\end{aligned}
$$

After substitution in the cost function, the functional to minimize is

$$
\begin{aligned}
& J'(\varphi) = \\
& \quad \frac{\alpha}{2}\left\|h_1(\varphi, \varphi_t, \varphi_{xx}) - T_d\right\|_{L^2(\Omega)} \\
& \quad + \frac{\beta}{2}\left\|\varphi - \varphi_d\right\|_{L^2(\Omega)} \\
& \quad + \frac{\gamma}{2}\left\|h_2(\varphi, \varphi_t, \varphi_{tt}, \varphi_{txx}, \varphi_x, \varphi_{xx}, \varphi_{xxxx})\right\|_{L^2(\Omega)}.
\end{aligned}
$$

We are currently numerically investigating examples of this kind and believe that for such systems the computation time reduction induced by the use of inversion will be very attractive.

5. REFERENCES

Bergounioux, M., M. Haddou, M. Hintermuller and K. Kunisch (1998). A comparision of interior point methods and a Moreau-Yosida based active set strategy for constrained optimal control problems. Preprint MAPMO 98 - 15 Université d'Orléans.

Bonnans, F. and E. Casas (1995). An extension of Pontryagin's principle for state-constrained optimal control of semilinear elliptic equations and variational inequalities. *SIAM J. Control & Opt.* **33**(1), 274–298.

de Boor, C. (1978). *A Practical Guide to Splines*. Springer-Verlag.

Fliess, M., J. Lévine, Ph. Martin and P. Rouchon (1995). Flatness and defect of nonlinear systems: introductory theory and examples. *Int. J. Control* **61**(6), 1327–1361.

Fliess, M., J. Lévine, Ph. Martin and P. Rouchon (1999). A Lie-Bäcklund approach to equivalence and flatness of nonlinear systems. *IEEE AC* **44**, 922–937.

Gill, P., W. Murray, M. Saunders and M. Wright (1998). *User's Guide for NPSOL 5.0: A Fortran Package for Nonlinear Programming*. Systems Optimization Laboratory. Stanford University, Stanford, CA 94305.

Hargraves, C. and S. Paris (1987). Direct trajectory optimization using nonlinear programming and collocation. *AIAA J. Guidance and Control* **10**, 338–342.

Heinkenschloss, M. and E. W. Sachs (1994). *Numerical solution of a constrained control problem for a phase field model*. Vol. 118 of *International Series of Numerical Mathematics*.

Lions, J.-L. (1971). *Optimal control of systems governed by partial differential equations*. Springer-Verlag.

Maurer, H. and H. Mittelmann (2001). Optimization techniques for solving elliptic control problems with control and state constraints. Part 2: Distributed control. *Comp. Optim. Applic.* **18**, 141–160.

Milam, M. B., K. Mushambi and R. M. Murray (2000). A new computational approach to real-time trajectory generation for constrained mechanical systems. In: *IEEE Conference on Decision and Control*.

Milam, M. B., R. Franz and R. M. Murray (2002). Real-time constrained trajectory generation applied to a flight control experiment. In: *Proc. of the IFAC World Congress*.

Petit, N., M. B. Milam and R. M. Murray (2001). Inversion based constrained trajectory optimization. In: 5*th IFAC symposium on nonlinear control systems*.

Seywald, H. (1994). Trajectory optimization based on differential inclusion. *J. Guidance, Control and Dynamics* **17**(3), 480–487.

von Stryk, O. and R. Bulirsch (1992). Direct and indirect methods for trajectory optimization. *Annals of Operations Research* **37**, 357–373.

Westerterp, K. R., W. P. M. Van Swaaij and A. A. C. M. Beenackers (1988). *Chemical Reactor Design and Operation*. Wiley, John & Sons, Inc.