How Should Rate Constraints be Implemented in Nonlinear Optimal Control Solvers?

Yuanbo Nie ∗ Eric C. Kerrigan ∗,**

∗ Department of Aeronautics, Imperial College London, UK (e-mail: yuanbo.nie15@imperial.ac.uk).
** Department of Electrical and Electronic Engineering, Imperial College London, UK (e-mail: e.kerrigan@imperial.ac.uk)

Abstract: This paper investigates the problem of implementing rate constraints when solving nonlinear optimal control problems with direct transcription methods. We generalize the approach of directly implementing rate constraints on the discretization mesh to all types of collocation methods (h, p and hp), for both state and input variables. This “on mesh” implementation replaces the additional dynamic equations and nonlinear path constraints in classical implementations with linear equations. Thus, there is no contribution to the Hessian and the contribution to the Jacobian can be precomputed, enabling faster iterations. Through an example, the benefits of the proposed approach are demonstrated, both in terms of obtaining singular arc-free solutions, as well as reductions in computation time of more than 20%.

Keywords: Optimal control, rate constraints, direct transcription, orthogonal collocation

1. INTRODUCTION

With rapid advancements in computing technology, optimization-based control strategies, such as model predictive control (MPC), can be seen in an increasing number of real-time applications. For example, Blackmore (2016) revealed that the recent success in the autonomous landing of the Falcon series rockets by SpaceX was achieved through high-speed onboard convex optimization, with flight code generated by CVXGEN (Mattingley and Boyd, 2012).

For many advanced applications, a linear approximation of the controlled system may not be sufficient, making the use of nonlinear models necessary. In addition, there is an increasing trend to directly formulate the control problem according to formal specifications. Therefore, instead of designing a regulator for stabilization or set-point tracking, minimum-time and minimum-energy costs are directly implemented as the optimization objective. This variant is frequently known as economic model predictive control (EMPC, see Müller (2014)).

Transforming the relatively mature linear MPC framework to nonlinear MPC and EMPC still gives rise to many numerical challenges. One of the key reasons is that the underlying optimization problem can often be formulated in a number of different ways. Under a linear framework, where the control problem can be formulated as a convex quadratic program, most implementations are computationally comparable, thus the most intuitive approaches are often used.

However, under a nonlinear framework with nonlinear programming, different implementations that might seem equivalent can often lead to remarkable differences in computational complexity. Unfortunately, more often than not, the intuitive approaches are the inefficient ones. Implementation of rate constraints on the state and input variables is one of these cases.

In this paper, we will analyze different methods for implementing rate constraints, focusing specifically on the solution quality and computational performance. Sections 2–3 aim at providing a short background on solving optimal control problems (OCPs) with direct transcription methods. Following this, different approaches for implementing rate constraints in the OCP are introduced in Section 4. This is followed by a classical example in Section 5, where the pros and cons of each implementation are demonstrated and analyzed. In Section 6, we provide concluding remarks and some guidelines for implementation.

2. OPTIMAL CONTROL PROBLEMS

Generally speaking, optimization-based control requires the solution of optimal control problems with the objective functional expressed in the general Bolza form:

\[
\min_{x,u,p,t_0,t_f} \Phi(x(t_0),t_0,x(t_f),t_f,p) + \int_{t_0}^{t_f} L(x(t),u(t),t,p)dt
\]

subject to

\[
\begin{align*}
\dot{x}(t) &= f(x(t),u(t),t,p), \forall t \in [t_0,t_f] \\
c(x(t),u(t),t,p) &\leq 0, \forall t \in [t_0,t_f] \\
\phi(x(t_0),t_0,x(t_f),t_f,p) &= 0,
\end{align*}
\]

with \(x(t) \in \mathbb{R}^n\) the state of the system, \(u(t) \in \mathbb{R}^m\) the control input, \(p \in \mathbb{R}^s\) static parameters, \(t_0 \in \mathbb{R}\) and \(t_f \in \mathbb{R}\) the initial and terminal time. \(\Phi\) is the Mayer cost
functional \((\Phi: \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R})\), \(L\) is the Lagrange cost functional \((L: \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R})\), \(f\) is the dynamic constraint \((f: \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^n)\), \(c\) is the path constraint \((c: \mathbb{R}^n \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^m)\) and \(\phi\) is the boundary condition \((\phi: \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \times \mathbb{R} \times \mathbb{R}^n \rightarrow \mathbb{R}^m)\).

In practice, most optimal control problems in this formulation need to be solved with numerical discretization schemes. Indirect methods require analytical expressions from the optimality conditions, with are often difficult to obtain. Consequently its counterpart, direct methods, have become the de facto standard for solving practical optimal control problems.

With the direct method, the OCP problem is first discretized, and then the resulting nonlinear programming problem (NLP) is numerically solved. Afterwards, the OCP solution can be reconstructed from the NLP output. In this process, if the solutions to the dynamics equations and the boundary conditions are solved altogether, the corresponding schemes are often referred as direct collocation methods, or direct transcription methods.

### 3. DIRECT TRANSCRIPTION METHODS

Direct transcription methods can be categorized into fixed-order \(h\) methods (e.g. Euler, Trapezoidal, Hermite-Simpson (HS) and the Runge-Kutta (RK) family)(Betts, 2010), and variable higher-order \(p/hp\) methods (??).

When \(h\) methods are used for direct collocation, a fixed degree polynomial is used in each mesh interval for state approximations. Improving the approximation accuracy is achieved by mesh refinement, thus placing additional mesh points and increasing the number of intervals.

Gaussian quadrature pseudo-spectral methods (\(p\) methods, or orthogonal collocation methods) provide another alternative. The fundamental idea is to use higher-order orthogonal polynomials (Legendre or Chebyshev) for the direct collocation process, with Lagrange interpolating polynomials as basis functions. The main benefit is that for smooth and well-behaved solutions, the method has an exponential convergence rate (Canuto et al., 2012). As a result, better approximation can be achieved with a relatively low degree polynomial and a coarse mesh.

Approximation of discontinuities and rapid changes in the solution with a global \(p\) method can be difficult, and may result in significant reduction in the convergence rate as well as the solution accuracy (due to Runge’s phenomenon). The development of \(hp\)-adaptive methods aims at retaining the best of both sides; increase the degree of the polynomial with \(p\) methods in regions where the solution is expected to be smooth, and only conduct mesh refinement near potential discontinuities or regions of rapid solution changes (Patterson and Rao, 2014).

In this section, we aim to provide a high level overview for direct transcription methods, which is valid for both \(h\) and \(p/hp\) methods. To start with, the states can be approximated as

\[
x^{(k)}(\tau) \approx X^{(k)}(\tau) = \sum_{j=1}^{N^{(k)}} X_j^{(k)} B_j^{(k)}(\tau)
\]

with mesh interval index \(k \in \{1, \ldots, K\}\) and \(N^{(k)}\) denoting the number of collocation points for interval \(k\). In addition, \(B_j^{(k)}(\cdot)\) are basis functions. For classical \(h\) methods, \(\tau\) is defined on the interval \([0, 1]\) representing \([t_0, t_f]\), and \(B_j^{(k)}(\cdot)\) are elementary B-splines of \(v^{th}\) order.

For \(p/hp\) methods, \(B_j^{(k)}(\cdot)\) are Lagrange interpolating polynomials over the normalized time interval \(\tau \in [-1, 1]\).

Consequently, the optimal control problem (1) can be approximated by

\[
\begin{align}
\min_{X, U, t_0, t_f, p} \quad & \Phi(X_1^{(1)}, t_0, X_K^{(K)}, t_f, p) \\
& + \sum_{k=1}^{N^{(h)}} u_i^{(k)} L(X_i^{(k)}, U_i^{(k)}, \tau_i^{(k)}, t_0, t_f, p) \\
\text{subject to} & \\
& \sum_{j=1}^{N^{(h)}} A_{ij}^{(k)} X_j^{(k)} + D_i^{(k)} f(X_i^{(k)}, U_i^{(k)}, \tau_i^{(k)}, t_0, t_f, p) = 0 \\
& c(X_i^{(k)}, U_i^{(k)}, \tau_i^{(k)}, t_0, t_f, p) \leq 0 \\
& \phi(X_i^{(1)}, t_0, X_K^{(K)}, t_f, p) = 0
\end{align}
\]

with

- \(u_i^{(k)}\) the quadrature weights for the respective discretization method chosen.
- \(\tau_i^{(k)}\) are the collocation points in mesh interval \(k\).
- \(U_i^{(k)}\) the approximated control inputs.
- \(\mathcal{A}\) the numerical differentiation matrix, with dependency on the basis function: \(\mathcal{A} = \mathcal{A}(B_j^{(k)})\).
- \(\mathcal{D}\) a constant matrix.

The discretized problem can then be solved with off-the-shelf NLP solvers, such as interior point solver IPOPT (Wächter and Biegler, 2006) or sequential quadratic programming solver SNOPT (Gill et al., 2005).

### 4. ALGEBRAIC VARIABLE RATE CONSTRAINTS

Consider a nonlinear system as described by (1b) with \(x(t)\) the states and \(u(t)\) the inputs. In many problems, constraints of the form

\[
\dot{u}_L \leq \frac{du}{dt}(t) \leq \dot{u}_U \quad (4)
\]

\[
\dot{x}_L \leq \frac{dx}{dt}(t) \leq \dot{x}_U \quad (5)
\]

may need to be implemented to account for actuator limits or fulfilling ride comfort requirements, for example.

A common approach is to introduce \(u\) as an additional state variable, and \(v\) as the new input with a simple bound through the dynamic equation

\[
\dot{u}(t) = v(t) \text{ with } u_L \leq v(t) \leq u_U. \quad (6)
\]

For rate constraints on the state, additional path constraints are needed:

\[
\dot{x}_L \leq f(x, u, t, p) \leq \dot{x}_U. \quad (7)
\]

Unfortunately this intuitive implementation exhibits many shortcomings, as explained by Betts (2010). Not only is the number of state variables and constraint equations
increased, resulting in a larger NLP, but the index of the DAE system of the transcribed problem may also grow, leading to a problem that is often more difficult to solve numerically.

Even if the solver is successful, when \((6)\) is used, singular ares may occur as a result of the new control \(\nu\) appearing linearly in the dynamic equations. Furthermore, for the compressed version of Hermite-Simpson collocation, as well as similar schemes of higher order, ensuring state rate constraint satisfaction with \((7)\) at collocation points is not sufficient for guaranteeing constraint compliance inside the intervals. Consequently, additional mesh refinement criteria may need to be specified to bring the constraint violation to an acceptable level.

To mitigate the above mentioned shortcomings, Betts (2010) proposed a method to impose algebraic rate constraints for input variables directly on the discretization grid, for trapezoidal and Hermite-Simpson discretization schemes. We generalize this approach for all collocation methods \((h, p \text{ and } hp \text{ type})\), as well as for state variables.

Since the treatment for state variables \(z\) and input variables \(u\) are similar, for simplicity we will use \(z\) to represent the variable on which the rate constraints are imposed. If \(Z_i\) represents the discretized version of \(z\) at time instance \(i\), then the numerical differentiation of \(z\) at that grid point \((Z_i')\) can be calculated using \(s\)-point finite difference approximations, with \(s\) the number of data points in the interval (including endpoints). See Table 1 for the formulations of some of the most commonly used discretization methods, with \(\Delta t = t_{i+1} - t_i\) and \(\Delta t = t_f - t_0\).

Note that for \(p/hp\) methods, the numerical differentiation for all grid points on the polynomial \((i = 1, \ldots, N(k))\) are obtained altogether. It is also worth mentioning that if Legendre-Gauss-Radau (LGR) collocation is used, the end-point value for the control \((U_{N+1})\) may need to be approximated.

It is straightforward to implement the rate constraints as linear constraints

\[
\begin{align*}
0 & \leq Z'_i - \dot{z}_L \quad \text{(8a)} \\
0 & \geq Z'_i - \dot{z}_U \quad \text{(8b)}
\end{align*}
\]

for all possible values of \(i\). This approach will be referred to as the on mesh implementation in later sections.

Specifically, for Hermite-Simpson discretization, with the control \(u\) discretized as a quadratic function of time within each interval, the rate of change w.r.t. time \((\dot{u})\) is linear, thus extreme values only occur at the end-points of each interval \((U_i \text{ and } U_{i+1})\). In this special case only, the rate constraints relating to the middle points \((U'_{i+1/2})\) can be neglected. Another situation at which the number of constraints equations may be reduced is when \(\ddot{z}_L = \ddot{z}_U\), with equality constraints used instead.

5. EXAMPLE: AIRCRAFT GO-AROUND IN THE PRESENCE OF WINDSHEAR

Based on the developments by Miele et al. (1988) and Bulirsch et al. (1991a,b), Betts (2010) presented a problem where the aircraft needs to stay as high above the ground as possible after encountering a severe windshear during landing. The simplified dynamics of the aircraft can be described by

\[
\begin{align*}
\dot{d}(t) = & v(t) \cos(\gamma(t)) + w_d(d(t)) \quad \text{(9a)} \\
\dot{h}(t) = & v(t) \sin(\gamma(t)) + w_h(d(t), h(t)) \quad \text{(9b)} \\
\dot{v}(t) = & \frac{1}{m} \left[T(v) \cos(\alpha(t) + \delta) + D(v, \alpha)\right] - g \sin(\gamma(t)) - w_d(d(t), d(t)) \cos(\gamma(t)) \sin(\gamma(t)) - w_h(d(t), h(t), \dot{h}(t)) \sin(\gamma(t)) \quad \text{(9c)} \\
\dot{\gamma}(t) = & \frac{1}{mv(t)} \left[T(v) \sin(\alpha(t) + \delta) + L(v, \alpha)\right] - g \cos(\gamma(t)) + \frac{1}{v(t)} w_d(d(t), \dot{d}(t)) \sin(\gamma(t)) - \frac{1}{v(t)} w_h(d(t), h(t), \dot{h}(t)) \cos(\gamma(t)) \quad \text{(9d)}
\end{align*}
\]

with \(d\) the horizontal distance, \(h\) the altitude, \(v\) the true airspeed, \(\gamma\) the flight path angle, and \(\alpha\) the angle of attack. Polynomial models are used for the maximum thrust \(T_{\text{max}}\), lift coefficient \(C_L\) and drag coefficient \(C_D\), to model the thrust, lift \(L\) and drag \(D\). A simplified windshear model is used with wind speed contributions represented by the horizontal component \(w_d\) and vertical component \(w_h\), respectively.

Details about the aerodynamic modeling, as well as parameter values of all relevant variables, are available in the references above, thus will not be reproduced here. The following simple bounds on some of the state variables

\[
\begin{align*}
0 \leq d & \leq 10000 \text{ [ft]} & 0 \leq h & \leq 1000 \text{ [ft]} \\
0 \leq v & \leq \infty \text{ [ft/s]} & -\infty \leq \gamma & \leq \infty \text{ [deg]} \\
-17 \leq \alpha & \leq 17 \text{ [deg]} & -3 \leq \dot{\alpha} & \leq 3 \text{ [deg/s]}
\end{align*}
\]

are imposed together with the boundary conditions

\[
\begin{align*}
d(0) = & 0 \text{ [ft]} & h(0) = & 600 \text{ [ft]} & v(0) = & 239.7 \text{ [ft/s]} \\
\gamma(0) = & -2.25 \text{ [deg]} & \alpha(0) = & 7.35 \text{ [deg]} \\
d(t_f) = & \text{free} & h(t_f) = & \text{free} & v(t_f) = & \text{free} \\
\gamma(t_f) = & 7.43 \text{ [deg]} & \alpha(t_f) = & \text{free}
\end{align*}
\]

To avoid discontinuities and to assist convergence, an additional optimization parameter \(h_{\text{min}}\) is introduced to represent the minimum altitude. After introducing a new path constraint \(h(t) \geq h_{\text{min}}\), the expression of the optimization cost functional simply becomes \(\Phi := -h_{\text{min}}\).

Figure 1 illustrates the solution to the problem using Hermite-Simpson discretization. All figures presented in this paper are the outcome of a mesh refinement scheme that minimizes the maximum absolute discretization error,
Fig. 1. Solution to the aircraft go-around in the windshear problem, with input rate constraints

as specified in Table 2. It is important to note that, although different implementations of rate constraints can influence the computational performance, the singular arc behaviour will not have noticeable effects on the solution to the state variables. Thus the solution in Figure 1 should be obtainable regardless of the discretization method and the rate constraint implementation.

5.1 Implementation of Rate Constraints for Input Variables

The last set of constraints in (10) applies directly on the rate of change for the control input \( \alpha \). Using the classical approach, \( \alpha \) can be treated as additional state variable, and \( \nu \) introduced as the new control input with the dynamics relationship

\[
\dot{\alpha}(t) = \nu(t)
\]  

Thus the rate constraints for \( \alpha \) can be implemented as simple bounds on \( \nu \): 

\[-3 \leq \nu(t) \leq 3 \text{ [deg/s]}\]  

As mentioned earlier, due to the fact that the original control input \( \alpha \) appears nonlinearly in the system, whereas the new input \( \nu \) appears linearly, singular arc behaviour can occur, which is evidently shown in Figure 2(a), with large fluctuations in the solution.

When the rate constraints are directly implemented on the discretization mesh instead (Figure 2(b)), the optimal control input trajectory can be obtained with little ambiguity. Comparing the solutions from the two implementations, it is interesting to observe that, although the integrated values (i.e. angle of attack) along the singular arc solution at the collocation points are generally the same as the on mesh approach, the interpolated trajectory is actually distorted by the fluctuations of its rate values.

With the LGR orthogonal collocation method (Figure 3), relative improvements are minor, but can still be observed. Since the end-point control is only approximated, the errors have distortion effects on all previous points of the polynomial (Figure 3(b)). On the other hand, thanks to this extra level of continuity imposed by higher order polynomials, the problem of singular arc behaviour is far less pronounced with the classical \( \dot{\alpha} = \nu \) implementation (Figure 3(a)), when comparing to the same implementation with \( h \) type discretization methods.

Table 2. Mesh refinement criteria

| d [ft] | h [ft] | v [ft/s] | \( \gamma \) [deg] | \( \alpha \) [deg] |
|--------|--------|---------|-----------------|----------------|
| Max. Abs. Dis. Error | 1 | 0.5 | 0.1 | 0.5 |

Table 3 compares the computation performance for different rate constraint implementations with both \( h \) and \( hp \) discretization methods. The results presented in Table 3 and 4 were all obtained on an Intel Core i7-4770 computer with 8 GB of RAM, running 64-bit Windows 10 with...
Matlab 2017a, and IPOPT version 3.12 compiled with the sparse linear solver MA57 (Duff, 2004). The computation time are the averages of 50 independent runs. An equally sized grid of 40 nodes is used for the $h$ method, and $hp$ LGR method results are obtained with 5 mesh intervals, each with a polynomial order of 8.

It can be seen that, in both cases, the NLP converges with roughly the same number of iterations and a similar computation time. This can be explained by the fact that the number of additional linear dynamics constraints required for the classical approach is approximately equal to the number of additional linear constraints needed for the on mesh implementation.

Since the linear constraints have no contribution to the Hessian and the contributions to the Jacobian are constants, the on mesh implementation explicitly exploited this property and precomputed the contributions, allowing slightly lower computational cost per iteration.

It can be seen that, in both cases, the NLP converges with roughly the same number of iterations and a similar computation time. This can be explained by the fact that the number of additional linear dynamics constraints required for the classical approach is approximately equal to the number of additional linear constraints needed for the on mesh implementation.

Since the linear constraints have no contribution to the Hessian and the contributions to the Jacobian are constants, the on mesh implementation explicitly exploited this property and precomputed the contributions, allowing slightly lower computational cost per iteration.

5.2 Implementation of Rate Constraints for State Variables

We will additionally impose a rate constraint for the velocity state

$$-5 \leq \dot{v} \leq 5 \ [\text{ft/s}^2] \quad (13)$$

Figure 4 illustrates the solution to this new formulation, with the minimum altitude achievable being slightly lower.

One intuitive implementation inherited from the linear MPC framework is to impose additional path constraints based on the dynamics equation (9b). With regards to solutions obtained, the proposed on mesh approach does not yield obvious differences compared to the classical method (Figure 5).

However, when looking at Table 4, it is obvious that the two methods are not computationally comparable. Path constraints based on (9b) are a nonlinear relationship between different state variables at the same time instance, whereas rate constraints with (8) are linear equations relating the same state variable at different sampled times. As explained before, linear constraints have no contribution to the Hessian and the contribution to the Jacobian can be precomputed, enabling faster iterations.

As a result, although the increase in the number of NLP constraints due to (13) is higher for the on mesh implementation, the resulting (larger) NLP problems are actually much easier to solve. Consequently, regardless of the discretization method, the computation time recorded for the on mesh implementations are still significantly (more than 20%) lower than the method of adding path constraints. Because of the narrow scope of this study, it is still difficult to exactly determine the scale of computational benefits for the on mesh implementation.

### Table 4. Comparison of computational performance, with state and input rate constraints

| Discretization Method | Hermite | Legendre Gauss-Radau (hp) |
|-----------------------|---------|--------------------------|
| **Rate Const.**       |         |                          |
| add path             |         | add path                 |
| on mesh              |         | constraints on mesh      |
|                      |         |                          |
| **NLP Variables**    | 469     | 240                       |
|                      | 391     | 200                       |
| **NLP Constraints**  | 553     | 280                       |
|                      | 785     | 360                       |
| **NLP Iterations**   | 22      | 19                        |
|                      | 20      | 24                        |
| **Total NLP Comp. Time [s]** | 1.4764 | 1.8056                     |
|                      | 0.9079  | 1.3951                     |
Fig. 5. Control input for the solution to the aircraft go-around in the windshear problem, with different implementations for state and input rate constraints

6. CONCLUSIONS

Through a classical example problem, we demonstrated that mathematically equivalent formulations for rate constraints on state and input variables may not result in the same computational complexity when it comes to numerical implementation. For all other collocation methods tested, and for both state and input variables, the proposed approach to directly implement rate constraints on the discretization mesh appears to result in better (singular arc-free) solution quality and/or computational efficiency, making this alternative implementation attractive for nonlinear optimization based control problems.

REFERENCES

Betts, J. (2010). *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming: Second Edition*. Advances in Design and Control. Society for Industrial and Applied Mathematics.

Blackmore, L. (2016). Autonomous precision landing of space rockets. *Winter Bridge on Frontiers of Engineering*, 4(46).

Bulirsch, R., Montrone, F., and Pesch, H. (1991a). Abort landing in the presence of windshear as a minimax optimal control problem, part 1: Necessary conditions. *Journal of Optimization Theory and Applications*, 70(1), 1–23.

Bulirsch, R., Montrone, F., and Pesch, H. (1991b). Abort landing in the presence of windshear as a minimax optimal control problem, part 2: Multiple shooting and homotopy. *Journal of Optimization Theory and Applications*, 70(2), 223–254.

Canuto, C., Hussaini, M.Y., Quarteroni, A.M., Thomas Jr, A., et al. (2012). *Spectral methods in fluid dynamics*. Springer Science & Business Media.

Duff, I.S. (2004). MA57—a code for the solution of sparse symmetric definite and indefinite systems. *ACM Transactions on Mathematical Software (TOMS)*, 30(2), 118–144.

Fahroo, F. and Ross, I.M. (2008). Advances in pseudospectral methods for optimal control. In *AIAA guidance, navigation and control conference and exhibit*, 7309.

Gill, P.E., Murray, W., and Saunders, M.A. (2005). SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM review*, 47(1), 99–131.

Liu, F., Hager, W.W., and Rao, A.V. (2014). An hp mesh refinement method for optimal control using discontinuity detection and mesh size reduction. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, 5868–5873.

Mattingley, J. and Boyd, S. (2012). Cvxgen: A code generator for embedded convex optimization. *Optimization and Engineering*, 13(1), 1–27.

Miele, A., Wang, T., Wang, H., and Melvin, W. (1988). Optimal penetration landing trajectories in the presence of windshear. *Journal of Optimization Theory and Applications*, 57(1), 1–40.

Müller, M. (2014). *Distributed and economic model predictive control: beyond setpoint stabilization*. Logos Verlag Berlin.

Patterson, M. and Rao, A. (2014). GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming. *ACM Transactions on Mathematical Software*, 41(1), 1.

Wächter, A. and Biegler, L.T. (2006). On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1), 25–57.