UNDERSTANDING CONCEPTS OF OPTIMIZATION AND OPTIMAL CONTROL WITH WORHP LAB

M. Knauer, C. Büskens

University of Bremen
Bibliothekstraße 1
28359 Bremen

ABSTRACT

The ESA-NLP solver WORHP is already used in several academic and industrial projects in a wide range of applications, as aerospace, automotive or logistics. Currently over 500 users worldwide code their problem formulations using the the standard interfaces to Fortran, C/C++ and MATLAB.

To simplify the formulation of optimisation problems for demonstration and educational purposes WORHP Lab is developed as a graphical user interface (GUI). With a growing set of applied examples and visualisation techniques it shows the capabilities of the underlying solver WORHP and opens access to more involved concepts like parametric sensitivity analysis using WORHP Zen.

Furthermore, WORHP Lab provides the possibility to solve optimal control problems using our transcription method TransWORHP. Different approaches like full discretisation with grid refinement or multiple shooting are compared easily within this tool. Additionally, optimal control problems can be solved on reduced time horizons to illustrate concepts of nonlinear model predictive control.

WORHP Lab was already employed successfully in several industrial workshops as well as for educational purposes with pupils and students. In this talk we illustrate its features with aerospatial examples of optimisation and optimal control problems.

Index Terms— Nonlinear optimization, Optimal control, Sensitivity analysis, Visualisation

1. INTRODUCTION

A large number of problems from diverse applications as aerospace, automotive, logistics or energy management can be led back to nonlinear programming problems (NLP). For the numerical solution of these kind of problems, ESA funded the development of the NLP solver WORHP ("We Optimize Really Huge Problems") at the University of Bremen [1]. By exploiting special sparsity structures, WORHP is able to solve NLP problems with over 10^9 variables and 10^9 constraints on standard hardware.

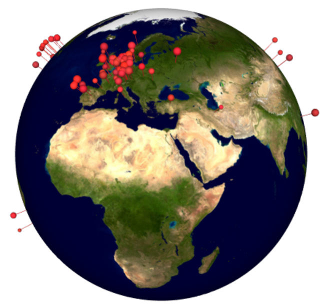

Fig. 1. In 2015, WORHP had over 500 users worldwide.

Though WORHP is announced as the official ESA NLP solver, it has over 500 users worldwide, see Fig. 1. They use WORHP with the common interfaces to C/C++, FORTRAN, MATLAB or AMPL.

With WORHP Lab we present a graphical user interface to WORHP. It acts as a showcase for the features of WORHP and is applied in industrial workshops as well as in education projects at schools and universities.

WORHP Lab offers a simplified interface to WORHP by hiding the organization of optimization variables and the programming interface from the users. Functionalities like interpolation of characteristic maps assist the users to implement their problem formulations rapidly. The iterative solving process of WORHP can easily be analyzed by visualizing the solution, by monitoring optimality conditions or even by clocking the solver stages.

WORHP Lab also supports the module WORHP Zen for a parametric sensitivity analysis, i.e. the effect of small changes in the problem formulation can immediately be understood using the graphical interface.

Another WORHP module, the transcription method TransWORHP can also be accessed via WORHP Lab in order to solve different types of discretized optimal control problems [2].

2. PARAMETRIC OPTIMIZATION

A parametric optimization problem can be formulated depending on a parameter vector $p \in \mathbf{R}^{n_p}$. For a fixed nominal value of p, the vector of variables $z \in \mathbf{R}^n$ has to be optimized, such that an objective function $F : \mathbf{R}^n \times \mathbf{R}^{n_p} \to \mathbf{R}$ is minimized while constraints hold. To follow the notation of WORHP, we differentiate between box constraints for components of the optimization vector, and constraints given by some (nonlinear) function $G : \mathbf{R}^n \times \mathbf{R}^{n_p} \to \mathbf{R}^m$. In both cases, lower limits $l \in \mathbf{R}^n$, $L \in \mathbf{R}^m$ and upper limits $u \in \mathbf{R}^n$, $U \in \mathbf{R}^m$ can be stated.

Putting all this together, WORHP solves nonlinear programming problems of the form:

$$\begin{aligned} \min_{z \in \mathbf{R}^n} \quad & F(z,p) \\ \text{s.}t. \quad & l \le z \le u \\ & L \le G(z,p) \le U \end{aligned} \tag{1}$$

Note, that F and G should be twice continuously differentiable in all arguments.

2.1. WORHP Lab

A parametric optimization problem (1) can be typed straightforward into WORHP Lab. First, the variables have to be declared. Each variable can either be flagged to be used in optimization or to be constant:

Optimization variable. Starting from the initial guess, WORHP will optimize this variable. Box constraints can be provided.

Parameter. This variable will stay on its initial value, but can be used for sensitivity analysis, see Section 3.

Both types of variables might refer to scalar, vector or matrix data, if appropriate dimensions are set.

Second, code for the objective function has to be provided as a fragment of C++ code.

Third, (nonlinear) constraints have to be specified by their upper and lower boundary, their dimension (scalar, vector, matrix) and a name to reference them in the user defined C++ code fragment.

These settings can be stored to an XML file.

After setting this up, WORHP Lab creates a C++ file and compiles it to a dynamic link library (DLL) using compilers from MinGW or Visual Studio to ensure fast model evaluations. On successful compilation the function handles of the DLL are loaded into WORHP Lab. Otherwise the compilation output is prompted to the user.

The console output of the optimization process of WORHP is shown to the user. However, in WORHP Lab he can monitor the changes in the variables, and gets bar charts of the optimality and feasibility conditions per iteration as well as a detailed view of calculation times, see Fig. 2.

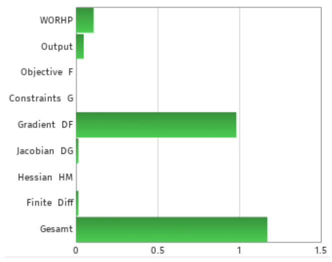

Fig. 2. Bar chart for calculation times in WORHP Lab.

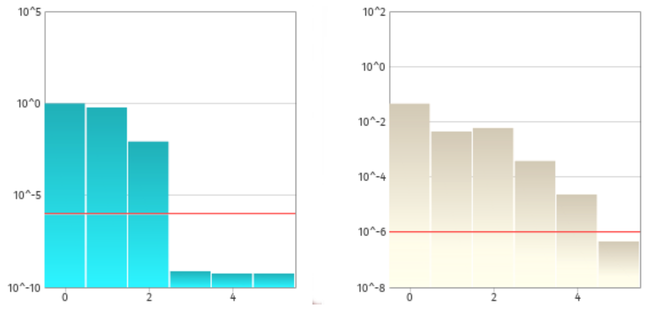

Fig. 3. Iterative development of optimality (left) and feasibility conditions (right) for minimal surface problem.

2.2. Example: Minimal surface

For a given boundary a surface with minimal area is called a minimal surface. For simplicity let's assume that the surface can be expressed as the graph of a function $z : \Omega \to \mathbf{R}$ for $\Omega = [a,b] \times [c,d]$ and is fixed on the boundary $\partial\Omega$.

If the graph of the function z is a minimal surface, then z has to fulfill a partial differential equation, formulated by Lagrange in 1762:

$$(1 + z_x^2)z_{yy} - 2z_x z_y z_{xy} + (1 + z_y^2)z_{xx} = 0 \tag{2}$$

For a numerical solution of this problem, we discretize the axes of Ω by equidistant grids with n_1 resp. n_2 points, and instead of a function z we consider a matrix $z \in \mathbf{R}^{n_1 \times n_2}$.

The minimal surface is the solution of this optimization problem:

$$\begin{aligned} \min_{z \in \mathbf{R}^{n_1 \times n_2}} \quad & \sum_{i=2}^{n_1-1} \sum_{j=2}^{n_2-1} c(z,i,j) \\ \text{s.}t. \quad & z_{1,j} = B_j^1 \\ & z_{n,j} = B_j^2 \\ & z_{i,1} = B_i^3 \\ & z_{i,m} = B_i^4 \end{aligned} \tag{3}$$

where the curvature $c(z,i,j)$ is set to the left hand side of (2), after replacing the derivatives by common finite differences, and fixed values B_i^k are given on the boundary. In this example we use constant and sine functions.

WORHP solves this problem in 5 iterations, see Fig. 3.

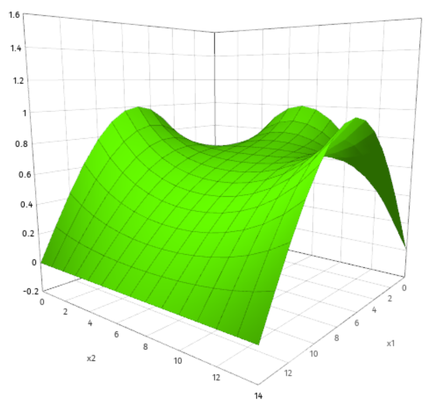

Fig. 4. Solution of minimal surface problem with one constant and three sine shaped borders.

The discretized partial differential equation is fulfilled at all points to (numerical) zero.

As the optimization variables are structured in matrix form, WORHP Lab can display the solution as in Fig. 4. However, user defined plots consisting of points and lines are also supported for more complex data. This example is a simple case of shape optimization. However, it might be extended with a finite element method to arbitrary shapes, in order to optimize shapes of wings or shields.

3. PARAMETRIC SENSITIVITY ANALYSIS

For a given set of nominal parameters $p = p_0$ an NLP solver like WORHP finds an optimal solution $z = z(p_0)$. Without measurable computational cost, the effect of small perturbations in p_0 on the optimal solution can be calculated using the module WORHP Zen in form of sensitivity derivatives $\frac{\mathrm{d}z}{\mathrm{d}p}(p_0)$.

Algorithmic methods and applications for this parametric sensitivity analysis are given in [3]. These results are applied to nonlinear optimization problems in [4].

The sensitivity derivatives can be used to verify the model. They can also be used for a first order approximation of the perturbed solution for a parameter $p \neq p_0$:

$$z(p) \approx z(p_0) + (p - p_0) \cdot \frac{\mathrm{d}z}{\mathrm{d}p}(p_0) \qquad (4)$$

This approximation might violate the constraints of (1). However, as these violations can be interpreted as perturbations of another set of parameters, they can be reduced iteratively.

3.1. WORHP Lab

The sensitivity derivatives calculated by WORHP Zen are directly available in WORHP Lab. The user can generate first order approximations (4) with variable values of $p - p_0$.

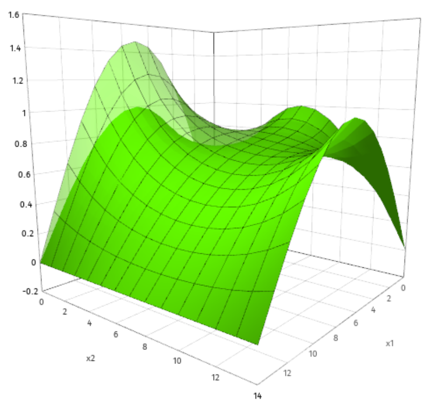

Fig. 5. Approximation of the optimal minimal surface for a changed amplitude of a sine shaped border.

The parameters p can appear in the objective function or the constraints. In Fig. 5 the amplitude of a bounding sine function of the example from Section 2.2 is used as the sensitivity parameter. The assumed effects of changes in the parameter can be visualized instantaneously.

If the sensitivity parameter appears in the objective function, these results can be used to get a linear approximation of the Pareto front.

4. SPARSITY STRUCTURES

WORHP employs a sequential quadratic programming (SQP) method to solve (1). Iteratively quadratic approximations of the problem at the current solution will be generated (and solved). For this, several derivatives have to be calculated:

- the gradient of the objective function,
- the Jacobian of the constraints,
- the Hessian of the Lagrangian (where the objective function is added to constraints weighted by adjoint variables).

The Jacobian, for example is a matrix with n columns and m rows. Within WORHP, the structure and values of the entries of the matrix can be provided by the user. The values can also be calculated with finite differences.

To reduce calculation times in WORHP Lab, where the user should not care about these details, the structure of the Jacobian can be generated from the user's code before starting the optimization.

After setting an optimization variable to the value NaN (not a number) introduced by the IEEE 754 floating-point standard in 1985, the user defined constraints can be checked for this value as any mathematical operation with NaN is infectious.

5. OPTIMAL CONTROL PROBLEMS

While a vector of variables has to be determined for nonlinear programming problems in order to minimize an objective function, for optimal control problems vectors of state and control functions (and additional variables) have to minimize an objective while holding a system dynamics and other constraints:

$$\begin{array}{rrcl}\min\limits_{x,u,p,t_f} & I[x,u] & = & \phi(x(t_f),p) \\ \text{s.t.} & \dot{x}(t) & = & f(x(t),u(t),p,t), \\ & \omega(x(0),x(t_f),p) & = & 0, \\ & g(x(t),u(t),p) & \leq & 0, \quad t \in [0;t_f]. \end{array} \tag{5}$$

In detail, this means: The state of the system and the control are functions over the time interval $[0;t_f]$, where the final time t_f can be fixed or free.

The state of a system at time $t \in [0;t_f]$ may be expressed by the state vector

$$x(t) = (x_1(t),\dots,x_n(t))^T \in \mathbf{R}^n.$$

The behavior of the system has to be controlled using a control vector

$$u(t) = (u_1(t),\dots,u_m(t))^T \in \mathbf{R}^m.$$

Here, $x \in C_p^1([0;t_f],\mathbf{R}^n)$ is a vector of piecewise continuously differentiable functions, and the components of $u \in C_p^0([0;t_f],\mathbf{R}^m)$ are piecewise continuous.

The changes in time of the state of the system can be expressed by a system of differential equations of first order

$$\dot{x}(t) = f(x(t),u(t),p,t), \quad t \in [0;t_f].$$

Here, the continuous function $f : \mathbf{R}^n \times \mathbf{R}^m \times \mathbf{R}^{\widehat{p}} \times [0;t_f] \to \mathbf{R}^n$ is continuously differentiable with respect to x, u, p and t.

Note, that the behaviour of the system can depend on free parameters

$$p = (p_1,\dots,p_{\widehat{p}})^T \in \mathbf{R}^{\widehat{p}},$$

which have to be determined as well.

The initial state $x(0)$ or the final state $x(t_f)$ of the system, or just individual components of these, can be limited with boundary conditions. Generally, these conditions are formulated using a continuously differentiable function $\omega : \mathbf{R}^n \times \mathbf{R}^n \times \mathbf{R}^{\widehat{p}} \to \mathbf{R}^r$:

$$\omega(x(0),x(t_f),p) = 0$$

Additionally, constraints for the control $u(t)$ and the states $x(t)$ can be demanded at any time t using a function

$$g : \mathbf{R}^n \times \mathbf{R}^m \times \mathbf{R}^{\widehat{p}} \to \mathbf{R}^l$$

in the form

$$g(x(t),u(t),p) \leq 0, \quad t \in [0;t_f].$$

In practical cases, if parts of these constraints are just box constraints for some states or controls, they can be handled more easily.

Finally, the objective function $\phi(x(t_f),p)$ (here given in Mayer form) has to be continuously differentiable with respect to all arguments.

For the numerical solution of (5) two classes of methods exist:

- The indirect methods reformulate it to a boundary value problem using necessary conditions of Pontryagin.
- The direct methods reformulate it to a nonlinear optimization problem (1) by discretization of the time.

TransWORHP uses direct methods to solve (5) and replaces the continuous time axes $[0;t_f]$ by N discrete time points

$$0 = t_1 \leq t_2 \leq \dots \leq t_N = t_f$$

The control function $u(t)$ will only be considered at the discrete time points t_i, i.e. $u^i = u(t_i)$, $i = 1,\dots,N$. In between, the control will be interpolated linearly. The state function $x(t)$ can be handled in two ways:

- In single shooting only the initial state $x^1 = x(0)$ can be optimized. Subsequent states are recursively integrated by explicit Runge-Kutta-Schemes. This leads to a small and dense NLP problem.

  Similarly in multiple shooting, the states at some time points can be optimized.
- Using full discretization, the state is used as an optimization variable by the NLP solver at any discrete time point, $x^i = x(t_i)$, $i = 1,\dots,N$. The system dynamics is forwarded to the NLP solver as constraints using different integration schemes, e.g. Euler's method:

  $$0 = x^{i+1} - x^i - (t_{i+1} - t_i) \cdot f(x^i,u^i,p,t_i)$$

  This leads to a large but sparse NLP problem.

The discretized version of (5) is equivalent to an NLP problem (1):

$$\begin{array}{rrcl}\min\limits_{x,u,p,t_f} & \phi(x^N,p) & & \\ \text{s.t.} & x^{i+1} - x^i & = & (t_{i+1}-t_i) f(x^i,u^i,p,t_i), \\ & \omega(x^1,x^N,p) & = & 0, \\ & g(x^i,u^i,p) & \leq & 0, \quad i \in 1,\dots,N. \end{array} \tag{6}$$

5.1. WORHP Lab

An optimal control problem can be edited in WORHP Lab similar to an optimization problem. Each variable can be assigned to one of these types:

State. For each state box constraints as well as initial and final states can be set. A differential equation has to be provided in the next step.

Control. For each control box constraints can be set.

Free parameters. Box constraints can be provided for free parameters.

Zen parameters. These parameters allow a sensitivity analysis.

Data. Characteristic maps can be loaded and used in the model.

Next the codes for the system of differential equations, the objective function, path constraints and boundary conditions have to be given by the user in C++ fragments.

The process time can be set to a fixed value, or the name of a free parameter for free final time.

Full discretization works with Euler's method, the Trapezoidal rule or Hermite-Simpson. The user can also switch to shooting methods providing the indices of the multiple shooting nodes.

5.2. Example: Automated vehicle

The movement of an automated motor vehicle can be modeled by this system of differential equations

$$\begin{aligned}\dot{x} &= v\cos\theta\\ \dot{y} &= v\sin\theta\\ \dot{v} &= u\\ \dot{\theta} &= \tfrac{v}{b}\sin\varphi\\ \dot{\varphi} &= \sigma\end{aligned}$$

Here, x and y describe the position of the vehicle in the x-y plane. v is its current velocity. The orientation of the vehicle θ can be changed by the steering angle φ depending on the parameter $b = 1$ describing the gear ratio.

The vehicle can be controlled using the rate of the steering angle σ and the acceleration u.

For a parking maneuver, these initial and final states are given:

$$\begin{array}{ll} x(0) = 0 & x(t_f) = 5\\ y(0) = 0 & y(t_f) = 2\\ v(0) = 0 & v(t_f) = 0\\ \theta(0) = 0 & \theta(t_f) = -\frac{\pi}{2}\\ \varphi(0) = 0 & \varphi(t_f) = 0\end{array}$$

The objective function can be chosen as:

$$\min t_f + \int_0^{t_f} u^2 + \sigma^2\, dt$$

Note that the integral term can be eliminated by adding a differential equation to the system.

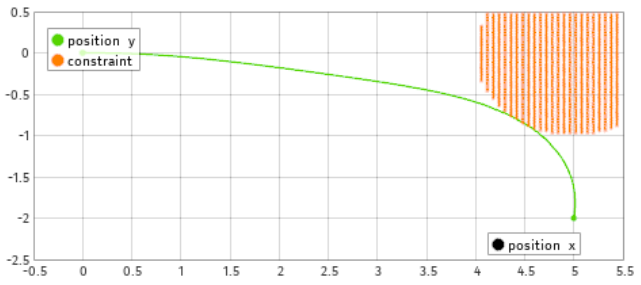

Fig. 6. Optimal trajectory of the automated vehicle respecting an obstacle.

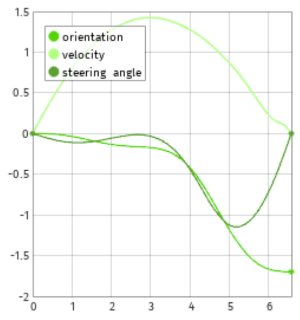

Fig. 7. Additional states of the optimal solution.

Additionally, path constraints can be used, e.g. to define a prohibited circular region around $(x, y) = (5, 0)$:

$$(x-5)^2 + y^2 \geq 1$$

The optimal trajectory of this problem is given in Fig. 6 and 7. A large number of discrete time points is needed in order to avoid crossing the obstacle between to adjacent points.

As optimal control problems are solved in TransWORHP by reducing them to NLP problems, the solution can be analyzed by its parametric sensitivities. In Fig. 8 the direction of the sensitivities in the x-y plane are indicated by thin lines for changes of the gear ratio.

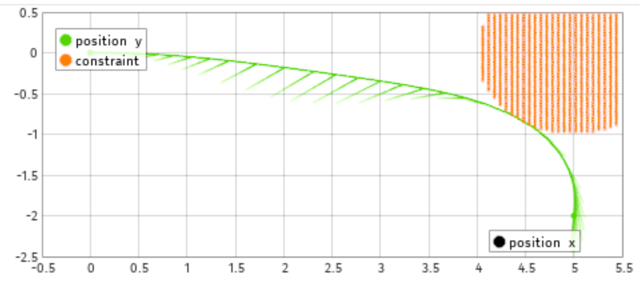

Fig. 8. Sensitivities indicating changes of optimal solution for larger values of gear ratio.

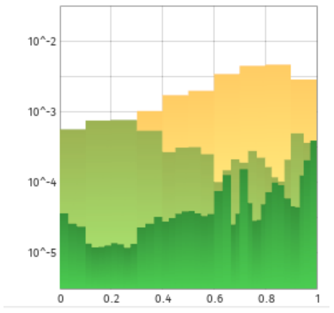

Fig. 9. Reduction of the estimated error on the grid.

5.3. Grid refinement

WORHP Lab uses an equidistant grid for discretizing the optimal control problem. Larger number of grid points promise a more precise result, but increase also the calculation time. However, by calculating a discretization error between adjacent grid points, additional points can be inserted efficiently where they are needed most.

Using an adaptation of the grid refinement method detailed in [5], the user is able to reduce the approximation error of the optimal solution iteratively, see. Fig 9.

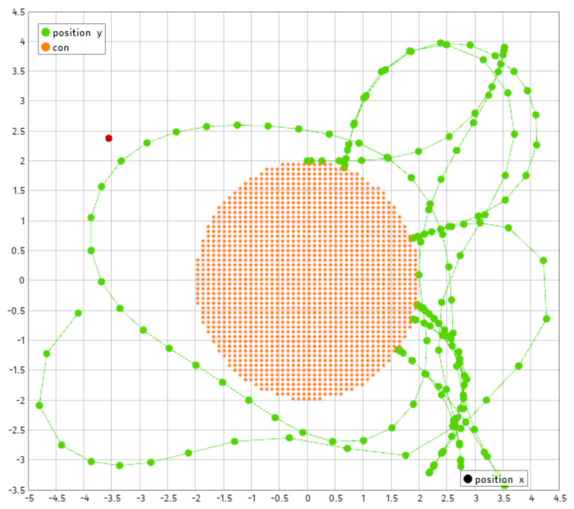

Fig. 10. Following a moving target (red) using an MPC algorithm within WORHP Lab.

5.4. Model predictive control

Tracking problems occur in several industrial applications:

- In robot-based simulators for flight training the simulation has to react to user input in real-time.
- In assistance systems a controller can simplify handling. This might apply to oscillation free movements of container cranes, e.g.

In both examples the task is unknown in advance, and due to constraints too complex for feedback controllers.

In nonlinear model predictive control (MPC), the optimization problem is only solved on a finite prediction horizon. From this solution, only the first part will be accepted. The time frame is shifted accordingly and the optimization continues iteratively.

The consideration of final states which can't be reached within the prediction horizon, has to be shifted into the objective function. In WORHP Lab, an MPC algorithm is prepared, where the weights of the final states can be adjusted.

Fig. 10 shows the iterative steps of the vehicle following an object moving on a lissajous figure. Note, that the given path constraints hold during the optimization.

6. SUMMARY

WORHP Lab was designed to prototype problem formulations and to easily access and visualize solutions of optimization and optimal control problems, and its sensitivity information.

Currently we collect applied examples from different fields of application.

7. REFERENCES

[1] C. Büskens and D. Wassel, *Modeling and Optimization in Space Engineering*, vol. 73 of *Springer Optimization and Its Applications*, chapter The ESA NLP Solver WORHP, Springer, 2013.

[2] M. Knauer and C. Büskens, "From WORHP to TransWORHP," in *Proceedings of the 5th International Conference on Astrodynamics Tools and Techniques*, 2012.

[3] A. V. Fiacco, *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*, vol. 165 of *Mathematics in Science and Engineering*, Academic Press, 1983.

[4] C. Büskens and H. Maurer, *Online Opimization of Large Scale Systems: State of the Art*, chapter Sensitivity Analysis and Real-Time Optimization of Parametric Nonlinear Programming Problems, pp. 3–16, Springer, 2001.

[5] John T. Betts, *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, SIAM, 2010.