Fast Recursive SQP Methods for Large-Scale Optimal Control Problems

Gutachter: Prof. Dr. Hans Georg Bock
Prof. Dr. Gerhard Reinelt

Abstract

Direct boundary value problem methods in combination with SQP iteration have proved to be very successful in solving nonlinear optimal control problems. Such methods use parameterized control functions, discretize the state differential equations by, e.g., multiple shooting or collocation, and treat the discretized boundary value problem as an equality constraint in a large, nonlinear, constrained optimization problem. In real-life applications several thousand variables may appear in the NLP. Solution by standard techniques is therefore impractical. This dissertation develops a general concept for a class of structured direct SQP methods based on a decoupling strategy. A careful choice of the discretization reveals an inherent multistage block structure of the QP subproblems. We present a recursive solution algorithm for the associated KKT systems which makes full use of this sparse structure, and propose a structure-preserving primal-dual interior point method for treating the generally large number of path inequality constraints. Extensive numerical tests including a comparison with other sparse solvers document the high efficiency of the recursive algorithm.

Contents

List of Figures

List of Tables

Chapter 0

Introduction

Historical overview

The history of optimal control problems dates back to the end of the seventeenth century, having its origin in the calculus of variations. For instance, the famous brachistochrone problem which was proposed by the swiss mathematician Johann Bernoulli in 1696, may be formulated as an optimal control problem. A specific theory of optimal control, however, has only been developed in the fifties and sixties of this century when real-life problems had to be solved in engineering and science; some classical works include [13, 70, 63, 25]. These textbooks and other early works study the questions of existence and uniqueness, controllability and attainability, and apply variational principles to derive necessary and sufficient optimality conditions for many problem classes. Extremal trajectories of a control problem are thus characterized as solutions of an associated boundary value problem. Numerical methods that rely on the solution of such boundary value problems are called *indirect* methods.

Due to its inherent instabilities the indirect approach was limited to relatively simple problems as long as only single shooting methods were available. Bulirsch [26] has achieved a major breakthrough when multiple shooting has been developed in the sixties [67, 53, 68]. His code OPTSOL has been successfully used to solve several hard application problems involving jumps and switches in the two-point boundary value problem. Later on Deuflhard [30] has improved the Newton method of this approach, and Bock [15] has introduced interior point conditions to stabilize the treatment of jumps and switches. At the same time collocation methods have been developed [73, 29, 93]; a general purpose code and major theoretical contributions are due to Ascher and Russell [7, 8]. Recent extensions to more general problem classes, especially problems associated with differential-algebraic equations, can be found in [78, 10].

Optimality conditions for finite-dimensional constrained optimization (or nonlinear programming) have already been obtained in the forties and fifties by Karush [52], and by Kuhn and Tucker [60]. A variety of numerical methods have been developed in the sequel; among the most recent and most important ones is the technique of sequential quadratic programming (SQP). This method has first been proposed by Wilson [94] and has further been developed by Han, Powell, and others [45, 46, 71, 72]; a good overview and more references can be found in the textbook by Gill, Murray, and Wright [43]. Numerical methods that employ nonlinear programming algorithms to solve discretized optimal control problems are known as *direct* methods.

The direct methods used in the sixties and seventies were typically gradient-type algorithms based on single shooting, see, e.g., Breakwell, Speyer, and Bryson [24] or Miele [65]. The breakthrough in solving real-life problems by direct methods has only been achieved in the eighties, largely due to the boundary value problem approach. This approach is characterized by a multiple shooting or collocation discretization of the multipoint BVP, which is treated as an equality *side condition* in the optimization problem. The BVP approach has been introduced by Bock [16, 17] in the context of parameter estimation for ODE. His code PARFIT combines multiple shooting with a generalized Gauß-Newton iteration for the constrained nonlinear least-squares cost function; a similar collocation code, COLFIT, has been developed by Bär [12]. The first direct algorithm for optimal control problems is due to Bock and Plitt [69, 20]. Their code MUSCOD is based on multiple shooting in connection with a structured SQP method and high-rank block updates of the Hessian; it has been successfully applied, e.g., to trajectory optimization problems in robotics [57, 56, 84, 78, 79]. Another multiple shooting code based on the same theory, PROMIS, has later been developed by Jänsch, Schnepper, and Well [50]; some direct collocation methods are described by Hargraves and Paris [47], Betts and Huffman [14], and von Stryk [92].

Classical applications of optimal control are typically found in the aerospace field, such as planning interplanetary missions of space probes or bringing astronauts into orbit and safely back to earth. These problems require very accurate solutions, but they involve rather small systems with highly exact models, and there is plenty of time for off-line optimization. The indirect approach is ideally suited for their solution.

Nowadays optimal control has become an important means to improve efficiency in technical and industrial applications; it is used, for instance, to speed up robot operation on assembly lines, to give higher yields in chemical production processes, or to reduce energy consumption in railway or subway systems, thus saving natural and economic resources and giving an advantage in competition. Such applications require moderately accurate solutions, but there are usually available only approximate models, and there may be little time for optimization. Direct methods are well-suited under these conditions.

Typical direct boundary value problem methods involve four levels and two iterations; here we describe them for the SQP approach. On the first level lies a mathematical formulation of the original problem in an infinite-dimensional function space. Parameterization of the control function and discretization of the state function leads to the second level, where a large, sparse, constrained, nonlinear optimization problem (NLP) is to be solved. Iterative treatment of this NLP by an SQP method generates a sequence of constrained quadratic problems (QP) on the third level. Finally, inequality constraints in the QP are treated by an active set strategy or interior point method. Both approaches require a secondary iteration on the optimality conditions of the QP. The resulting linear indefinite KKT systems constitute the fourth, lowest level.

Contributions

Non-sparse direct SQP methods for optimal control reach their limit in large-scale applications, i.e., when the number of differential equations, n, or the number of discretization stages, m, is very large. Several thousand variables and constraints may then appear in the discrete optimization problem, and QP solution becomes a bottleneck. More precisely, a KKT matrix factorization costs $\mathcal{O}(m^3n^3)$ operations while the numbers of variables and of nonzero matrix elements grow only as $\mathcal{O}(mn)$ and $\mathcal{O}(mn^2)$ respectively. Hence exploitation of the sparseness is crucial in large systems. However, general purpose sparse solvers have several disadvantages. The matrix structure must be analyzed first, the amount of fill-in is unknown a priori, and a failure of the factorization does not give much information where to find the defect in the original problem. In order to overcome these difficulties we will develop a structured approach which is specifically designed for optimal control problems. In particular, a KKT matrix factorization will cost only $\mathcal{O}(m)$ operations—or more precisely $\mathcal{O}(mn^3)$; the factor n^3 cannot be avoided.

Our structured approach, which has already been outlined in [82], is based on a general strategy for the numerical treatment of nonlinear problems. When the discrete problem is set up, we do *not* attempt to find a compact formulation with as few variables as possible. Instead, our principal goal is a reduction of nonlinear coupling in the problem formulation. Usually this means that the problem becomes blown up by introducing additional variables and conditions, but along with this it receives a clearer structure. The surplus variables and conditions are then eliminated first from the *linearized* system; in doing so the known structure can be exploited. This strategy will be explained for the boundary value problem approach and in the interior point context. Its effect is a shifting of computational effort from "difficult" nonlinear tasks to "easy" linear computations.

Based on the previously explained strategy, the goal of this work is to find the inherent structure of discrete optimal control problems associated with ODE, to develop a general open concept for a class of structured direct SQP methods, and to investigate their common algorithmic core in full detail. Consequently, the KKT solution algorithm will completely be worked out and implemented as a basic module. Other modules may be built on the basic KKT solver to construct various direct SQP methods. Due to increasing freedom of choice on the three higher levels, the description of our general concept becomes gradually less detailed, but on each level we will point out the conditions that are needed to ensure the specific structure. Details of the nonlinear, iterative algorithmic parts, such as the choice of merit functions or step-size strategies, are not subject of this work; these can be adopted from well-tried SQP and interior point methods.

On the highest level we consider optimal control problems in a certain standard form. Only one structural condition must be imposed, namely that multipoint boundary conditions are second order decoupled. The basic multistage structure on the NLP level is then induced by the boundary value problem approach. It appears automatically in the case of multiple shooting, and requires a clever choice of the discretization in collocation, implying mild additional conditions. Without further restrictions this yields a multistage block-sparse structure in the Hessian and constraint matrices appearing on the QP level.

If the QP is treated by an active set strategy, its structure is obviously reproduced in the KKT subproblems. The same proves true for a basic interior point approach, but in the spirit of the central strategy we will develop an improved version to exploit the typical substructure of inequality constraints.

A thorough analysis on the KKT level, based on the theory of dynamic programming, allows to determine a general block-level elimination scheme independently of the specific problem. Each elimination step and the required regularity assumptions will be seen to possess higher level interpretations. The fixed elimination scheme leads to a highly efficient recursive solution algorithm, consisting of alternating local hierarchical projections and local minimizations of the objective.

Organization

The dissertation is divided into five chapters and an appendix. In chapter 1 we introduce the problem statement in the standard formulation, summarize basic results from classical optimization and optimal control theory, and discuss numerical issues of the boundary value problem approach. Chapter 2 presents the general concept of our structured direct SQP approach, i.e., the descent from the top level of infinite optimal control problems to the bottom level of KKT systems. The central part of this work is chapter 3, where the recursive KKT solution algorithm is developed from the projected Hessian method and the Schur complement method as building blocks. Chapter 4 presents the structure-preserving interior point method for the constrained QP, and its refined version. A detailed documentation of the extensive numerical tests is given in chapter 5. The recursive KKT solver is compared to the general sparse solver MA28 and to the band solver from the LAPACK subroutine library. Finally, applications of the solver in multibody system dynamics are presented. Generalized control problem formulations are discussed in the appendix, and transformations for a reduction to standard form are given.

Acknowledgements

I wish to thank Professor Hans Georg Bock for the invaluable help and advice he has given me over the course of my research, and for the time he spent in numerous discussions. I would also like to acknowledge Professor Robert Vanderbei for helpful conversations on the interior point matter, Volker Schulz for many valuable suggestions and discussions, and Reinhold von Schwerin and Michael Winckler for joint work on the applications to multibody systems. Special thanks also to Petra Schulz for proofreading the final text, and to all other people who helped me during my work on this dissertation.

Finally I wish to thank the graduate program „Modellierung und Wissenschaftliches Rechnen in Mathematik und Naturwissenschaften“ of the Interdisciplinary Center for Scientific Computing (Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, IWR) at the University of Heidelberg, both for travel support and for supplemental hardware equipment.

Chapter 1

Basic concepts of optimal control

The following chapter provides some theoretical background and numerical concepts for optimal control problems. First we introduce the problem setting, give general explanations, study an example from robotics, and summarize basic results of the theory of optimization. The second section presents the classical indirect approach which is based on necessary conditions for the optimal control problem in its original infinite form. A discussion of numerical aspects of boundary value problems concludes the chapter. Single shooting, multiple shooting and collocation discretizations are introduced, and the generation of consistent derivatives by Internal Numerical Differentiation is discussed.

1.1 General theory

1.1.1 Problem statement

In practical applications optimal control problems may appear in various formulations. Let us state the basic problem in a standard form on unit interval $I = [0, 1]$.

$$\text{[OCP]} \qquad \begin{aligned} \phi(x(1)) &= \min && (1.1) \\ r(x(t_1), \ldots, x(t_k)) &= 0 && (1.2) \\ \dot{x}(t) - f(x(t), u(t)) &= 0 && (1.3) \\ g(x(t), u(t)) &\geq 0 && (1.4) \end{aligned}$$

The aim is to find a control function u and a state function x that solve [OCP] for given problem functions f, g, r, ϕ. Here ϕ denotes the cost function, r describes boundary conditions, f models the underlying dynamic process, and g specifies path inequality constraints. The independent variable t will be referred to as time. Conditions (1.2–1.4) alone constitute a multi-point boundary value problem (MPBVP) with path constraints. Any pair (u, x) satisfying the MPBVP is called a *feasible* solution; any feasible pair minimizing ϕ is called an *optimal* solution of [OCP].

Regularity assumptions. All problem functions f, g, r, ϕ are assumed to be at least twice continuously differentiable, and to possess higher order derivatives when required:

$$f \in C^2(\mathbb{R}^{n_x+n_u}, \mathbb{R}^{n_x}), \qquad r \in C^2(\mathbb{R}^{n_x \cdot k}, \mathbb{R}^{n_r}), \tag{1.5}$$
$$g \in C^2(\mathbb{R}^{n_x+n_u}, \mathbb{R}^{n_g}), \qquad \phi \in C^2(\mathbb{R}^{n_x}, \mathbb{R}). \tag{1.6}$$

The spaces of admissible control functions and trajectories are

$$u \in L^\infty(I, \mathbb{R}^{n_u}) \tag{1.7}$$

and the Sobolev space

$$x \in H^{1,\infty}(I, \mathbb{R}^{n_x}). \tag{1.8}$$

Range restrictions for states and controls may be specified as path constraints (1.4).

1.1.2 General remarks

Our standard problem [OCP] is an autonomous first order problem with Mayer cost function and fixed time instances $t_1 < \cdots < t_k$. We assume without loss of generality that $t_1 = 0$ and $t_k = 1$, even if initial or final conditions are not specified. This form has been chosen not only for ease of presentation. Fixing the time interval and interior points t_i is important for numerical reasons, so that [OCP] is actually an appropriate formulation for computations. As indicated in the following, optimal control problems are often in a sense more general and in a sense less general than [OCP]. Generalized problems can usually be reduced to the above standard form by simple transformations.

- We assume that higher order differential equations are always written as a first order system. Numerically, however, certain higher order discretization schemes may be applied directly in order to gain higher computational efficiency. This is of special interest in collocation methods, where the derivative information needed for optimization is obtained easily at relatively low cost, cf. section 2.2.3.

- Non-autonomous problems, problems with free initial, final or interior times t_i, or problems with additional system parameters appear rather frequently in practice. The reduction to standard form is given explicitly in appendix A.1 for these problem types. Optimal control problems involving other types of cost functions are treated in appendix A.2.

- The multipoint boundary conditions (1.2) are given in the most general form. For the development of our structured method (chapters 2 and 3) it is essential to require that the boundary conditions are *second order decoupled.* If this decoupling property holds, boundary conditions r can be expressed as

$$r(x_1, \ldots, x_k) = \begin{pmatrix} r^0(x_1, \ldots, x_k) \\ r^1(x_1, \ldots, x_k) \end{pmatrix}, \tag{1.9}$$

where r^0 denotes completely separated constraints,

$$r^0(x_1, \dots, x_k) = \begin{pmatrix} r_1^0(x_1) \\ \vdots \\ r_k^0(x_k) \end{pmatrix}, \tag{1.10}$$

and r^1 denotes linearly coupled constraints,

$$r^1(x_1, \dots, x_k) = r_1^1(x_1) + \dots + r_k^1(x_k). \tag{1.11}$$

The vast majority of application problems (including the important case of periodic optimal control problems) are naturally given in first or second order decoupled form. If this is not the case, nonlinear coupling can always be eliminated by introducing dummy variables, see appendix A.3.

- Certain applications require optimization of multi-phase processes. Each of the individual phases may be governed by different system dynamics—even with differing numbers of state variables, and state jumps may occur at phase transition points. We will not consider such problems in the present framework, although the numerical algorithms developed in the following chapters are designed to handle the resulting discrete problems.

- Path constraints $g^i(x, u)$ are often pure state or control constraints, $g^i(x)$ or $g^i(u)$. Usually they include box constraints on some of the control and state components,

$$u^i_{\min} \le u^i(t) \le u^i_{\max}, \qquad x^i_{\min} \le x^i(t) \le x^i_{\max}. \tag{1.12}$$

Such constraints are typically "hard" limits given by the (physical) nature of the system under consideration. Examples are the voltage range of a motor or the angular range of a robot joint.

- Additional, possibly nonlinear path constraints are often used to model certain security restrictions, such as maintaining a safe distance of a robot gripper to a wall, $w(x(t)) - w_{\text{safe}} \ge 0$. However, "safe" means that small violations of the constraint might occur and are acceptable as long as the gripper does not touch the wall. As in the example, such limits are often "soft" limits.

- When path constraints become active, ODE (1.3) becomes a differential-algebraic equation (DAE), where u is the algebraic variable. Using DAE terminology, control constraints are index 1 constraints (provided $\partial g / \partial u$ is non-singular), whereas state constraints of order k have index $k + 1$, see [80]. Index reduction of the latter leads to more involved optimality conditions in classical theory, cf. section 1.2. Therefore, control constraints are traditionally distinguished from state constraints and given separately in the form of an admissible set, $u(t) \in \mathcal{U}$. No index reduction is performed in direct methods, so the distinction is not that strict and we use a common notation.

- The given regularity assumptions on x, u are suitable for theoretical purposes such as proving existence theorems or optimality conditions; they are of course not useful in practice or numerics. For actual computations u must be piecewise C^l, and x must be continuous and piecewise C^{l+1}, where l depends on the choice of the local discretization scheme. (Note that $x \in C^{l+1}$ requires $f \in C^l$.)

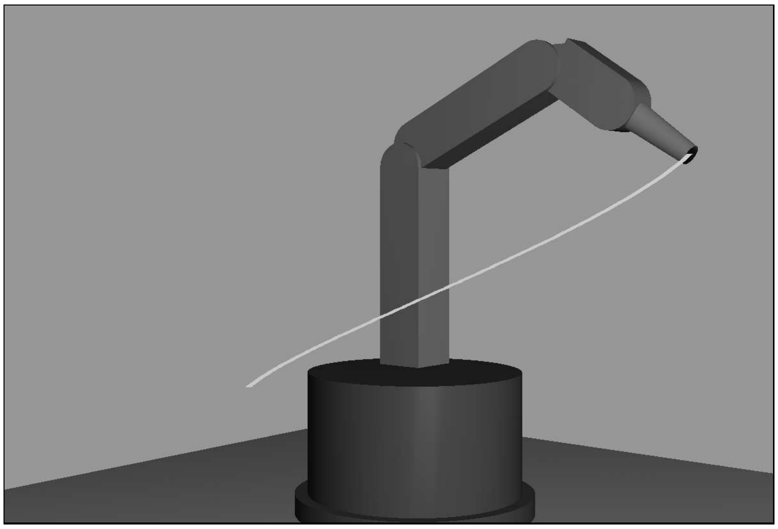
- Even though piecewise differentiability is essential, a finite number of control jumps should be permitted in numerical computations, cf. section 2.1. If the actual control function u^* must be differentiable n times, say, then $u = d^n u^*/dt^n$ should be regarded as control in the numerical formulation. Derivatives up to order $n-1$ and u^* itself will then become state functions.

Example. Minimum time point-to-point maneuver of a robot. Here we consider the problem of steering a robot manipulator from an initial position $q(0) = A$ to a final position $q(T) = B$, where the duration of the maneuver, T, is being minimized. We assume that the robot is at rest initially and finally, $\dot{q}(0) = 0 = \dot{q}(T)$. Such problems have been studied for various basic robot types and maneuvers in [81, 55, 57, 56, 83, 84, 85]. Related investigations for satellite mounted robots can be found in [78, 79].

Point-to-point (PTP) robot maneuvers are a very common task in industry. Examples are the transport of an object from A to B, picking up or replacing a tool, or simply traversing a distance to start working at B after finishing a task at A. A different common type of maneuver (which will not be considered here) is motion along a prescribed path, as in grinding, welding, or spray-painting processes.

We restrict ourselves to the most common class of robots, which have the topological structure of a kinematic chain. Such robots are widely used on assembly lines in industry; a typical representative is the elbow robot (figure 1.1).

Figure 1.1: Simplified model of an elbow robot



Chain-structured robots may be modeled as multibody systems (MBS) in minimal coordinates, with one degree of freedom and one motor in each joint. That is, the generalized coordinate vector $q \in I\!R^n$ contains one component for each joint of the robot, and $\dot{q} \in I\!R^n$ contains the corresponding velocities, where n is the number of joints. Joint angles are used as generalized coordinates for revolute joints, and linear displacements are used for prismatic joints. Positions and velocities may be limited by constraints $q^i_{\min} \leq q^i(t) \leq q^i_{\max}$ and $\dot{q}^i_{\min} \leq \dot{q}^i(t) \leq \dot{q}^i_{\max}$. The position and orientation of the tool center point are both functions of the coordinate vector q. The control vector $u \in I\!R^n$ contains torques and forces applied at the joints, with limits $u^i_{\min} \leq u^i(t) \leq u^i_{\max}$.

The motion of the robot is governed by Lagrange's equations, which take the form of a second order semi-implicit ODE,

$$\begin{aligned} M(q)[\ddot{q} + \Gamma(q)(\dot{q},\dot{q})] &= \\ M(q)\ddot{q} - C(q)(\dot{q},\dot{q}) &= \gamma(q) + Du. \end{aligned} \tag{1.13}$$

Here $M(q)$ is the mass matrix which represents the position-dependent inertia of the robot. This matrix is always symmetric and positive definite due to physical reasons, and hence defines a Riemannian metric on the state space.

$\Gamma(q)$ denotes the Levi-Cività covariant derivation associated with $M(q)$; its components $\Gamma^k_{ij}(q)$ are known as the Christoffel symbols. Physically the term $C(q)(\dot{q},\dot{q}) = -M(q)\Gamma(q)(\dot{q},\dot{q})$ includes centrifugal and Coriolis terms which are quadratic with respect to the velocity $\dot{q}$.

$\gamma(q)$ is the gravity vector, and Du denotes the control input, where the constant non-singular matrix D describes the influence of each motor on each of the joints.

The left hand side of (1.13) models all dynamic interactions that are *internal* to the robot, whereas the right hand side collects *external* forces. When external forces vanish, $\gamma(q) + Du = 0$, the robot trajectory $q(t)$ follows a geodesic on the state space endowed with its metric $M(q)$.

Basically (1.13) is just a general form of Newton's force law; it is derived from variational principles in the context of Lagrangian or Hamiltonian mechanics; details can be found in the textbook [6]. For the theory of Riemannian geometry see, e.g., [54].

Equation (1.13) is of course only a simplified mechanical robot model. More realistic modeling might include joint elasticities, friction, and servo equations for the motors. A DAE formulation of the MBS equations will be given in section 5.3.

Let us now write down the standard formulation of the PTP optimal control problem (for the transformation see appendix A.1). The state vector includes position and velocity vectors, and the final time as a parameter,

$$x(t) = \begin{pmatrix} x_0(t) \\ x_1(t) \\ x_2(t) \end{pmatrix} = \begin{pmatrix} T \\ \dot{q}(Tt) \\ q(Tt) \end{pmatrix} \in I\!R^{2n+1}. \tag{1.14}$$

This gives the cost function

$$\phi(x(1)) = x_0(1) = T \tag{1.15}$$

and decoupled two-point boundary conditions

$$r(x(t_1), x(t_k)) = r^0(x(0), x(1)) = \begin{pmatrix} x_1(0) \\ x_2(0) - A \\ x_1(1) \\ x_2(1) - B \end{pmatrix} = \begin{pmatrix} \dot q(0) \\ q(0) - A \\ \dot q(T) \\ q(T) - B \end{pmatrix} \in I\!R^{4n}. \tag{1.16}$$

ODE (1.13) is expanded as a first order system, and the right hand side is scaled by T due to the time transformation,

$$\begin{aligned} f(x,u) &= \begin{pmatrix} 0 \\ M(x_2)^{-1}[C(x_2)(x_1, x_1) + \gamma(x_2) + Du] \\ x_1 \end{pmatrix} \cdot x_0 \\ &= \begin{pmatrix} 0 \\ M(q)^{-1}[C(q)(\dot q, \dot q) + \gamma(q) + Du] \\ \dot q \end{pmatrix} \cdot T. \end{aligned} \tag{1.17}$$

Finally, state and control constraints $g(x, u)$ are given as upper and lower limits on components of x_1, x_2, and u. Their total number may range from $2n$ (control constraints only) up to $6n$ (upper and lower bounds on all the variables). The order of velocity, position and control constraints is 2, 1, 0 respectively.

Fundamental steps for the numerical treatment. Any numerical solution method will have to deal with three major difficulties: Optimal control problems are (1) *infinite* (2) *nonlinear* (3) *constrained optimization* problems. On a computer we can basically solve only *finite-dimensional linear equations* directly. Hence, three "simplifications" must be applied.

(1) Discretization Infinite-dimensional function spaces are approximated by finite-dimensional (sub-)spaces so that functions can be represented by discrete values. In our case discretization applies basically to states and controls x, u (or states and co-states x, λ with the indirect approach), and to the flow operator of the dynamic system. The discretization scheme is either given explicitly or generated internally by a step-size control or grid strategy.

(2) Local approximation Nonlinear problems are solved iteratively by solving a sequence of easier subproblems. In our case derivative methods will be used, and the subproblems are local approximations—linear or quadratic models—of the nonlinear problem.

(3) Optimality conditions First order necessary conditions are applied in order to characterize *stationary* points of the optimization problem by a set of equalities and inequalities. If inequality constraints are present, a stationary point is found iteratively by solving a sequence of equality-constrained subproblems. Upon convergence, second order conditions may be used to distinguish between minima, maxima, and saddle points.

It is worth noting that optimization problems are in general much more complicated (and much harder to solve) than equations due to the potential complexity of inequality constraints, and due to the need for higher order derivatives. A minimum is typically attained at a corner (or edge) of the feasible region, and the geometry in its neighborhood may be arbitrarily complicated. Even if a correct solution is found, certain regularity properties are needed to conclude minimality from local derivative information only.

Different solution methods arise depending on the manner and order in which discretization, approximation and optimality conditions are applied. Two major classes of methods will be distinguished. The traditional *indirect approach* applies optimality conditions to the infinite problem, leaving an infinite nonlinear *equation* in the form of a multipoint boundary value problem to solve (cf. section 1.2). With the *direct approach* a discretization is applied first, leaving a *finite-dimensional* constrained optimization problem (cf. chapter 2). Before we discuss these in detail, the most important results from the theory of optimization will be summarized.

1.1.3 Optimality conditions

Necessary and sufficient optimality conditions characterize the local geometry around a minimum by derivatives of the cost and constraint functions. Section 1.1.3.1 presents the basic theoretical concepts for infinite optimization problems without inequality constraints. A general, abstract formulation using dual spaces is given, emphasizing basic geometric relationships. The theoretical background for finite-dimensional problems including inequality constraints will be summarized in section 1.1.3.2. Here the emphasis is on the additional geometric complexity introduced by inequalities. No proofs will be given throughout these sections.

1.1.3.1 Infinite optimization problems

In this section we introduce the concepts of the Lagrangian and of regular points in order to formulate the most important necessary and sufficient optimality conditions.

Consider the equality-constrained optimization problem

$$F_1(x) = \min \tag{1.18}$$

$$\text{subject to} \quad F_2(x) = 0, \tag{1.19}$$

where operators $F_1 \colon X \to I\!R$, $F_2 \colon X \to Y$ are twice continuously differentiable and X, Y are Banach spaces.

Definition 1 (Feasible set) *The* feasible set *of problem (1.18–1.19) is*

$$\mathcal{F} = F_2^{-1}(\{0\}) = \{x \in X \mid F_2(x) = 0\}. \tag{1.20}$$

Definition 2 (Lagrangian) *The* Lagrangian *of problem (1.18–1.19) is defined as*

$$L \colon X \times Y^* \to I\!R, \qquad L(x, \lambda) = F_1(x) - \lambda \circ F_2(x), \tag{1.21}$$

where Y^* is the dual space of Y. We call x the primal variable, *and λ the* dual variable *or* Lagrange multiplier.

The Lagrangian combines the full problem description into one single function. Its value coincides with the value of the cost function in all feasible points. Conditions on its partial derivatives will characterize a "regular" minimum.

Definition 3 (Regular point) *A feasible point x is called* regular, *if*
(a) the derivative mapping $DF_2(x)$ is surjective, $Y = \mathrm{im}(DF_2(x))$, and
(b) the (closed) kernel $X_0 = \ker(DF_2(x))$ splits, i.e., a closed *complement $X_1 \subset X$ exists so that $X = X_0 + X_1$.*

These conditions ensure that the feasible set has the structure of a differentiable manifold in a neighborhood of x. Hence, $\ker(DF_2(x))$ is the tangent space at x, and derivatives $DF_2(x), D^2F_2(x)$ provide correct first and second order descriptions of the local geometry of $\mathcal{F}$. Note that the topological condition (b) is only needed in proper Banach spaces; it is always satisfied in a Hilbert space X, and in particular in any finite-dimensional space.

Theorem 1 (Necessary conditions) *If problem (1.18–1.19) has a local minimum at a regular point x, then there exists a dual variable $\lambda \in Y^*$ so that the following conditions hold.*
(a) First order necessary conditions.

$$\begin{aligned} 0 = D_x L(x,\lambda) &= DF_1(x) - \lambda \circ DF_2(x) \\ &= DF_1(x) - DF_2(x)^*\lambda \in X^*, && (1.22) \\ 0 = D_\lambda L(x,\lambda) &= -F_2(x) \in Y. && (1.23) \end{aligned}$$

Any regular point x satisfying (a) is called a stationary *point.*
(b) Second order necessary condition. *The bilinear mapping $D^2_{xx}L(x,\lambda)\colon X \times X \to \mathbb{R}$ is positive semi-definite on $\ker(DF_2(x))$,*

$$D^2_{xx}L(x,\lambda)(v,v) \geq 0 \quad \forall v \in \ker(DF_2(x)). \quad \diamondsuit \tag{1.24}$$

From the dual feasibility condition (1.22) follows that the "gradient" of the cost function is "orthogonal" to the tangent space of the feasible manifold at x: $DF_1(x) \in \mathrm{im}(DF_2(x)^*)$. This interpretation is strictly true only if X is a Hilbert space, of course. In Banach spaces gradients and orthogonality are not defined due to the lack of a scalar product. Hence, we must rely on dual spaces and dual mappings to express similar facts here. Condition (1.23) merely restates (primal) feasibility, where $-F_2(x) \in Y$ is canonically identified with $D_\lambda L(x,\lambda) \in Y^{**}$.

Condition (b) is a convexity condition on the Lagrangian—not on the cost function. Convexity of the cost function alone would not give enough information, since the curvature of $\mathcal{F}$ must be taken into account. *Strict* convexity of the Lagrangian turns out to be a sufficient condition for a minimum.

Theorem 2 (Sufficient conditions) *If x is a stationary point of problem (1.18–1.19) and the bilinear mapping $D^2_{xx}L(x,\lambda)$ is positive on* $\ker(DF_2(x))$,

$$D^2_{xx}L(x,\lambda)(v,v) > 0 \quad \forall v \in \ker(DF_2(x)) \setminus \{0\}, \tag{1.25}$$

then x is a strict *local minimum.* $\diamondsuit$

1.1.3.2 Finite optimization problems

Optimality conditions for finite-dimensional problems with inequality constraints have been derived independently by Karush [52] and by Kuhn and Tucker [60], therefore they are called Karush-Kuhn-Tucker (KKT) conditions. In this section the most important definitions and results will be presented following the lines of Fiacco and McCormick [35, Chapter 2], where details and proofs can be found. However, theorems will usually be formulated under stronger assumptions which are suitable for numerical verification.

Consider the general nonlinear programming problem

$$\text{[NLP]} \qquad \text{subject to} \qquad \begin{aligned} F_1(x) &= \min & (1.26)\\ F_2(x) &= 0, & (1.27)\\ F_3(x) &\geq 0. & (1.28)\end{aligned}$$

Here $F_1\colon I\!R^n \to I\!R$, $F_2\colon I\!R^n \to I\!R^{n_2}$, $F_3\colon I\!R^n \to I\!R^{n_3}$ are twice continuously differentiable, and inequalities F_3 are understood component-wise. The definition of the Lagrangian and the necessary conditions are similar to the infinite case, except that spaces $I\!R^{n_i}$ are canonically identified with their respective duals, and derivatives $DF_i(x)$ are represented by their respective Jacobians $J_i(x)$.

Definition 4 (Lagrangian) *The Lagrangian of problem [NLP] is*

$$L\colon I\!R^n \times I\!R^{n_2} \times I\!R^{n_3} \to I\!R, \qquad L(x,w_2,w_3) = F_1(x) - w_2^T F_2(x) - w_3^T F_3(x). \tag{1.29}$$

Definition 5 (Feasible set) *The feasible set of problem [NLP] is*

$$\mathcal{F} = \{x \in I\!R^n \mid F_2(x) = 0, F_3(x) \geq 0\}. \tag{1.30}$$

Definition 6 (Active set) *The* active *constraints at a feasible point x of problem [NLP] are given by the index set*

$$\mathcal{I}(x) = \{i \mid F_{3i}(x) = 0\} = \{i_1, \dots, i_{\tilde{n}_3}\}. \tag{1.31}$$

Notation: Let

$$\tilde{F}_3(x) = \begin{pmatrix} F_{i_1}(x) \\ \vdots \\ F_{i_{\tilde{n}_3}}(x) \end{pmatrix}, \qquad F_c(x) = \begin{pmatrix} F_2(x) \\ \tilde{F}_3(x) \end{pmatrix} \in I\!R^{n_c}, \tag{1.32}$$

and denote by $\tilde{J}_3(x), J_c(x)$ the corresponding Jacobians.

In the presence of inequalities, the feasible set is usually a proper subset of $F_2^{-1}(\{0\})$. If the active set $\mathcal{I}(x)$ is non-empty, then x lies in a corner, edge, or higher-dimensional face of the boundary; that face is locally given by $F_c = 0$. None of the constraints F_{3i} is active in the relative interior of $\mathcal{F}$. The linearized feasible set, which is a cone, will be partitioned in the following definition according to the behavior of the cost function.

Definition 7 *At a feasible point x of problem [NLP] we define the following subsets of $I\!R^n$.*

(i) The cone of feasible directions (also called feasible perturbations),

$$Z_0(x) = \{z \in \ker(J_2(x)) \mid \tilde{J}_3(x)z \geq 0\}, \tag{1.33}$$

(ii) the directions of increase,

$$Z_1(x) = \{z \in Z_0(x) \mid J_1(x)z \geq 0\}, \tag{1.34}$$

(iii) the directions of strict decrease,

$$Z_2(x) = \{z \in Z_0(x) \mid J_1(x)z < 0\}, \tag{1.35}$$

(iv) and the set of infeasible directions,

$$Z_3(x) = I\!R^n \setminus Z_0(x). \tag{1.36}$$

Observe that $0 \in Z_1$ and $I\!R^n = Z_0 \uplus Z_3 = (Z_1 \uplus Z_2) \uplus Z_3$ for all x, where $\uplus$ denotes a disjoint union.

The next theorem, which follows from the lemma of Farkas [34], is fundamental in developing the following theory. It establishes a purely geometric condition for the existence of Lagrange multipliers, relating them to the set of directions of strict decrease, $Z_2(x)$. In the remainder of this section D_x, D_w denote gradients with respect to x, w, i.e., column vectors rather than linear mappings.

Theorem 3 (Existence of Lagrange multipliers) *If and only if the set of decrease directions at a feasible point x is empty, $Z_2(x) = \emptyset$, there exist Lagrange multipliers w_2, w_3 satisfying*

$$D_x L(x, w_2, w_3) = 0, \tag{1.37}$$

$$w_3 \geq 0, \tag{1.38}$$

$$w_{3i} F_{3i}(x) = 0 \ \forall i. \tag{1.39}$$

The complementarity condition *(1.39) can also be written $w_3^T F_3(x) = 0$, since $F_3(x) \geq 0$ and $w_3 \geq 0$.* $\diamondsuit$

As in the infinite case, we need an additional regularity condition in order to characterize a minimum by conditions (1.37–1.39), i.e., to ensure that $Z_2(x)$ is empty at the minimum. Here the topological condition (b) of definition 3 is not needed, but we must take into account active inequalities.

Definition 8 (Constraint Qualification) *A feasible point x is called* regular, *if the augmented Jacobian $J_c(x)$ is surjective. This condition is called* constraint qualification,

$$\text{[CQ]} \qquad \operatorname{rank}(J_c(x)) = n_c. \tag{1.40}$$

Theorem 4 (Karush-Kuhn-Tucker first order necessary conditions) *If problem [NLP] has a local minimum at a regular point x, then unique Lagrange multipliers w_2, w_3 exist, so that conditions (1.37–1.39) are satisfied. Including primal feasibility, the full set of necessary conditions is*

$$\begin{aligned} J_1(x)^T - J_2(x)^T w_2 - J_3(x)^T w_3 = D_x L(x, w_2, w_3) &= 0, & (1.41)\\ F_2(x) = -D_{w_2} L(x, w_2, w_3) &= 0, & (1.42)\\ F_3(x) = -D_{w_3} L(x, w_2, w_3) &\geq 0, & (1.43)\\ w_3 &\geq 0, & (1.44)\\ w_3^T F_3(x) &= 0. & (1.45) \end{aligned}$$

A triple (x, w_2, w_3) satisfying (1.41–1.45) is called a KKT point. $\diamondsuit$

Second order conditions are again related to convexity of the Lagrangian, namely on a certain cone which is a subset of $Z_0(x)$. A description of that cone involves strictly active constraints.

Definition 9 (Strict complementarity) *At a KKT point (x, w_2, w_3) the set of* strictly active *constraints is given by*

$$\hat{\mathcal{I}}(x) = \{i \in \mathcal{I}(x) \mid w_{3i} > 0\}. \tag{1.46}$$

Strict complementarity *holds if each active constraint is strictly active,*

$$\text{[SC]} \qquad \hat{\mathcal{I}}(x) = \mathcal{I}(x). \tag{1.47}$$

This means that either $F_{3i}(x) = 0$ and $w_{3i} > 0$ or $F_{3i}(x) > 0$ and $w_{3i} = 0$ for all i.

Theorem 5 (Second order necessary conditions) *If a local minimum of problem [NLP] is attained at a regular point x, then the Hessian of the Lagrangian, $H(x, w_2, w_3) = D^2_{xx} L(x, w_2, w_3)$, is positive semi-definite on the cone*

$$C(x) = \{z \in Z_0(x) \mid J_{3i}(x) z = 0 \ \forall i \in \hat{\mathcal{I}}(x)\}. \quad \diamondsuit \tag{1.48}$$

Obviously, $C(x)$ contains the subspace $\ker(J_c(x))$. In the case of strict complementarity these sets are identical.

Theorem 6 (Second order sufficient conditions) *If (x, w_2, w_3) is a KKT point and the Hessian of the Lagrangian is positive definite on $C(x)$,*

$$v^T H(x, w_2, w_3) v > 0 \quad \forall v \in C(x) \setminus \{0\}, \tag{1.49}$$

then x is a strict *local minimum. (Note that x does not need to be regular!)* $\diamondsuit$

When strict complementarity holds, such a minimum can be shown to be stable with respect to small perturbations of the problem data. (This is a consequence of the implicit function theorem.) Stability of the minimum is essential for numerical computations, since roundoff errors always cause small perturbations. Therefore we conclude this section with some fundamental stability results. Consider the embedded optimization problem

$$\begin{aligned} \bar F_1(x;\varepsilon) &= \min_x && (1.50)\\ \text{subject to} \quad \bar F_2(x;\varepsilon) &= 0, && (1.51)\\ \bar F_3(x;\varepsilon) &\geq 0, && (1.52)\end{aligned}$$

with Lagrangian

$$\bar L(x, w_2, w_3; \varepsilon) = \bar F_1(x;\varepsilon) - w_2^T \bar F_2(x;\varepsilon) - w_3^T \bar F_3(x;\varepsilon), \tag{1.53}$$

where $\bar F_i \in C^2(I\!R^n \times I\!R^l, I\!R^{n_i})$ depend on a perturbation parameter $\varepsilon \in I\!R^l$, and the case $\varepsilon = 0$ represents the original problem, $F_i(x) = \bar F_i(x; 0)$.

Theorem 7 (Perturbation of minimum) *If (a) F_i satisfy the sufficient conditions of theorem 6 at (x, w_2, w_3), (b) the constraint qualification [CQ] and (c) strict complementarity [SC] hold, then there exist open neighborhoods $\mathcal{E} \ni 0$ and $\mathcal{Z} \ni (x, w_2, w_3)$, and a C^1-mapping $z\colon \mathcal{E} \to \mathcal{Z}$, so that*

(i) $z(\varepsilon) = (x(\varepsilon), w_2(\varepsilon), w_3(\varepsilon))$ is the unique KKT point of the perturbed problem on $\mathcal{Z}$.

(ii) The perturbed problem has a strict local minimum at $x(\varepsilon)$.

(iii) The active set $\mathcal{I}(x(\varepsilon)) = \hat{\mathcal{I}}(x(\varepsilon))$ is constant on $\mathcal{E}$.

(iv) First order perturbations of the KKT point $z(0)$ are given by

$$\begin{pmatrix} x'(0) \\ -w_c'(0) \end{pmatrix} = -\begin{pmatrix} \bar H(z(0);0) & \bar J_c(x(0);0)^T \\ \bar J_c(x(0);0) & 0 \end{pmatrix}^{-1} \begin{pmatrix} D_\varepsilon D_x \bar L(z(0);0) \\ -D_\varepsilon D_{w_c} \bar L(z(0);0) \end{pmatrix} \tag{1.54}$$

$$w_{3i}'(0) = 0 \quad \forall i \notin \mathcal{I}(x(0)), \tag{1.55}$$

where the meaning of $\bar F_c, \bar J_c, w_c$ is analogous to definition 6. $\diamondsuit$

Corollary 1 (Perturbation of cost function) *Under the assumptions of theorem 7 we have*

$$\frac{d}{d\varepsilon}\bar{F}_1(x(\varepsilon);\varepsilon)\Big|_{\varepsilon=0} = \frac{\partial}{\partial\varepsilon}\bar{L}(z(0);\varepsilon)\Big|_{\varepsilon=0}. \tag{1.56}$$

If, in particular, $\bar{F}_1(x;\varepsilon) \equiv F_1(x)$, $\bar{F}_2(x;\varepsilon) \equiv F_2(x)$, $\bar{F}_3(x;\varepsilon) = F_3(x) - \varepsilon e_i$, $\varepsilon \in I\!R$, *then*

$$\frac{d}{d\varepsilon}F_1(x(\varepsilon))\Big|_{\varepsilon=0} = w_{3i}(0), \tag{1.57}$$

or equivalently,

$$F_1(x(\varepsilon)) = F_1(x(0)) + \varepsilon w_{3i}(0) + \mathcal{O}(\varepsilon^2). \quad \diamondsuit \tag{1.58}$$

This corollary allows an interpretation of the dual variables. If constraint i is slightly enforced ($\varepsilon > 0$), then the cost grows roughly by an amount $\varepsilon w_{3i} > 0$. Vice versa, if it is relaxed ($\varepsilon < 0$), then the cost decreases accordingly. Therefore the dual variables are often called *shadow prices* in economics. If constraint i is inactive, $w_{3i} = 0$, it has no influence on the cost in a neighborhood of $x(0)$, and hence might be dropped from the problem. Strict complementarity means that none of the constraints just happens to be active at a KKT point.

1.2 An indirect approach

The classical, indirect approach to optimal control problems is based on optimality conditions for the *infinite* problem, which eliminate the control by expressing it as a function of state and co-state (primal and dual) variables. For a variety of simple problems the exact solution can be found analytically. In general, however, even the numerical solution of the symplectic multipoint boundary value problem representing the necessary conditions is rather difficult. A more severe problem is that the switching structure of the solution must already be known just to formulate the necessary conditions correctly. Therefore, from a practical viewpoint, the indirect approach is only suitable for problems with low dimension and a small number of state inequality constraints. Even then the solution structure may be surprisingly complicated (see, e.g., [81, 83, 85]).

1.2.1 Pontryagin's Minimum Principle

Optimality conditions for control problems can be found, e.g., in the classical textbooks by Pontryagin et al. [70], Bellman [13], or Isaacs [49] who studies the more general problem class of differential games. Here we state the well-known minimum principle due to [70] in a basic version for the *regular* case without path constraints. That is, the second derivative of the Hamiltonian with respect to the control variables is assumed to be positive definite along the solution trajectory, $H_{uu}(x(t), u(t), \lambda(t)) > 0$. We will not go into details about the constrained case, but summarize some principal statements in section 1.2.3.

Theorem 8 (Minimum Principle) *Let (x,u) be an optimal solution of (1.1–1.3), i.e., [OCP] without path constraints. Then there exist Lagrange multipliers*

$$\alpha \in I\!R^{n_r} \tag{1.59}$$

and co-state variables *(also called* adjoint variables*)*

$$\lambda \in \prod_{i=1}^{k-1} H^{1,\infty}([t_i, t_{i+1}), I\!R^{n_x}) \tag{1.60}$$

which satisfy adjoint differential equations

$$\dot{\lambda}(t)^T = -\lambda(t)^T f_x(x(t), u(t)) \tag{1.61}$$

on each subinterval (t_i, t_{i+1}), *and* transversality conditions

$$\lambda(t_1+) - \alpha^T r_{x_1}(x(t_1), \ldots, x(t_k)) \;=\; 0, \tag{1.62}$$

$$\lambda(t_i+) - \lambda(t_i-) - \alpha^T r_{x_i}(x(t_1), \ldots, x(t_k)) \;=\; 0, \quad 1 < i < k, \tag{1.63}$$

$$\phi'(x(t_k)) - \lambda(t_k-) - \alpha^T r_{x_k}(x(t_1), \ldots, x(t_k)) \;=\; 0. \tag{1.64}$$

Furthermore, the optimal control u satisfies

$$\lambda(t)^T f_u(x(t), u(t)) = 0, \tag{1.65}$$

and the Hamiltonian

$$H(x, u, \lambda) = \lambda^T f(x, u) \tag{1.66}$$

is constant along the optimal solution $(x(t), u(t), \lambda(t))$. $\diamondsuit$

The name minimum principle is related to condition (1.65) which implicitly determines the optimal control $u(t) = u^*(x(t), \lambda(t))$. Under our regularity assumption, $H_{uu} > 0$, u^* is the unique unconstrained minimum, i.e., $\lambda^T f_u = H_u = 0$ is equivalent to

$$H(x(t), u(t), \lambda(t)) = \min_{u^*} H(x(t), u^*, \lambda(t)). \tag{1.67}$$

Traditionally this minimization condition appears in the theorem instead of (1.65). It remains true in the constrained case, except that the definition of H must be modified (see section 1.2.3).

In the original work [70] and in much of the literature, the same theorem is referred to as *maximum* principle. In this case α, λ are defined with the opposite sign, so that the Hamiltonian is maximized rather than minimized by u^*.

In the following section we will see that the minimum principle can be seen as a special case of necessary conditions (1.22–1.23), at least in the case of a regular Hamiltonian. Only the algebraic relationship will be indicated; the difficult part of proving the minimum principle is to establish regularity of a given solution (x, u). Existence of an optimal solution has been shown, e.g., by Filippov [36] under certain convexity assumptions, or by Frehse [37, 38], who makes use of parameterized control functions in a partially constructive proof.

1.2.2 The symplectic multipoint boundary value problem

The primal feasibility condition of the optimal control problem is a multipoint boundary value problem; it consists of an infinite part (the state differential equation 1.3) and a finite part (the boundary conditions 1.2). Accordingly, infinite and finite dual variables λ, α appear in co-state differential equations and transversality conditions, respectively. If we set $\lambda_1^- = \lambda_k^+ = 0$ and define an augmented cost function

$$G(x_1, \ldots, x_k, \alpha, \lambda_1^\pm, \ldots, \lambda_k^\pm) = \phi(x_k) - \alpha^T r(x_1, \ldots, x_k) - \textstyle\sum_i (\lambda_i^- - \lambda_i^+)^T x_i, \tag{1.68}$$

then finite and infinite first order conditions can be expressed in terms of partial derivatives of G, H, constituting the symplectic multipoint boundary value problem

$$\dot{x}(t) = H_\lambda(x(t), u(t), \lambda(t))^T, \tag{1.69}$$
$$\dot{\lambda}(t) = -H_x(x(t), u(t), \lambda(t))^T, \tag{1.70}$$
$$0 = H_u(x(t), u(t), \lambda(t))^T, \tag{1.71}$$

$$0 = G_\alpha(x(t_1), \ldots, x(t_k), \alpha, \lambda(t_1\pm), \ldots, \lambda(t_k\pm))^T, \tag{1.72}$$
$$0 = G_x(x(t_1), \ldots, x(t_k), \alpha, \lambda(t_1\pm), \ldots, \lambda(t_k\pm))^T. \tag{1.73}$$

Here (1.69–1.71) form a symplectic DAE, where $H_u = 0$ is an index 1 condition for the algebraic variable u. All these conditions follow from (1.22–1.23) when the Lagrangian is written as

$$L(x, u; \alpha, \lambda) = G(x, \alpha, 0) + \textstyle\sum_i \displaystyle\int_{t_i}^{t_{i+1}} [H(\lambda, x, u) - \lambda^T \dot{x}]\, dt \tag{1.74}$$
$$= G(x, \alpha, \lambda) + \textstyle\sum_i \displaystyle\int_{t_i}^{t_{i+1}} [H(\lambda, x, u) + \dot{\lambda}^T x]\, dt. \tag{1.75}$$

Both forms use an integral representation for the infinite dual variables; they are related to each other by a partial integration leading to the boundary integral term in G. Primal conditions are most easily derived from (1.74), dual conditions from (1.75).

Highly accurate and efficient numerical methods exist for the solution of multipoint boundary value problems, such as the multiple shooting code PARFIT [16, 17, 80, 40] and the collocation code COLFIT [12, 76]. These methods combine a generalized Gauß-Newton iteration with a BVP discretization (cf. section 1.3).

However, the indirect approach always bears some specific difficulties. Firstly, the system of state and co-state differential equations is potentially unstable due to its symplectic nature; this can cause rather severe numerical problems. Secondly, since the co-states have in general no physical meaning, it may be hard to come up with good initial guesses for the Newton iteration. Finally, computing u as a function of x, λ often introduces additional, highly nonlinear coupling into the system of differential equations.

1.2.3 Path inequality constraints

We have seen that the solution (x, u, λ) of a regular control problem satisfies the symplectic DAE (1.69–1.71), where index 1 condition (1.71) determines u on the whole time interval. (A closer examination shows that u is even differentiable.) When path constraints are present, the Hamiltonian is redefined as

$$H(x, u, \lambda, \mu) = \lambda^T f(x, u) + \mu^T g(x, u), \tag{1.76}$$

and the symplectic MPBVP remains formally unchanged. For numerical computations, however, condition (1.71) must be replaced by more complicated formulations, since nontrivial switching structures will usually occur:

- different algebraic conditions may apply on different subintervals,
- higher index algebraic conditions may occur,
- additional necessary conditions may arise, such as entry and exit conditions associated with constraint arcs.

To be more precise, define a time-dependent active set

$$\mathcal{I}(t) = \{i \mid g^i(x(t), u(t)) = 0\} \tag{1.77}$$

for each point $t \in I$. We assume that only finitely many changes of $\mathcal{I}$ occur along the solution (x, u), say at *switching points* t_i^*. At these points u may have jumps or higher order discontinuities. Since numerical discretization methods require certain differentiability properties, the switching points must be located exactly to be included in the discretization scheme. Except for simple cases this is impossible unless the sequence of active sets $\mathcal{I}|_{(t_i^*, t_{i+1}^*)}$ is correctly specified *a priori*; hence the switching structure must be known to compute a solution.

Since in general it is impossible to find the switching structure of a solution just by careful examination of the problem, one must perform a kind of "manual active set strategy". Two principle techniques are commonly used.

(a) Test the necessary conditions on a set of "candidate" solution structures in order to find the ones that admit local minima. This technique is usually applied to *control* constraints.

(b) Solve the relaxed problem *without* path constraints first, then drive the constraint violation to zero using a *homotopy* technique. The new structure in each homotopy step can be predicted based on the previous step's structure and theoretical considerations, or can be found according to strategy (a) in more difficult cases. This technique is usually applied to *state* constraints.

The distinction has several reasons. First, control constraints are generically present. In fact, they are often needed to ensure the existence of a minimum. In the example of chapter 1, for instance, the robot could perform any maneuver in an arbitrarily short time $T > 0$ if the motor torques were unbounded. Second, pure control constraints are relatively easy to handle. They restrict u to some admissible set $\mathcal{U} = \{u \mid g(u) \geq 0\}$ which is typically compact. When (1.65) is replaced by a constrained version of (1.67),

$$H(x(t), u(t), \lambda(t)) = \min_{u^* \in \mathcal{U}} H(x(t), u^*, \lambda(t)), \tag{1.78}$$

the minimum principle applies without any additional or higher order conditions, and without including $\mu^T g$ in the Hamiltonian.

On the other hand, state constraints are always of higher order (and index), hence their handling may be much more difficult. Index reduction must be performed and additional necessary conditions are produced. Similar phenomena arise in problems with linear control, especially if singular arcs occur.

To sum up, a correct solution structure is in general detected only after solving a sequence of MPBVP's, each of which may involve complicated analytical derivations. This is an iterative, interactive process requiring a person who has enough experience with the problem itself as well as detailed knowledge of the mathematical background.

Certainly, so much effort will only be spent in a limited number of practical applications. It makes sense if substantial savings are anticipated, if the dynamics are not too complicated, and if sufficiently accurate models are available. Typical applications include, e.g., optimization of the flight path of a spacecraft, or minimization of energy consumption in a subway system, see [58, 18, 19].

1.3 Solving boundary value problems

This section is devoted to the numerical treatment of pure boundary value problems such as the symplectic MPBVP arising in the indirect approach. To be more specific, we study the questions how to *discretize* and how to *linearize* a general MPBVP

$$\Phi(x) = \begin{pmatrix} r(x(t_1), \ldots, x(t_k)) \\ \dot{x} - f(x) \end{pmatrix} = 0, \tag{1.79}$$

where

$$\Phi\colon H^{1,\infty}(I, I\!R^n) \to I\!R^n \times L^\infty(I, I\!R^n) \tag{1.80}$$

is twice continuously differentiable, and $\operatorname{rank}(Dr) \equiv n$. The results of this section apply to any boundary value problem. In particular, we will make use of them also in the context of the direct approach, see chapter 2.

1.3.1 Discretization

The process of discretizing a boundary value problem can be divided into two stages. First, a discrete *global* representation of the trajectory is chosen to replace the BVP by a finite-dimensional nonlinear equation. Second, a *local* discretization scheme for the flow operator of the dynamic system is generated for actual numerical computations. In the first section we will introduce two fundamental classes of global representations: the single shooting or initial value problem (IVP) approach, and multiple shooting as a typical representative of the boundary value problem (BVP) approach. After treating the issue of consistent linearization within a Newton iteration, collocation will be discussed as a second major class of BVP discretizations.

1.3.1.1 The flow operator

We wish to find a global representation $x_g \in I\!R^N$ of the trajectory $x \in H^{1,\infty}$ in order to obtain a finite-dimensional BVP

$$\Phi_g(x_g) = 0, \qquad \Phi_g \in C^2(I\!R^N, I\!R^N) \tag{1.81}$$

which is *equivalent* to the original infinite BVP, $\Phi(x) = 0$. This is possible due to the fact that the solution set

$$X_f = \left\{ x \in H^{1,\infty} \mid \dot{x} - f(x) = 0 \right\} \tag{1.82}$$

of the ODE is an n-dimensional manifold, and *any* "initial" point t^* canonically defines a global chart

$$\delta_{t^*} \colon X_f \to I\!R^n, \quad \delta_{t^*}(x) = x(t^*). \tag{1.83}$$

The inverse mapping is given by

$$F_{t^*} \colon I\!R^n \to X_f, \quad F_{t^*}(\,\cdot\,, x^*) = F(\,\cdot\,, t^*, x^*), \tag{1.84}$$

where

$$F(t, t^*, x^*) = x^* + \int_{t^*}^{t} f(F(s, t^*, x^*))\, ds \tag{1.85}$$

implicitly defines the flow operator F associated with f. In the following we will derive several possible representations Φ_g in terms of the flow operator.

1.3.1.2 The initial value problem approach: single shooting

Applying the IVP approach, we use the flow to eliminate the ODE first by tracing back each $x(t_i)$ to $x_1 = x(t_1)$:

$$x(t_i) = F(t_i, t_1, x_1), \qquad i = 2(1)k. \tag{1.86}$$

The results are then simply plugged into r, and the BVP is stated in terms of the initial value $x_g \equiv x_1$ of a *feasible* trajectory:

$$\Phi_g(x_1) = r(x_1, F(t_2, t_1, x_1), \ldots, F(t_k, t_1, x_1)). \tag{1.87}$$

In practice the values $F(t_i, t_1, x_1)$ are computed by numerical integration over the whole interval $[t_1, t_k] = [0, 1]$, and a Newton iteration is performed on $x_1 \in I\!R^n$ until $\Phi_g(x_1) = 0$. This method is known as single shooting.

1.3.1.3 The boundary value problem approach: multiple shooting

The BVP approach is characterized by a *divide and conquer* strategy. A grid divides the time interval into a number of subintervals on which *independent* local initial value problems are solved, possibly in parallel. These solutions are feasible segments of an *infeasible* trajectory with jump discontinuities at the nodes. A global Newton iteration on all the local initial values $x_i \in I\!R^n$ is now performed to solve boundary conditions and additional continuity conditions simultaneously, thus treating the ODE as a side condition.

Technically the procedure is as follows. Choose a grid $\Gamma_x: 0 = \tau_1 < \cdots < \tau_m = 1$ with $m \geq k$ nodes, which, for simplicity and without loss of generality, include all instances where boundary conditions are evaluated: $t_i = \tau_{\sigma(i)}$ for some $\sigma(i)$. Then define variables $x_j = x(\tau_j)$, $x_g = (x_1, \ldots, x_m)$ and trajectory segments

$$x(t) = F(t, \tau_j, x_j) \quad \text{on } [\tau_j, \tau_{j+1}). \tag{1.88}$$

Finally, specify continuity conditions

$$h_j(x_j, x_{j+1}) = F(\tau_{j+1}, \tau_j, x_j) - x_{j+1}, \qquad j = 1(1)m-1. \tag{1.89}$$

This transforms the BVP into

$$\Phi_g(x_g) = \begin{pmatrix} h_1(x_1, x_2) \\ \vdots \\ h_{m-1}(x_{m-1}, x_m) \\ r(x_{\sigma(1)}, \ldots, x_{\sigma(k)}) \end{pmatrix} = 0 \tag{1.90}$$

where

$$\Phi_g \in C^2(I\!R^N, I\!R^N), \quad N = mn. \tag{1.91}$$

This method is called multiple shooting. If only one subinterval is chosen, $m = 2$, multiple shooting reduces to a (slightly stabilized) single shooting variant, where final trajectory values and a single continuity condition are explicitly included. An alternative BVP approach, the collocation method, will be described below.

Both IVP and BVP methods need solutions of initial value problems, which are obtained through local discretizations of the flow operator on each subinterval. In single and multiple shooting methods the local discretization scheme is usually generated by the integration routine, whereas it is prescribed explicitly in collocation methods (see below).

Numerical integration on subinterval $[\tau_j, \tau_{j+1}]$ yields a local representation of the trajectory which can be written in the form

$$p_j(t, x_j) = P_j[\Gamma_j^0](t, x_j) \approx F_{\tau_j}(t, x_j). \tag{1.92}$$

Here Γ_j^0 denotes the local grid and $P_j[\Gamma_j^0]$ denotes the corresponding discretization of the flow F_{τ_j}. In practice p_j may be defined in the grid points only, and only the final value $p(\tau_{j+1}, x_j)$ may be accessible after the integration. The grid

$$\Gamma_j^0 = \Gamma_j(\tau_j, \tau_{j+1}, x_j, \varepsilon_j, h_j^0) \tag{1.93}$$

is chosen by the step-size selection of the integration routine. Typically it depends on interval bounds τ_j, τ_{j+1}, initial value x_j, desired accuracy ε_j and initial stepsize h_j^0, possibly also on stepsize limits $h_{\min}, h_{\max}$. (Of course, a uniform step-size $h_j \equiv h_j^0$ may also be used.) We will have a closer look at the integration mechanism in the context of consistent linearization.

1.3.1.4 Comparison of IVP and BVP approach

We have seen that both approaches yield formally equivalent finite representations of the BVP, which can be solved by Newton's method. At first sight the IVP approach might appear more natural than the BVP approach with its artificial variables $x_2, \ldots, x_m$ and corresponding continuity conditions $h_1, \ldots, h_{m-1}$. Certainly the higher number of variables increases the effort being spent on linear algebra during iterative solution.

However, single shooting has a severe disadvantage. During numerical integration over the *whole* interval the trajectory may drift off very far from the exact IVP solution, or worse, it may run into singularities. This may happen even if the correct initial value x_1 is given with full machine precision. Such phenomena typically occur with the indirect approach due to the inherent instability of symplectic boundary value problems. As a consequence of the drift-off, convergence of the Newton iteration is often slowed down considerably. In the presence of a singularity the BVP cannot be solved at all.

In the BVP approach drift-off can always be controlled by choosing sufficiently small subintervals. The subdivision into "short" intervals also leads to a reduction of nonlinearity and an enlarged domain of convergence. For a specific class of parameter estimation problems one can even prove one-step convergence for arbitrary initial parameters [76]. Furthermore, any local information on the solution trajectory can be incorporated in the problem formulation through appropriate initial estimates, thereby speeding up the iterative solution. These properties make the BVP approach much more stable and robust than the IVP approach. In fact, the former is even more efficient on sufficiently hard problems, since the faster convergence due to reduced nonlinearities then over-compensates for the linear algebra. Finally, independence of local computations in the BVP approach offers an enormous potential of parallelism.

1.3.2 Consistent linearization

When the boundary value problem $\Phi_g(x_g) = 0$ is solved by Newton's method, a sequence of iterates x_g^k is generated from an initial guess x_g^0 according to $x_g^{k+1} = x_g^k + \alpha^k \Delta x_g^k$, where α^k is a suitable step-size and Δx_g^k solves the linearized problem $\Phi_g(x_g^k) + \Phi_g'(x_g^k)\Delta x_g^k = 0$. Hence, numerical approximations of $\Phi_g(x_g^k)$ and its derivative $\Phi_g'(x_g^k)$ are needed in each iteration step. In multiple shooting, the derivative matrix is

$$
\Phi_g'(x_g) = \begin{pmatrix} G_1 & -I & & \\ & \ddots & \ddots & \\ & & G_{m-1} & -I \\ D_1 & \ldots & \ldots & D_m \end{pmatrix}, \tag{1.94}
$$

where

$$
G_j = \frac{\partial h_j}{\partial x_j}(x_j, x_{j+1}) = \frac{\partial F}{\partial x_j}(\tau_{j+1}, \tau_j, x_j), \qquad D_j = \frac{\partial r}{\partial x_j}(x_{\sigma(1)}, \ldots, x_{\sigma(k)}). \tag{1.95}
$$

Derivatives D_j of the boundary conditions are either given analytically or may be numerically approximated by finite differences. Computation of the Wronskian matrices G_j is more involved, and some care must be taken.

We have seen that numerical integration of the nominal trajectory yields approximations $p_j(\tau_{j+1}, x_j) = P_j[\Gamma_j^0](t, x_j)$ of the local terminal values $F(\tau_{j+1}, \tau_j, x_j)$. If we regard this whole integration process as a black box, derivatives can be approximated in several ways: by finite differences, by integrating the variational differential equation, or by automatic differentiation. Consider, e.g., a finite difference approximation of the Wronskian G_j which is generated by perturbing x_j and integrating varied trajectories. When δ_j^i is the i-th perturbation parameter, then column i of the approximation reads

$$\frac{\partial F}{\partial x_j^i}(\tau_{j+1}, \tau_j, x_j) \approx \frac{F(\tau_{j+1}, \tau_j, x_j + \delta_j^i e_i) - F(\tau_{j+1}, \tau_j, x_j)}{\delta_j^i} \tag{1.96}$$

$$\approx \frac{P_j[\Gamma_j^i](\tau_{j+1}, x_j + \delta_j^i e_i) - P_j[\Gamma_j^0](\tau_{j+1}, x_j)}{\delta_j^i}. \tag{1.97}$$

In general, the stepsize selection scheme will determine different grids

$$\Gamma_j^i = \Gamma_j(\tau_j, \tau_{j+1}, x_j + \delta_j^i e_i, \varepsilon_j, h_j^0) \tag{1.98}$$

in such a way that the mapping

$$x \to P_j[\Gamma_j(\tau_j, \tau_{j+1}, x, \varepsilon_j, h_j^0)](\tau_{j+1}, x) \tag{1.99}$$

is *not* differentiable around x_j, maybe not even continuous. Furthermore, other adaptive components of the discretization algorithm, such as order selection or pivoting in linear systems solution, may introduce additional nondifferentiabilities or discontinuities. Such *external derivatives* of the whole integration process are therefore much less accurate than the integration itself; they may even be completely meaningless.

In order to obtain reliable derivatives we must differentiate the discretization scheme only, not the adaptive process. Considering finite differences again, the correct formula is

$$\frac{\partial F}{\partial x_j^i}(\tau_{j+1}, \tau_j, x_j) \approx \frac{P_j[\hat{\Gamma}_j](\tau_{j+1}, x_j + \delta_j^i e_i) - P_j[\hat{\Gamma}_j](\tau_{j+1}, x_j)}{\delta_j^i}, \tag{1.100}$$

where the nominal trajectory and all varied trajectories are computed on a common grid $\hat{\Gamma}_j$. (All the trajectories should be included in the stepsize selection here, so that the discretization approximates with the desired accuracies not only the solution but also its derivatives.)

This technique of *Internal Numerical Differentiation* (IND) has been developed by Bock [16]. It produces Wronskian matrices G_j that are *exact* derivatives of the approximate solution $p_j(\tau_{j+1}, x_j)$. In other words, IND generates *consistent* discretizations for Φ_g and Φ_g': the discretization of the derivative of Φ_g is the derivative of the discretization of Φ_g. Therefore, IND is stable in the sense of backward analysis and yields useful derivatives even for coarse approximation accuracies. Details for various integration methods and extensions to DAE can be found in [16, 17, 80, 21].

1.3.3 Collocation

Collocation, the second major BVP approach, differs from multiple shooting by the choice of local discretization schemes. On each subinterval (τ_j, τ_{j+1}), a polynomial p_j of a certain prescribed degree ℓ is taken as local representation of the trajectory. This polynomial must satisfy the differential equation in each node of the local grid, which consists of ℓ *collocation points*

$$\tau_j^i = \tau_j + \rho_i(\tau_{j+1} - \tau_j), \qquad 0 \leq \rho_1 < \cdots < \rho_\ell \leq 1. \tag{1.101}$$

Different classes of collocation methods arise according to the choice of ℓ and the distribution of collocation points.

Let us formally parameterize the polynomial $p_j(t, x_j, z_j)$ by its initial value x_j and ℓ derivatives z_j^i at the collocation points,

$$p_j(\tau_j, x_j, z_j) = x_j \tag{1.102}$$
$$\dot{p}_j(\tau_j^i, x_j, z_j) = z_j^i, \qquad i = 1(1)\ell. \tag{1.103}$$

Here the derivatives are explicitly introduced as *collocation variables* $z_j = (z_j^1, \ldots, z_j^\ell)$; they are implicitly determined by *collocation conditions*

$$c_j^i(x_j, z_j) = f(p_j(\tau_j^i, x_j, z_j)) - z_j^i = 0. \tag{1.104}$$

These conditions characterize p_j as approximate solution of the local IVP, thereby replacing the numerical integration.

In order to derive an explicit expression for p_j, we introduce a base of polynomials, $\{\psi_\nu\}_{\nu=0}^n$, defined by $\psi_0 \equiv 1$ and

$$\psi_\nu(0) = 0, \quad \dot{\psi}_\nu(\rho_i) = \delta_{\nu i}, \qquad \nu, i = 1(1)\ell. \tag{1.105}$$

This yields the formula

$$p_j(t, x_j, z_j) = x_j + (\tau_{j+1} - \tau_j) \sum_{\nu=1}^{\ell} \psi_\nu \left(\frac{t - \tau_j}{\tau_{j+1} - \tau_j} \right) z_j^\nu, \tag{1.106}$$

which reveals an important relationship between collocation and Runge-Kutta methods: evaluating $p_j(\tau_{j+1}, x_j, z_j)$ is equivalent to taking a single step of the implicit Runge-Kutta scheme

$$\begin{array}{c|ccc} \rho_1 & \psi_1(\rho_1) & \ldots & \psi_\ell(\rho_1) \\ \vdots & \vdots & & \vdots \\ \rho_\ell & \psi_1(\rho_\ell) & \ldots & \psi_\ell(\rho_\ell) \\ \hline & \psi_1(1) & \ldots & \psi_\ell(1) \end{array} \tag{1.107}$$

Therefore, (1.106) is called the Runge-Kutta representation of the collocation polynomial. For detailed information on collocation in BVP solution and the relation to Runge-Kutta schemes see, e.g., [73, 7, 9, 11, 8, 76, 10, 77].

Let us now write down the discrete BVP for collocation. When continuity conditions are defined in analogy to multiple shooting,

$$h_j(x_j, z_j, x_{j+1}) = p_j(\tau_{j+1}, x_j, z_j) - x_{j+1}, \tag{1.108}$$

it takes the form

$$\Phi_g(x_g) = \begin{pmatrix} c_1(x_1, z_1) \\ h_1(x_1, z_1, x_2) \\ \vdots \\ c_{m-1}(x_{m-1}, z_{m-1}) \\ h_{m-1}(x_{m-1}, z_{m-1}, x_m) \\ r(x_{\sigma(1)}, \ldots, x_{\sigma(k)}) \end{pmatrix}. \tag{1.109}$$

Here the global variable vector is

$$x_g = (x_1, z_1, \ldots, x_{m-1}, z_{m-1}, x_m) \in I\!R^N, \qquad N = [(m-1)\ell + m]n, \tag{1.110}$$

and each $c_j = (c_j^1, \ldots, c_j^\ell)$ collects all the local collocation conditions.

A Newton iteration on this BVP is equivalent to a single iteration on the implicit Runge-Kutta scheme (1.107) followed by one iteration on the continuity and boundary conditions. Hence, the collocation approach could be interpreted as a very special multiple shooting variant. However, local "integration" in collocation is part of the BVP solution process rather than an independent procedure. This causes some principal differences to multiple shooting. For instance, derivatives of the collocation polynomials p_j with respect to trajectory and collocation variables x_j, z_j can easily be computed analytically. In combination with automatic differentiation of the ODE right hand side f this yields highly accurate Wronskian matrices at low cost. Consistency of the linearization is always guaranteed. On the other hand, the fixed discretization scheme does not allow for adaptive local accuracy control. Accuracy is therefore controlled globally by the BVP solution algorithm through a grid strategy. From a theoretical viewpoint, the analytical representation of the trajectory permits a more detailed analysis of discretization errors, convergence properties, etc., see the references mentioned above.

Chapter 2

A structured direct approach

In contrast to indirect methods, which are based on optimality conditions for the infinite problem [OCP], direct methods discretize [OCP] first and then solve the resulting NLP. In order to obtain reliable numerical solutions it is important that the discrete problem reflects the structure of the original problem as well as possible. It is also important, for efficiency and stability, that a numerical method for the solution of the NLP exploits this structure.

Keeping this in mind, we will outline the direct SQP boundary value problem approach in this chapter. After having described the discretization process for multiple shooting and collocation methods, we present the resulting NLP along with an SQP method for its solution. Exploitation of the specific sparseness patterns of gradient and Hessian matrices in QP generation and solution are discussed. We conclude this chapter with a comparison of active set strategies and interior point methods for the treatment of inequality constraints. Both approaches lead to the same type of m-stage block-sparse KKT systems.

2.1 Direct BVP methods

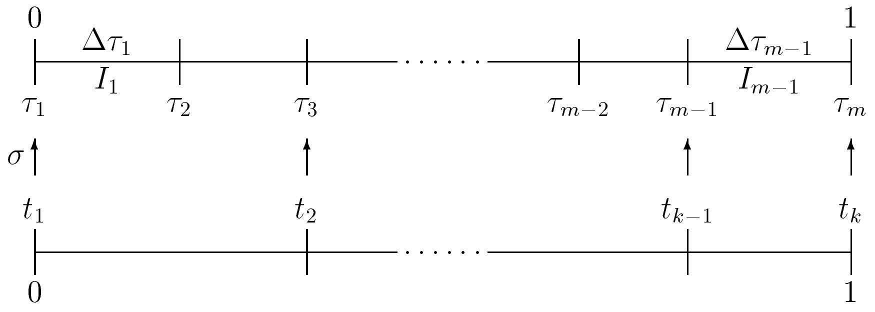
It is possible to extend the initial value problem approach for boundary value problems (cf. section 1.2) to optimization boundary value problems such as [OCP]. Any finite-dimensional parameterization of the control function can be used together with initial states in a discretization, provided there are enough variables to satisfy all the constraints. Such direct algorithms based on single shooting are described, for example, by Kraft [59] and Horn [48]. However, in the following we will consider the boundary value problem approach only, which is again superior with regard to efficiency, stability and robustness.

Discretizing [OCP] by multiple shooting or collocation proceeds as in section (1.3), except that control functions and path constraints must also be taken into account. After briefly having described the four basic steps involved in the discretization process we will give some more detailed explanations.

2.1.1 Discretization

1) Grid choice. In the first step a grid $\Gamma\colon 0 = \tau_1 < \cdots < \tau_m = 1$ with $m \geq k$ nodes is chosen as a refinement of the grid $0 = t_1 < \cdots < t_k = 1$ on which boundary and interior point conditions are specified. We denote subintervals by $I_j = (\tau_j, \tau_{j+1})$, their respective lengths by $\Delta\tau_j = \tau_{j+1} - \tau_j$, and the grid inclusion by $\sigma\colon \{1, \ldots, k\} \to \{1, \ldots, m\}$, $t_i = \tau_{\sigma(i)}$; see figure 2.1.

Figure 2.1: A typical grid



The cost function and boundary conditions are easily reformulated on such a grid. Using trajectory values $x_j = x(\tau_j)$ at the nodes, we get

$$\phi(x(t_k)) = \phi(x_m) = \min, \tag{2.1}$$

and

$$r(x(t_1), \ldots, x(t_k)) = r(x_{\sigma(1)}, \ldots, x_{\sigma(k)}) = 0. \tag{2.2}$$

For notational simplicity we will denote boundary conditions by $r(x_1, \ldots, x_m)$ in the following.

2) Control parameterization. Next, control functions are defined piece-wise by fixed *local* base functions v_j and free control parameters u_j,

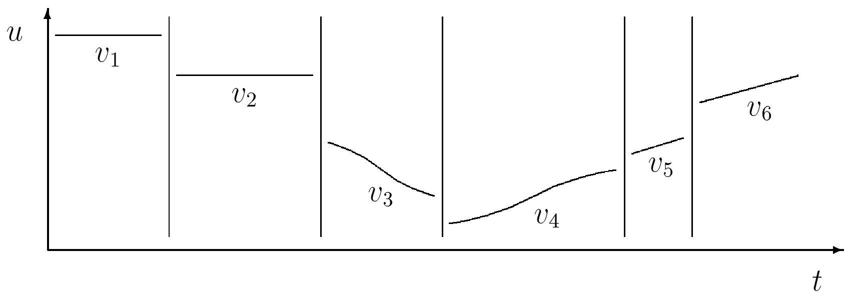
$$u(t) = v_j(t, u_j) \quad \text{on } I_j. \tag{2.3}$$

That is, u is restricted to some finite-dimensional space of admissible controls independently on each of the subintervals. The full control space is simply the cross product of local control spaces. Thus, u is allowed to jump at nodes τ_j as indicated in figure 2.2.

3) State discretization. As in section (1.3), we define local IVP solutions p_j and corresponding continuity conditions h_j. In multiple shooting these are given by

$$\begin{aligned} p_j(t, x_j, u_j) &= x_j + \int_{\tau_j}^{t} f(p_j(s, x_j, u_j), v_j(s, u_j))\, ds, && (2.4)\\ h_j(x_j, u_j, x_{j+1}) &= p_j(\tau_{j+1}, x_j, u_j) - x_{j+1} = 0. && (2.5) \end{aligned}$$

Figure 2.2: Sketch of a parameterized control function



In collocation, polynomials $p_j(t, x_j, z_j)$ enter into the collocation and continuity conditions

$$c_j^i(x_j, z_j, u_j) = z_j^i - f(p_j(\tau_j^i, x_j, z_j), v_j(\tau_j^i, u_j)) = 0, \tag{2.6}$$

$$h_j(x_j, z_j, x_{j+1}) = p_j(\tau_{j+1}, x_j, z_j) - x_{j+1} = 0. \tag{2.7}$$

Note that polynomials p_j and hence the continuity conditions depend on collocation variables z_j instead of u_j here; control variables appear in the collocation conditions only.

4) Constraint discretization. Substituting local state and control representations into the definition yields continuous path constraints

$$g(p_j(t, x_j, u_j), v_j(t, u_j)) \geq 0 \tag{2.8}$$

or

$$g(p_j(t, x_j, z_j), v_j(t, u_j)) \geq 0. \tag{2.9}$$

These have to be replaced by discrete versions

$$g_j(x_j, u_j) \geq 0. \tag{2.10}$$

Of course, it is in general impossible to choose the latter in such a way that they are exactly equivalent to (2.8,2.9) on the whole subinterval I_j. Therefore we simply check the constraints at nodes τ_j,

$$g_j(x_j, u_j) = g(x_j, v(\tau_j, u_j)), \tag{2.11}$$

and accept the possibility of small violations in the interior of the subintervals.

2.1.2 Remarks

- For simplicity we consider the grids for control parameterization and state discretization identical here, $\Gamma_u = \Gamma_x = \Gamma$. Note, however, that in multiple shooting each subinterval I_j might contain several control parameterization intervals, $\Gamma_x \subset \Gamma_u$. (This is formally equivalent to increasing the dimension of u_j.) Vice versa, a collocation grid might be chosen as a refinement of the control grid, $\Gamma_u \subset \Gamma_x$.

- If any of the points t_i were not included in the grid Γ, this would not only increase technical difficulties in implementing a numerical solution algorithm. Additional round-off errors as well as increased error propagation due to multiplication by the Wronskian $\partial x(t_i)/\partial x_j$ would inflict numerically computed derivatives of the boundary conditions with respect to local initial values x_j,

$$\frac{\partial r}{\partial x_j} = \frac{\partial r}{\partial x(t_i)}\frac{\partial x(t_i)}{\partial x_j}, \qquad x(t_i) = F(t_i, \tau_j, x_j, u_j). \tag{2.12}$$

  This may cause serious numerical difficulties in sensitive problems; for an extreme case see [74].

- Note that our local discretization schemes are *strictly decoupled* in the sense that no stage coupling terms are present in the definitions of p_j. Adjacent stages are connected to each other only by the continuity conditions (and possibly by boundary conditions, of course). Although this decoupling property is always satisfied in multiple shooting, it is not satisfied in some popular collocation schemes [47, 14, 92].

2.1.3 Local base functions

The characteristic property of direct methods is the use of parameterized control functions. Hence, the central question arises how a *good* parameterization should be chosen. As already indicated in section (1.1.2), base functions v_j must have certain differentiability properties so that higher order discretization schemes can be applied locally. On the other hand, the total number of control parameters must of course provide enough degrees of freedom to satisfy interior point and boundary conditions. One might therefore think that highly-dimensional local control spaces, such as higher order polynomials, are useful in order to obtain a good approximation to the true solution.

Instead, *very* simple base functions are usually preferable, and improved approximations should rather be produced by means of a refined grid, if necessary. Typical control representations are piecewise constant,

$$v_j(t, u_j) \equiv u_j, \tag{2.13}$$

or possibly piecewise linear,

$$v_j(t, u_j) = \frac{\tau_{j+1} - t}{\Delta\tau_j} u_j^l + \frac{t - \tau_j}{\Delta\tau_j} u_j^r, \tag{2.14}$$

where the control values at left and right ends of linear segments are chosen as parameters in the latter case, $u_j = (u_j^l, u_j^r)$.

Such simple base functions offer several advantages besides being easy to implement. Observe first that the control has by nature only a first order influence on the state history. Its effect averages out upon integration on each subinterval. Quite often a relatively coarse, piece-wise constant parameterization already allows rather accurate control over the state, and near-optimal solutions can be achieved. Higher order derivatives of the control function are not only of minor importance in the optimization, they also correspond to significantly smaller eigenvalues of the Hessian of the Lagrangian, and are therefore more difficult to determine numerically. Furthermore, fast control variations might introduce additional nonlinearities and artificial stiffness.

If a control component appears linearly in the right hand side, $f_{uu} \equiv 0$, the true optimal control typically shows bang-bang behavior, i.e., u jumps between upper and lower bounds and remains constant between switching points. Obviously, piece-wise constant control functions are ideally suited for this special, but very important problem class. The quality of the parameterization is determined by the accuracy of switching point approximation, in other words, by the grid resolution.

Finally, box control constraints can be treated exactly with constant or linear base functions, since minimum and maximum values are explicitly given by the control parameters.

2.2 An SQP method for the structured NLP

2.2.1 Formulation of the NLP

In order to obtain a unified formulation for both multiple shooting and collocation, we collect local variables in each discretization stage,

$$y_j = (x_j, z_j, u_j) \in I\!R^{n_j}, \qquad n_j = n_j^x + n_j^z + n_j^u, \tag{2.15}$$

and define the global variable vector

$$y = (y_1, \dots, y_m) \in I\!R^N, \qquad N = \sum n_j. \tag{2.16}$$

(Variables z_m, u_m are never present, $n_m^z = n_m^u = 0$, since there is no interval attached to the final node τ_m, and collocation variables z_j are of course not needed in multiple shooting, hence $n_j^z = 0$.) A reformulation of the discrete problem in these variables leads to the cost function

$$\phi(x_m) = \min, \tag{2.17}$$

continuity conditions

$$h_j(y_j, x_{j+1}) = p_j(y_j) - x_{j+1} = 0, \tag{2.18}$$

local equality constraints

$$e_j(y_j) = \begin{pmatrix} r_j^0(x_j) \\ c_j(y_j) \end{pmatrix} = 0, \tag{2.19}$$

global equality constraints

$$e(y) = r_1^1(x_1) + \cdots + r_m^1(x_m) = 0, \tag{2.20}$$

and path inequality constraints

$$g_j(y_j) = g_j(x_j, u_j) \geq 0. \tag{2.21}$$

Here p_j, h_j are independent of u_j in the collocation approach, and independent of z_j in multiple shooting. All the collocation conditions c_j are simply not present in the latter case (c_m never is), and each of the $r_j^i(x_j)$ may or may not be present depending on the actual structure of boundary conditions. Here and in the following we assume second order decoupling of the boundary conditions.

The discrete optimal control problem can now be stated as a standard nonlinear programming problem

$$[\text{NLP}] \qquad \begin{aligned} & & F_1(y) &= \min & (2.22)\\ &\text{subject to} & F_2(y) &= 0, & (2.23)\\ & & F_3(y) &\geq 0, & (2.24)\end{aligned}$$

where we have defined the cost function

$$F_1(y) = \phi(x_m), \tag{2.25}$$

equality constraints

$$F_2(y) = \begin{pmatrix} e_1(y_1) \\ h_1(y_1, x_2) \\ \vdots \\ e_{m-1}(y_{m-1}) \\ h_{m-1}(y_{m-1}, x_m) \\ e_m(y_m) \\ e(y) \end{pmatrix}, \tag{2.26}$$

and inequality constraints

$$F_3(y) = \begin{pmatrix} g_1(y_1) \\ \vdots \\ g_m(y_m) \end{pmatrix}. \tag{2.27}$$

A similar partitioning of the dual variables, $w_2 = (w_{21}, \ldots, w_{2m}, w_2^e)$, $w_{2j} = (w_{2j}^e, w_{2j}^h)$, and $w_3 = (w_{31}, \ldots, w_{3m})$, yields the Lagrangian

$$\begin{aligned} L(y, w_2, w_3) &= F_1(y) - w_2^T F_2(y) - w_3^T F_3(y) \\ &= \phi(x_m) - \sum_{j=1}^{m} w_{2j}^T \begin{pmatrix} e_j(y_j) \\ h_j(y_j, x_{j+1}) \end{pmatrix} - w_2^{eT} e(y) \qquad (2.28) \\ &\quad - \sum_{j=1}^{m} w_{3j}^T g_j(y_j). \end{aligned}$$

2.2.2 SQP method

For the numerical solution of [NLP] we apply an SQP iteration, $y^{k+1} = y^k + \alpha^k \Delta y^k$. Each search direction Δy^k is obtained as solution of a quadratic subproblem with linear constraints,

$$[\mathrm{QP}(y^k)] \qquad \begin{aligned} \frac{1}{2}\Delta y^T H^k \Delta y + J_1^k \Delta y &= \min_{\Delta y} && (2.29)\\ \text{subject to} \qquad J_2^k \Delta y + F_2^k &= 0, && (2.30)\\ J_3^k \Delta y + F_3^k &\geq 0, && (2.31) \end{aligned}$$

where $F_i^k = F_i(y^k)$ and $J_i^k = F_i'(y^k)$ denote function and Jacobian values, respectively, and $H^k \approx D_{yy}^2 L(y^k, w_2^k, w_3^k)$ approximates the Hessian of the Lagrangian.

This QP serves as a local approximation of the nonlinear problem [NLP] around iterate y^k. If H^k is the exact Hessian, $[\mathrm{QP}(y)]$ contains precisely the data that are needed to test first and second order optimality conditions. It is easily verified that in this case (y, w_2, w_3) is a KKT point of [NLP] if and only if $(0, w_2, w_3)$ is a KKT point of $[\mathrm{QP}(y)]$, and that a strict local minimum of $[\mathrm{QP}(y)]$ with zero increment $\Delta y = 0$ implies that y is a strict local minimum of [NLP]. The special structure of our QP is established in the following theorem.

Theorem 9 (QP structure) *The Hessian H^k and the Jacobian J_3^k are* block-diagonal; *the Jacobian J_2^k is almost block-diagonal, except for one upper diagonal and an additional full row of blocks. More precisely, partitionings can be written*

$$H^k = \begin{pmatrix} H_1^k & & \\ & \ddots & \\ & & H_m^k \end{pmatrix}, \qquad H_j^k = \begin{pmatrix} H_j^{xx} & H_j^{xz} & H_j^{xu} \\ H_j^{zx} & H_j^{zz} & H_j^{zu} \\ H_j^{ux} & H_j^{uz} & H_j^{uu} \end{pmatrix} = (H_j^k)^T, \tag{2.32}$$

where superscripts x, z, u indicate derivatives with respect to x_j, z_j, u_j, respectively, and

$$J_2^k = \begin{pmatrix} C_1^k & P_2^k & & \\ & \ddots & \ddots & \\ & & \ddots & P_m^k \\ & & & C_m^k \\ F_1^k & \dots & \dots & F_m^k \end{pmatrix}, \qquad J_3^k = \begin{pmatrix} T_1^k & & \\ & \ddots & \\ & & T_m^k \end{pmatrix}, \tag{2.33}$$

where the Jacobian blocks are further partitioned according to (2.25–2.28),

$$C_j^k = \frac{\partial}{\partial y_j}\begin{pmatrix} r_j^0 \\ c_j \\ h_j \end{pmatrix} = \begin{pmatrix} F_j^0 & 0 & 0 \\ C_j^x & C_j^z & C_j^u \\ G_j^x & G_j^z & G_j^u \end{pmatrix}, \qquad P_{j+1}^k = \frac{\partial}{\partial y_{j+1}}\begin{pmatrix} r_j^0 \\ c_j \\ h_j \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -I & 0 & 0 \end{pmatrix}, \tag{2.34}$$

$$F_j^k = \partial r_j^1 / \partial y_j = \begin{pmatrix} F_j^1 & 0 & 0 \end{pmatrix}, \qquad T_j^k = \partial g_j / \partial y_j = \begin{pmatrix} T_j^x & 0 & T_j^u \end{pmatrix}. \tag{2.35}$$

Proof. This follows immediately from the definitions of problem functions F_i and constituents ϕ, e_j, h_j, e, g_j. Stage-coupling terms appear only in the continuity conditions h_j and global boundary conditions e; they produce the off-diagonal blocks P_j^k and F_j^k in J_2^k, respectively. By definition and by assumption these coupling terms in h_j and e are linear. Hence, the Hessian $H^k = D_{yy}^2 L(y^k, w_2^k, w_3^k)$ is block-diagonal as a linear combination of second derivatives of ϕ, e_j, h_j, e, g_j, see (2.28). ∎

2.2.3 Local condensing for collocation

Recall that the local discretization scheme in collocation is solved as part of the global NLP, and hence in each SQP iteration as part of the QP. Compared to multiple shooting on the same global grid, the QP is therefore larger, but it has a more specific structure. More precisely, the blocks sitting at the corners of derivative matrices $H_j^k, C_j^k, P_j^k, F_j^k, T_j^k$ are also present in multiple shooting, whereas all the central blocks, being associated with collocation variables and conditions, belong solely to the local discretization. In the following we will show how collocation variables and conditions are locally eliminated, leaving a condensed QP which has exactly the same structure as in multiple shooting.

Let us first take a closer look at the derivatives of collocation and continuity conditions. Substitution of the explicit representation (1.106) of collocation polynomials p_j into the collocation conditions yields

$$c_j^i(y_j) = z_j^i - f(x_j + \Delta\tau_j \textstyle\sum_\nu \psi_\nu(\rho_i) z_j^\nu, v_j(\tau_j^i, u_j)) \tag{2.36}$$

and

$$C_j^x = -\begin{pmatrix} f_x(p_j(\tau_j^1, x_j, z_j), v_j(\tau_j^1, u_j)) \\ \vdots \\ f_x(p_j(\tau_j^\ell, x_j, z_j), v_j(\tau_j^\ell, u_j)) \end{pmatrix} = \begin{pmatrix} C_j^{1x} \\ \vdots \\ C_j^{\ell x} \end{pmatrix}, \tag{2.37}$$

$$C_j^z = \begin{pmatrix} I & & \\ & \ddots & \\ & & I \end{pmatrix} + \Delta\tau_j \begin{pmatrix} \psi_1(\rho_1) C_j^{1x} & \dots & \psi_\ell(\rho_1) C_j^{1x} \\ \vdots & & \vdots \\ \psi_1(\rho_\ell) C_j^{\ell x} & \dots & \psi_\ell(\rho_\ell) C_j^{\ell x} \end{pmatrix}, \tag{2.38}$$

$$C_j^u = -\begin{pmatrix} f_u(p_j(\tau_j^1, x_j, z_j), v_j(\tau_j^1, u_j)) v_{ju}(\tau_j^1, u_j) \\ \vdots \\ f_u(p_j(\tau_j^\ell, x_j, z_j), v_j(\tau_j^\ell, u_j)) v_{ju}(\tau_j^\ell, u_j) \end{pmatrix}, \tag{2.39}$$

where v_{ju} denotes the partial derivative with respect to u_j. Analogously, substitution into the continuity conditions yields

$$h_j(y_j, x_{j+1}) = x_j + \Delta\tau_j \textstyle\sum_\nu \psi_\nu(1) z_j^\nu - x_{j+1} \tag{2.40}$$

and derivative matrices

$$G_j^x = I, \tag{2.41}$$

$$G_j^z = \Delta\tau_j \Big(\psi_1(1) I \ \dots \ \psi_\ell(1) I\Big), \tag{2.42}$$

$$G_j^u = 0. \tag{2.43}$$

We assume without loss of generality that $C_j^z = I+\mathcal{O}(\Delta\tau_j)$ is non-singular. (Otherwise a finer grid must be chosen.) Collocation variables can then be expressed in terms of state and control variables,

$$\Delta z_j = -(C_j^z)^{-1}[C_j^x \Delta x_j + C_j^u \Delta u_j + c_j]. \tag{2.44}$$

Resubstitution into the QP formally eliminates collocation variables and conditions, leaving the desired QP with multiple shooting structure. The elimination is equivalent to a projection of the problem onto $\ker(C_j^x\ C_j^z\ C_j^u)$.

In practice the condensing may be performed as follows. First we compute factorizations $C_j^z = L_j U_j$ and solve for

$$\begin{pmatrix}\bar C_j^x & \bar C_j^u & \bar c_j\end{pmatrix} = L_j^{-1}\begin{pmatrix}C_j^x & C_j^u & c_j\end{pmatrix}, \tag{2.45}$$

which yields the equation

$$\Delta z_j = -U_j^{-1}\Delta \bar z_j = -U_j^{-1}[\bar C_j^x \Delta x_j + \bar C_j^u \Delta u_j + \bar c_j]. \tag{2.46}$$

For the QP transformation we compute matrices

$$\bar G_j^z = G_j^z U_j^{-1} \tag{2.47}$$

and

$$\begin{aligned}
\bar H_j^{xz} &= H_j^{xz} U_j^{-1} = {}^t\bar H_j^{zx}, && (2.48)\\
\bar H_j^{zz} &= {}^tU_j^{-1} H_j^{zz} U_j^{-1} = {}^t\bar H_j^{zz}, && (2.49)\\
\bar H_j^{uz} &= H_j^{uz} U_j^{-1} = {}^t\bar H_j^{zu}. && (2.50)
\end{aligned}$$

(Here pre-superscript ^{t} denotes the transpose of a matrix; we will use this notation within larger matrices and when post-superscripts are present.) Hessian blocks and continuity conditions are finally updated according to

$$\begin{aligned}
\begin{pmatrix}\bar H_j^{xx} & \bar H_j^{xu}\\ \bar H_j^{ux} & \bar H_j^{uu}\end{pmatrix} &= \begin{pmatrix}H_j^{xx} & H_j^{xu}\\ H_j^{ux} & H_j^{uu}\end{pmatrix} + \begin{pmatrix}{}^t\bar C_j^x\\ {}^t\bar C_j^u\end{pmatrix}\bar H_j^{zz}\begin{pmatrix}\bar C_j^x & \bar C_j^u\end{pmatrix}\\
&\quad - \left[\begin{pmatrix}\bar H_j^{xz}\\ \bar H_j^{uz}\end{pmatrix}\begin{pmatrix}\bar C_j^x & \bar C_j^u\end{pmatrix} + \begin{pmatrix}{}^t\bar C_j^x\\ {}^t\bar C_j^u\end{pmatrix}\begin{pmatrix}\bar H_j^{zx} & \bar H_j^{zu}\end{pmatrix}\right], && (2.51)\\
\begin{pmatrix}\bar G_j^x & \bar G_j^u & \bar h_j\end{pmatrix} &= \begin{pmatrix}G_j^x & G_j^u & h_j\end{pmatrix} - \bar G_j^z\begin{pmatrix}\bar C_j^x & \bar C_j^u & \bar c_j\end{pmatrix}. && (2.52)
\end{aligned}$$

The (local) gradient vectors appearing in the minimization condition are transformed accordingly; all the matrices and vectors in boundary conditions and inequality constraints remain unchanged. Note that the condensing is strictly local with respect to the grid intervals. Therefore it can be performed in parallel without any data exchange.

2.2.4 Remarks on structured SQP solution

- It is well-known that certain collocation schemes satisfy some of the nonlinear collocation and/or continuity conditions automatically. That is, these conditions and corresponding variables are *implicitly* eliminated. They do not even appear in the NLP, thus leading to savings in storage requirements as well as computational effort when generating the QP data. A widely used approach, e.g., applies continuous, piecewise linear control parameterizations and cubic collocation at Lobatto points. The trajectory approximation is then always globally continuous, and two out of three collocation conditions are automatically satisfied [47, 14, 92].

  However, we strongly recommend *not* to eliminate nonlinear conditions implicitly in the NLP formulation. This may increase nonlinearities in the NLP and, especially in the case of enforced continuity, severely influence the convergence behavior. Instead, our strategy is to include all conditions explicitly and defer any eliminations to the QP solution, thereby moving computational effort from nonlinear, iterative computations to linear systems solution. This strategy proves very efficient for difficult problems, and especially on modern computer architectures which nearly reach peak performance in linear algebra computations. On the other hand it calls for highly efficient QP solvers; otherwise QP solution will become a bottleneck.

- An alternative to local projections is the *direct* computation of the projected QP in *reduced SQP methods* [61, 77]. Instead of performing the local condensing described above, for instance, only the collocation conditions are differentiated with respect to collocation variables; directional derivatives of all the remaining restrictions are generated only within the subspace $\ker(C_j^x \ C_j^z \ C_j^u)$. The same technique can be applied to local boundary or interior conditions as well. This reduces the computational effort and the required amount of storage, especially for the Hessian blocks, see [77].

- Obviously, the decoupling in our approach makes a significant amount of computations completely independent. This is true for generating function values, gradient and Hessian matrices on each of the grid intervals, no matter whether derivatives are computed by finite differences or automatic differentiation. It is also true for the local condensing as well as for the direct projection in the reduced SQP method according to [77]. Hence, all the computations for generating the projected QP can be performed in parallel without any data exchange.

- In most SQP methods the Hessian is approximated according to certain rank-2 update formulae, $H^{k+1} = H^k + \alpha pp^T + \beta qq^T$. The exact Hessian is often unavailable, too expensive, or indefinite. In our case rank-2 BFGS updates are applied *block-wise*, yielding a global rank-$2m$ update. This leads to superlinear SQP convergence, where the convergence order is expected to be essentially grid-independent. For details see [69, 77].

2.3 Constrained QP subproblems and KKT systems

From now on we assume that collocation variables and conditions have been eliminated so that the QP has the basic structure arising in multiple shooting. In the following we will consider a single QP subproblem in the SQP iteration. In order to simplify notation, superscripts are dropped and vectors and matrix blocks are renamed. Furthermore, path constraints are interleaved with local equality constraints. The QP then fits into the more general m-stage block-sparse structure

$$\begin{aligned}
\sum_{j=1}^{m}\left[\frac{1}{2}\begin{pmatrix}\Delta x_j\\ \Delta u_j\end{pmatrix}^T\begin{pmatrix}H_j & {}^tJ_j\\ J_j & K_j\end{pmatrix}\begin{pmatrix}\Delta x_j\\ \Delta u_j\end{pmatrix}+\begin{pmatrix}f_j\\ d_j\end{pmatrix}^T\begin{pmatrix}\Delta x_j\\ \Delta u_j\end{pmatrix}\right] &= \min_{\Delta x,\Delta u} && (2.53)\\
\text{subject to}\qquad \begin{pmatrix}G_j & E_j\end{pmatrix}\begin{pmatrix}\Delta x_j\\ \Delta u_j\end{pmatrix}+h_j-x_{j+1} &= 0, && (2.54)\\
F_j^x\Delta x_j+e_j^x &= 0 \text{ or } \geq 0, && (2.55)\\
D_j^u\Delta u_j+e_j^u &= 0 \text{ or } \geq 0, && (2.56)\\
\begin{pmatrix}F_j^c & D_j^c\end{pmatrix}\begin{pmatrix}\Delta x_j\\ \Delta u_j\end{pmatrix}+e_j^c &= 0 \text{ or } \geq 0, && (2.57)\\
\sum_{j=1}^{m}\begin{pmatrix}F_j & D_j\end{pmatrix}\begin{pmatrix}\Delta x_j\\ \Delta u_j\end{pmatrix}+e_* &= 0. && (2.58)
\end{aligned}$$

Here the Hessian blocks H_j, J_j, K_j correspond to $H_j^{xx}, H_j^{ux}, H_j^{uu}$ in theorem 9, blocks G_j^x, G_j^u are renamed as G_j, E_j, and the constraint blocks F_j^0, F_j^1 appear as F_j^x and F_j. Finally, rows of T_j^x, T_j^u are split up between F_j^x, D_j^u and (F_j^c, D_j^c) according to whether the corresponding inequality involves only states, controls, or both. In addition to the minimization and continuity conditions we distinguish four categories of restrictions: local state constraints (2.55), local control constraints (2.56), locally coupled constraints (2.57), and globally coupled constraints (2.58). The four latter categories permit any possible combination of multipoint boundary conditions and path constraints. Although all of them could be subsumed under categories (2.57,2.58) or even (2.58) only, the distinction is made in order to exploit different levels of decoupling.

The feasible set of the QP is a polyhedron whose boundary is determined by the inequalities, the whole polyhedron being contained in a linear subspace given by the equality constraints. We assume for simplicity that the restricted quadratic cost function has a unique minimum. This minimum is typically attained at a corner of the polyhedron, possibly also on a higher-dimensional face.

An interior iteration (within the SQP iteration) is applied which reduces the solution of the *inequality*-constrained QP to the solution of a sequence of symmetric, indefinite *linear* equation systems

$$\Omega z + a = \begin{pmatrix}H & C^T\\ C & \end{pmatrix}\begin{pmatrix}y\\ -\eta\end{pmatrix}+\begin{pmatrix}f\\ c\end{pmatrix}=0, \qquad (2.59)$$

each of which represents KKT conditions for a purely *equality*-constrained QP,

$$\frac{1}{2} y^T H y + f^T y \quad = \quad \min \tag{2.60}$$

$$\text{subject to} \qquad Cy + c \quad = \quad 0. \tag{2.61}$$

Efficient solution of the KKT systems (2.59) with a substructure according to (2.53–2.58) is the core problem of our structured SQP approach; a recursive $\mathcal{O}(m)$ algorithm for this purpose will be developed in the following chapter.

There are two principal techniques for reducing the inequality-constrained QP to a sequence of equality-constrained ones: active set strategies and interior point methods; both are applicable here. Since every path constraint appears in each stage of the discrete problem, their total number grows linearly with m (whereas the number of multipoint boundary conditions is fixed for a specific problem). Efficient handling of so many inequalities remains a central difficulty in solving KKT system (2.59). Our $\mathcal{O}(m)$ algorithm relies on the local nature of discrete path constraints, which is preserved in both active set strategies and interior point methods. A brief summary of the characteristic properties of these approaches follows.

Active Set Strategy. Starting at a feasible point, an active set strategy determines a sequence of iterates on the *boundary* of the feasible polyhedron. The original Hessian H remains unchanged throughout the iteration, whereas constraint matrices C differ according to the currently active set. At each iterate a search direction is computed as solution of the current KKT system, and the step-size is determined by the first (inactive) inequality to become active along the search direction. If a full step is taken, one of the active restrictions becomes inactive, or else the active set is correct and a KKT point of the QP is found.

Under our uniqueness assumption this KKT point is the QP solution, and the active set strategy is guaranteed to terminate after finitely many steps. However, the number of steps may grow exponentially with the dimension of the problem. Furthermore, the accuracy of the QP solution is not controllable: there is no way to tell an iterate's distance to the solution unless the correct active set is identified, i.e., until the solution is found.

Interior Point Method. Starting at a *strictly* feasible point, interior point methods generate a sequence of iterates in the *interior* of the feasible polyhedron. The constraint matrix C includes original equalities only, whereas the Hessian H changes between iterates according to a weighted logarithmic barrier potential term in the cost function. The barrier potential always keeps the unconstrained KKT system solutions away from the boundary, where it becomes infinite. Its weight parameter is iteratively decreased in order to blend out the barrier potential as the iterates approach the solution of the original QP.

Although the solution is never attained exactly, this process converges rapidly. The number of iterations can be shown to grow only linearly with the square root of the problem dimension and with the number of digits gained in accuracy. Hence, QP solution accuracy can be adapted to the SQP needs in order to save computational effort.

Chapter 3

Efficient solution of structured KKT systems

In this chapter we treat the purely equality-constrained QP

$$\begin{aligned} \frac{1}{2}y^T H y + f^T y &= \min && (3.1)\\ \text{subject to} \qquad Cy + c &= 0, && (3.2) \end{aligned}$$

and its associated KKT system

$$\Omega z + a = \begin{pmatrix} H & C^T \\ C & \end{pmatrix} \begin{pmatrix} y \\ -\eta \end{pmatrix} + \begin{pmatrix} f \\ c \end{pmatrix} = 0. \qquad (3.3)$$

Unless otherwise stated we require as a *global regularity assumption* that C has full rank and H is positive definite on $\ker(C)$. Under these conditions (3.3) has a unique solution $z = (y, -\eta)$ for any right hand side, and y is the unique minimizer of the QP (3.1,3.2). We denote by n_v the number of optimization variables, y, and by n_c the number of constraints (or multipliers, η), giving a total of $n = n_v + n_c$ variables in the KKT system. Assume that $n_c < n_v$, i.e., y is not completely determined by the constraint equation.

After introducing the notion of elementary matrices we will present standard factorizations of the KKT matrix Ω, which exploit its optimization superstructure in solving equation (3.3), but treat H and C as dense blocks. These factorizations will be used as building blocks in developing a recursive $\mathcal{O}(m)$ KKT solver for the m-stage block-sparse substructure established in chapter 2. The recursive algorithm exploits every detail of the structure. It is therefore very fast, and offers additional advantages as will be seen below. For the special case of *separated* constraints, which cause a natural *banded* block structure of Ω, a family of alternative $\mathcal{O}(m)$ solution algorithms will finally be developed. These algorithms differ in the degree of structure exploitation, and accordingly in the degrees of freedom for pivoting. As compared to the recursive algorithm which allows pivoting on the block level only, they offer gradually increased levels of robustness at the cost of slightly decreased efficiency.

3.1 Dense KKT solution methods

Standard solution techniques for the symmetric, indefinite KKT system include the range space or *Schur complement* (SC) method and null space or *projected Hessian* (PH) method. Both techniques exploit the symmetry and cope with the indefiniteness of the system matrix, but both destroy the structure when applied to m-stage block-sparse systems with coupled constraints, leading to an $\mathcal{O}(m^3)$ operation count. Structured $\mathcal{O}(m)$ variants of these methods for *linearly decoupled* m-stage block-sparse systems are developed in section 3.3.

3.1.1 Factorizations and elementary matrices

Solving a linear equation always involves the inverse (or pseudo-inverse) of the system matrix. However, in practice one is usually not interested in the inverse itself but in its product with some right hand side. Therefore most numerical solution algorithms do not generate the inverse explicitly. The system matrix is rather decomposed into elementary factors, so that the product of the inverse of each factor with a vector (or matrix) can easily be computed. In our algorithms we will make use of the following elementary factors.

Definition 10 (Elementary matrices) *Non-singular matrices that admit a partitioning according to one of the following schemes will be referred to as* elementary matrices.

(a) Matrices that differ from the unit matrix only in a diagonal block,

$$A = \begin{pmatrix} I & & \\ & U & \\ & & I \end{pmatrix}, \tag{3.4}$$

where U is either lower or upper triangular, or orthogonal. Each of the unit blocks may or may not be present.

(b) Matrices differing from the unit matrix only in an off-diagonal block,

$$A = \begin{pmatrix} I & \\ C & I \end{pmatrix} \quad \textit{or} \quad A = \begin{pmatrix} I & C \\ & I \end{pmatrix}, \tag{3.5}$$

where C is any rectangular matrix.

(c) Matrices with unit blocks along the counter-diagonal and one off-diagonal block,

$$A = \begin{pmatrix} C & I \\ I & \end{pmatrix} \quad \textit{or} \quad A = \begin{pmatrix} & I \\ I & C \end{pmatrix}. \tag{3.6}$$

Lemma 1 *Inverses of the previously defined elementary matrices are*

(a)

$$\begin{pmatrix} I & & \\ & U & \\ & & I \end{pmatrix}^{-1} = \begin{pmatrix} I & & \\ & U^{-1} & \\ & & I \end{pmatrix}, \tag{3.7}$$

(b)

$$\begin{pmatrix} I & \\ C & I \end{pmatrix}^{-1} = \begin{pmatrix} I & \\ -C & I \end{pmatrix}, \qquad \begin{pmatrix} I & C \\ & I \end{pmatrix}^{-1} = \begin{pmatrix} I & -C \\ & I \end{pmatrix}, \tag{3.8}$$

(c)

$$\begin{pmatrix} C & I \\ I & \end{pmatrix}^{-1} = \begin{pmatrix} & I \\ I & -C \end{pmatrix}. \tag{3.9}$$

Proof. This is immediately verified. ∎

Remarks.

- In (a), triangular matrices include the special case of diagonal ones, and orthogonal matrices include permutation and reflection matrices as special cases. Nontrivial orthogonal matrices, as appearing in Householder factorizations, are represented as a product of elementary reflectors; their inverses are products of the *same* factors in reverse order (see, e.g., [4]).

- Type (b) matrices are block Frobenius matrices, including usual Frobenius matrices which have only a single nonzero column below the diagonal. Unit triangular matrices, as appearing in the Gauß factorization, can be represented as a product of Frobenius matrices (see, e.g., [86, Chapter 4.1]).

- Type (c) matrices are products of a type (b) matrix and a type (a) permutation matrix that exchanges rows of blocks. They will appear as factors in symmetric decompositions of indefinite matrices; we regard them as elementary matrices for symmetry reasons.

- Multiplication of an elementary matrix or its inverse by a vector (matrix) corresponds to a single BLAS level 2 (level 3) call, or an LAPACK subroutine call in the case of orthogonal matrices.

Dense matrices are usually decomposed into two factors of type (a) with none of the unit blocks present (and possibly a third factor which is diagonal or represents a permutation due to pivoting). Solving the system $Ax = LUx = b$ therefore requires a *transformation* step for the right hand side, $y = L^{-1}b$, and a *solution* step, $x = U^{-1}y$. Below we will develop recursive decompositions consisting of many factors, where the transformation and solution steps are recursively nested, $A = LU = L_1 \dots L_m U_m \dots U_1$.

3.1.2 The Schur complement method

The Schur complement method, which is also called *range space method*, generates a solution of the KKT system (3.3) as follows. First, y is formally eliminated using the first equation,

$$y = -H^{-1}(f + C^T(-\lambda)). \tag{3.10}$$

Substitution into the second equation yields an explicit expression for λ,

$$\lambda = -(CH^{-1}C^T)^{-1}(c - CH^{-1}f). \tag{3.11}$$

Finally, λ is inserted into (3.10) to actually compute y. The matrix $W = -CH^{-1}C^T$ is called the *Schur complement* of H. Note that the SC method requires to compute the multiplier λ before the QP solution y is obtained.

Since H and W must be invertible we require that (a) the Hessian is positive definite, $H > 0$, and (b) the constraint matrix has full rank, $CC^T > 0$. Under these regularity assumptions the Schur complement is negative definite, $W = -CH^{-1}C^T < 0$, and y is the unique minimizer of QP (3.1,3.2).

The numerical factorization of Ω proceeds in three steps, involving two symmetric (Cholesky) factorizations. (In the following, a colon next to an equals sign indicates which side of the equation is newly defined.)

(1) Factorization of the Hessian, $H =: L_1 L_1^T$.
Transformation of the constraint matrix, $\bar{C} := CL_1^{-T}$.

(2) Computation of the Schur complement, $-W := \bar{C}\bar{C}^T$.

(3) Factorization of the Schur complement, $-W =: L_2 L_2^T$.

These steps correspond to a symmetric decomposition of Ω into elementary factors,

$$\Omega = \Lambda_1 \Lambda_2 \Lambda_3 S \Lambda_3^T \Lambda_2^T \Lambda_1^T, \tag{3.12}$$

where

$$\Lambda_1 \Lambda_2 \Lambda_3 = \begin{pmatrix} L_1 & \\ & I \end{pmatrix} \begin{pmatrix} I & \\ \bar{C} & I \end{pmatrix} \begin{pmatrix} I & \\ & L_2 \end{pmatrix} \tag{3.13}$$

and

$$S = \begin{pmatrix} I & \\ & -I \end{pmatrix} = S^{-1}. \tag{3.14}$$

The inverse of the KKT matrix,

$$\Omega^{-1} = \Lambda_1^{-T} \Lambda_2^{-T} \Lambda_3^{-T} S \Lambda_3^{-1} \Lambda_2^{-1} \Lambda_1^{-1}, \tag{3.15}$$

has an explicit representation in terms of H and C only ($W \equiv -CH^{-1}C^T$),

$$\begin{pmatrix} H & C^T \\ C & \end{pmatrix}^{-1} = \begin{pmatrix} H^{-1} + H^{-1}C^T W^{-1} C H^{-1} & -H^{-1}C^T W^{-1} \\ -W^{-1} C H^{-1} & W^{-1} \end{pmatrix}. \tag{3.16}$$

When the KKT system is solved for a right hand side, the transformation phase involves multiplication by $\Lambda_3^{-1}\Lambda_2^{-1}\Lambda_1^{-1}$ according to (3.15), and the solution phase involves multiplication by $\Lambda_1^{-T}\Lambda_2^{-T}\Lambda_3^{-T}$. Multiplication by S may be included in either phase. The individual operations of each phase are of course closely related. They are put together in table 3.1, giving a complete description of the SC method.

Table 3.1: Operations of the SC method

Step	Factorization	Transformation	Solution
(1a)	$H =: L_1 L_1^T$		
(1b)	$\bar{C} := C L_1^{-T}$	$\bar{f} := L_1^{-1}(-f)$	$y := L_1^{-T}\bar{y}$
(2)	$-W := \bar{C}\bar{C}^T$	$\bar{c} := -\bar{C}\bar{f} + (-c)$	$\bar{y} := -\bar{C}^T(-\lambda) + \bar{f}$
(3a)	$-W =: L_2 L_2^T$		
(3b)		$\bar{\lambda} := L_2^{-1}\bar{c}$	$-\lambda := L_2^{-T}(-\bar{\lambda})$

Operations of the factorization and transformation phases proceed from top to bottom, 1a–3b, whereas operations of the solution phase proceed in reverse order. Step i describes generation of Λ_i and multiplications by $\Lambda_i^{-1}, \Lambda_i^{-T}$, respectively. Multiplication by S is indicated by the sign change $\bar{\lambda} \to -\bar{\lambda}$ in (3b). Note that the SC method is applied to the negative right hand side $-a = -(f, c)$ in order to obtain the solution $z = \Omega^{-1}(-a)$.

The Schur complement method exploits the symmetry of Ω as well as the known indefinite structure (i. e., the positive and negative eigenspaces corresponding to x and λ, respectively). Only the lower triangle of H needs to be stored, and H, C are replaced by $L_1, \bar{C}$ when being factorized. No additional storage is needed for the Schur complement W: its upper triangle fits into the free part of the array holding H, and the Cholesky factor L_2^T occupies the same space after the factorization. Of course, square root-free LDL^T factorizations may be applied to H and $-W$ as an alternative.

A close examination shows that the SC factorization works exactly like a (block-wise) Cholesky factorization, where the partitioning of Ω corresponds to positive and negative eigenspaces. The algorithm can be extended to general block-partitioned indefinite systems

$$\tilde{\Omega}z + a = \begin{pmatrix} H & C^T \\ C & -M \end{pmatrix} \begin{pmatrix} y \\ -\lambda \end{pmatrix} = 0 \tag{3.17}$$

if we redefine the Schur complement as $W := -(M + CH^{-1}C^T)$. Step (2) of the factorization algorithm then reads $-W = \bar{C}\bar{C}^T + M$. Furthermore we must replace regularity assumption (b) $CC^T > 0$ by (b′) $W < 0$, or equivalently, $CH^{-1}C^T > -M$. This is always true if $-M$ is negative definite, and it follows from (b) if $-M$ is negative semi-definite.

One should note that the SC method is potentially unstable: since H is always used as a block pivot, large errors in the solution may occur even if the system is well conditioned.

3.1.3 The projected Hessian method

The projected Hessian method, which is also known as *reduced Hessian* or *null space* method, is based on the observation that the feasible set of the QP is an affine subspace, and the cost function restricted to this subspace is again quadratic. Hence, the solution can be obtained by unconstrained minimization on this lower-dimensional space. In specific, y is represented as a sum $y = y_{\text{fix}} + y_{\text{free}}$, where y_{fix} is determined by the constraint equation, and $y_{\text{free}} \in \ker(C)$ is free for minimization. This is accomplished by a transformation

$$Cy + c = (L_1\ 0)U \cdot U^{-1}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + c = L_1 y_1 + c = 0. \tag{3.18}$$

Here L_1 is lower triangular, y_{fix} corresponds to $y_1 = -L_1^{-1}c$, which is immediately eliminated, and y_{free} corresponds to y_2. A similar transformation of $Hy + C^T(-\lambda) + f = 0$ yields the partitioning

$$\begin{pmatrix} H_{11} & H_{21}^T \\ H_{21} & H_{22} \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + \begin{pmatrix} L_1^T \\ 0 \end{pmatrix}(-\lambda) + \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = 0. \tag{3.19}$$

Inserting y_1 gives

$$y_2 = -H_{22}^{-1}(f_2 + H_{21}y_1) \tag{3.20}$$

by means of the bottom equation in (3.19), and inserting y_1, y_2 into the top equation of (3.19) yields

$$-\lambda = -L_1^{-T}(f_1 + H_{11}y_1 + H_{21}^T y_2). \tag{3.21}$$

Finally, y_1, y_2 are re-transformed to obtain y. Note that the QP solution y is obtained directly here, without necessarily computing the multiplier λ.

Let us define projections $P_2\colon \mathbb{R}^{n_v} \to \ker(C)$, $P_1\colon \mathbb{R}^{n_v} \to \ker(C)^\perp$,

$$P_2 = U^{-1}\begin{pmatrix} 0 & 0 \\ 0 & I \end{pmatrix}, \qquad P_1 = U^{-1}\begin{pmatrix} I & 0 \\ 0 & 0 \end{pmatrix}, \tag{3.22}$$

which formally describe the geometric restriction to the feasible linear subspace, and corresponding isomorphisms $Z_2\colon \mathbb{R}^{n_v - n_c} \cong \ker(C)$, $Z_1\colon \mathbb{R}^{n_c} \cong \ker(C)^\perp$,

$$Z_2 = U^{-1}\begin{pmatrix} 0 \\ I \end{pmatrix}, \qquad Z_1 = U^{-1}\begin{pmatrix} I \\ 0 \end{pmatrix}. \tag{3.23}$$

The relevant part of the Hessian is its projection $P_2^T H P_2$, or equivalently, the reduced Hessian $Z_2^T H Z_2 = H_{22}$.

Clearly, the PH method needs less restrictive regularity assumptions than the SC method. It works whenever (3.1,3.2) has a *unique* minimum, i.e., when (a) the constraint matrix has full rank, $CC^T > 0$, and (b) the Hessian is positive definite on $\ker(C)$, $Z_2^T H Z_2 = H_{22} > 0$. This is precisely our global regularity assumption.

The PH factorization can be implemented in several slightly different ways. Our choice corresponds to a symmetric decomposition of Ω of the form

$$\Omega = \Lambda_1\Lambda_2\Lambda_3\Lambda_4\Pi\Lambda_4^T\Lambda_3^T\Lambda_2^T\Lambda_1^T, \tag{3.24}$$

where

$$\Lambda_1\Lambda_2\Lambda_3\Lambda_4 = \begin{pmatrix} I & \\ & L_1 \end{pmatrix}\begin{pmatrix} U^T & \\ & I \end{pmatrix}\begin{pmatrix} I & & \\ & I & H_{21} \\ & & I \end{pmatrix}\begin{pmatrix} I & & \\ & L_2 & \\ & & I \end{pmatrix}, \tag{3.25}$$

$L_2L_2^T = H_{22}$, and

$$\Pi = \begin{pmatrix} H_{11} & & I \\ & I & \\ I & & \end{pmatrix}. \tag{3.26}$$

The inverse factorization

$$\Omega^{-1} = \Lambda_1^{-T}\Lambda_2^{-T}\Lambda_3^{-T}\Lambda_4^{-T}\Pi^{-1}\Lambda_4^{-1}\Lambda_3^{-1}\Lambda_2^{-1}\Lambda_1^{-1} \tag{3.27}$$

may be written

$$\begin{pmatrix} H & C^T \\ C & \end{pmatrix}^{-1} = \Lambda_1^{-T}\Lambda_2^{-T}\begin{pmatrix} 0 & 0 & I \\ 0 & H_{22}^{-1} & -H_{22}^{-1}H_{21} \\ I & -H_{21}^TH_{22}^{-1} & -H_{11} + H_{21}^TH_{22}^{-1}H_{21} \end{pmatrix}\Lambda_2^{-1}\Lambda_1^{-1}. \tag{3.28}$$

When an orthogonal factorization is applied to C, $U^TU = I$, the projection is also orthogonal, and the PH method is stable. When a general factorization is applied, the projection is not orthogonal and the projected Hessian method is less stable. In this case the matrix U may be further partitioned according to

$$U = \begin{pmatrix} R & W \\ & I \end{pmatrix} = \begin{pmatrix} R & \\ & I \end{pmatrix}\begin{pmatrix} I & \bar{W} \\ & I \end{pmatrix}, \tag{3.29}$$

where R is upper triangular. We will, however, not distinguish the two alternatives in the following, and the possible substructure of U will not be considered.

In analogy to the SC method, operations of the PH method are given in table 3.2. Multiplication by Π^{-1} is essentially performed in step 3a of the transformation phase rather than in a separate phase; hence the slight asymmetry of operations. Note that no operations take place in factorization step 3, since the nontrivial block H_{21} of Λ_3 is already generated in step 2.

Again, symmetry and indefinite structure are exploited in the factorization, and the results overwrite the original data. Essentially no additional storage is needed, except for pivoting indices, or a vector of length n_c when U is orthogonal.

Table 3.2: Operations of the PH method

Step	Factorization	Transformation	Solution
(1)	$C =: (L_1\ 0)U$	$y_1 := L_1^{-1}(-c)$	$-\lambda := L_1^{-T}\bar{\lambda}$
(2)	$\begin{pmatrix} H_{11} & H_{21}^T \\ H_{21} & H_{22} \end{pmatrix} := U^{-T}HU^{-1}$	$\begin{pmatrix} f_1 \\ f_2 \end{pmatrix} := U^{-T}(-f)$	$y := U^{-1}\begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$
(3a) (3b)		$\bar{f}_1 := -H_{11}y_1 + f_1$ $\bar{f}_2 := -H_{21}y_1 + f_2$	 $\bar{\lambda} := -H_{21}^T y_2 + \bar{f}_1$
(4)	$H_{22} =: L_2 L_2^T$	$\bar{y}_2 := L_2^{-1}\bar{f}_2$	$y_2 := L_2^{-T}\bar{y}_2$

3.2 A recursive $\mathcal{O}(m)$ algorithm

We are now prepared to consider the KKT system (3.3) with its full m-stage block-sparse substructure according to (2.53–2.58). The constraint equation

$$Cy + c = \begin{pmatrix} G \\ F \end{pmatrix} y + \begin{pmatrix} h \\ e \end{pmatrix} = 0 \tag{3.30}$$

includes continuity conditions $Gy + h = 0$ and boundary conditions $Fy + e = 0$; its multiplier is

$$-\eta = \begin{pmatrix} \lambda \\ \mu \end{pmatrix}. \tag{3.31}$$

According to (2.53–2.58), partitionings of the block-diagonal Hessian, the state increment and the gradient of the cost function are given by

$$H = \begin{pmatrix} H_1 & {}^tJ_1 & & & \\ J_1 & K_1 & & & \\ & & \ddots & & \\ & & & H_m & {}^tJ_m \\ & & & J_m & K_m \end{pmatrix}, \qquad \Delta y = \begin{pmatrix} \Delta x_1 \\ \Delta u_1 \\ \vdots \\ \Delta x_m \\ \Delta u_m \end{pmatrix}, \quad f = \begin{pmatrix} f_1 \\ d_1 \\ \vdots \\ f_m \\ d_m \end{pmatrix}. \tag{3.32}$$

The equations involving H_j or K_j will be referred to as state or control minimization conditions, respectively. Analogous partitioning of the matrix, multiplier and vector of the continuity conditions yields

$$G = \begin{pmatrix} G_1 & E_1 & -I & 0 & & & \\ & & \ddots & & & & \\ & & & G_{m-1} & E_{m-1} & -I & 0 \end{pmatrix}, \qquad \lambda = \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_{m-1} \end{pmatrix}, \quad h = \begin{pmatrix} h_1 \\ \vdots \\ h_{m-1} \end{pmatrix}. \tag{3.33}$$

The matrix, multiplier and vector of the boundary conditions are partitioned according to

$$F = \begin{pmatrix} F_1^x & & & & & \\ & D_1^u & & & & \\ F_1^c & D_1^c & & & & \\ & & \ddots & & & \\ & & & F_m^x & & \\ & & & & D_m^u \\ & & & F_m^c & D_m^c \\ F_1 & D_1 & \dots & F_m & D_m \end{pmatrix}, \qquad \mu = \begin{pmatrix} \mu_1^x \\ \mu_1^u \\ \mu_1^c \\ \vdots \\ \mu_m^x \\ \mu_m^u \\ \mu_m^c \\ \mu_* \end{pmatrix}, \quad e = \begin{pmatrix} e_1^x \\ e_1^u \\ e_1^c \\ \vdots \\ e_m^x \\ e_m^u \\ e_m^c \\ e_* \end{pmatrix}. \tag{3.34}$$

As in section 2.3, we distinguish (local) state constraints, F_j^x, (local) control constraints, D_j^u, (locally) coupled constraints, (F_j^c, D_j^c), and global constraints, (F_j, D_j).

3.2.1 Basic principles

The new algorithm combines PH and SC methods in an alternating fashion to compute the solution of the m-stage KKT system (3.3). In a backward recursion, $k = m(-1)1$, a sequence of equivalent k-stage KKT systems

$$\Omega_k z_k + a_k = 0 \tag{3.35}$$

is generated. Each recursion step "cuts off" the last stage by applying local projections on boundary, interior and continuity conditions, and then computing a local Schur complement. After solving the root system $\Omega_1 z_1 + a_1 = 0$ the final solution z is generated in a forward recursion.

In matrix terms, the backward recursion generates a block-sparse decomposition

$$\Omega = \Lambda \Pi \Lambda^T, \qquad \Lambda = \Lambda_m \dots \Lambda_1, \quad \Pi = \Pi_m \dots \Pi_1 \tag{3.36}$$

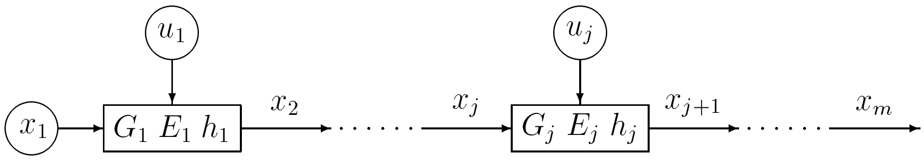
in the factorization phase, and a transformed right hand side $a_1 = \Lambda^{-1} a$ in the transformation phase. Finally, the forward recursion generates $z = \Lambda^{-T} z_1$ from $z_1 = -\Pi^{-1} a_1$ in the solution phase. Each phase requires $\mathcal{O}(m)$ operations; the local complexity in each stage is $\mathcal{O}(n_j^3)$ for the factorization and $\mathcal{O}(n_j^2)$ for transformation and solution phases, where n_j denotes the local block size.

The whole procedure is based on the optimal control nature of the discrete KKT system. Actually, its *linear-quadratic multistage* structure permits effective application of the *dynamic programming* principle. We will now develop a control-theoretical motivation of the recursive algorithm. In the remainder of this chapter we will drop all Δ's and write x_j, u_j for the increments $\Delta x_j, \Delta u_j$. Optimal values will be denoted by $\hat{x}_j, \hat{u}_j$.

Let us first consider the underlying discrete dynamic system only, i.e., the continuity conditions $x_{j+1} = G_j x_j + E_j u_j + h_j$, but no interior point or boundary conditions. Obviously all the state variables $x_2, \dots, x_m$ are uniquely determined when any initial state x_1

and arbitrary controls $u_1, \ldots, u_{m-1}$ are given. A precise description of the dependencies is expressed by the simple chain structure in figure 3.1. Each rectangular box repre-

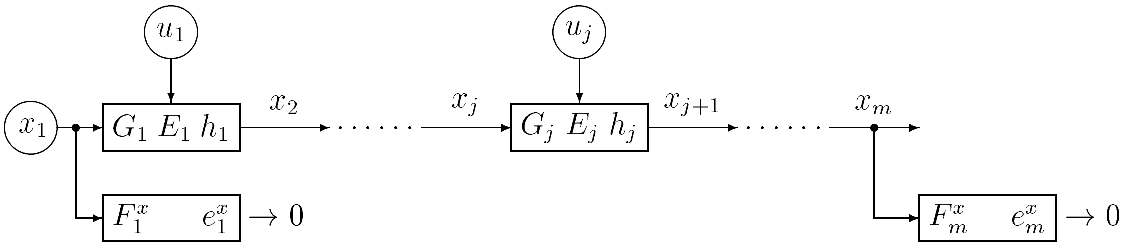
Figure 3.1: Chain structure of a discrete dynamic system



sents a stage connection which produces *dependent* variables $x_2, \ldots, x_m$ from encircled *independent* variables $x_1, u_1 \ldots, u_{m-1}$. The latter are free for minimization.

Next, consider a two-point boundary value problem with separated initial and final conditions, $F_1^x x_1 + e_1^x = 0$ and $F_m^x x_m + e_m^x = 0$, as indicated by the additional boxes in figure 3.2. Initial conditions simply fix some or all components of the independent

Figure 3.2: Chain structure of a decoupled TPBVP



variable x_1, thereby restricting x_1 to a certain subset of the state space, the initial manifold $\mathcal{S}_1^0$. Similarly, final conditions restrict x_m to a terminal manifold $\mathcal{S}_m$. However, this means fixing dependent variables which are indirectly controlled by the independent variables in *previous* stages. In order to analyze the resulting backward influence, we define the set $\mathcal{S}_{m-1}$ of preceding states from which $\mathcal{S}_m$ is attainable, and proceed recursively, defining local feasible manifolds

$$\mathcal{S}_j = \{x_j \mid \exists u_j \colon G_j x_j + E_j u_j + h_j \in \mathcal{S}_{j+1}\}. \tag{3.37}$$

(At stage one we must form the intersection with $\mathcal{S}_1^0$.) For each specific point $x_j \in \mathcal{S}_j$ the attainability condition determines a nonempty set of feasible controls,

$$\mathcal{C}_j(x_j) = \{u_j \mid G_j x_j + E_j u_j + h_j \in \mathcal{S}_{j+1}\}. \tag{3.38}$$

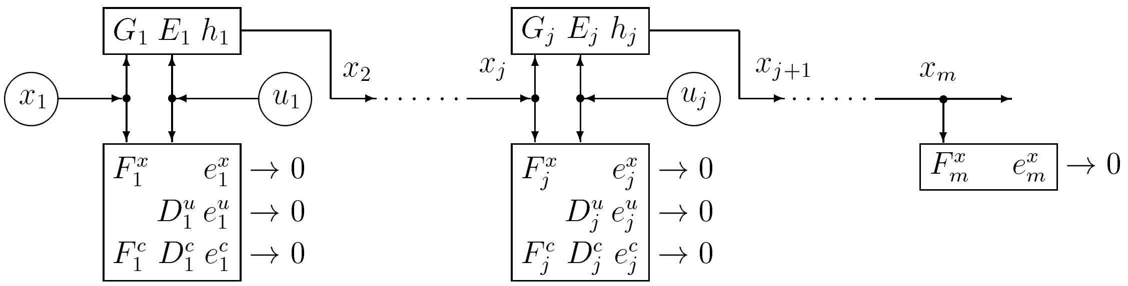
If $\mathcal{C}_j(x_j)$ contains only one element, then u_j is completely determined. Otherwise some components of u_j are free for minimization. Similarly, certain components of x_1 may remain free.

According to the dynamic programming principle the *last* optimal control value $\hat{u}_{m-1}$ is also optimal for the restricted problem of steering x from optimal state $\hat{x}_{m-1}$ to the terminal manifold $\mathcal{S}_m$. Hence, $\hat{u}_{m-1}$ is *locally* determined by minimization over $\mathcal{C}_{m-1}(\hat{x}_{m-1})$: it depends only on $\hat{x}_{m-1}$. Since the constraints are linear and the cost function is quadratic, we can explicitly express $\mathcal{S}_{m-1}$ and $\mathcal{C}_{m-1}(x_{m-1})$ by affine mappings. This corresponds to the projection part of the algorithm. Similarly, we can explicitly represent the minimizing $\hat{u}_{m-1}$ by an affine mapping, which is done in the Schur complement step. This reduces the m-stage problem to a similar $(m-1)$-stage problem, and the same principle applies recursively. Minimization over $\mathcal{S}_1$ finally yields the initial state x_1.

Higher order state constraints may produce a backward effect on several preceding stages. Consider, for instance, the discretized point-to-point robot trajectory planning problem in chapter 1. Final conditions are imposed on the position (order two) and velocity variables (order one). Control u_{m-1} may be used to satisfy the velocity constraint, i.e., for breaking, and control u_{m-2} may be used to satisfy the implied position constraint at stage $m-1$, i.e., to reach the position from where u_{m-1} steers the robot to the desired final position.

We will now study the effect of arbitrary local constraints as appearing in a general *decoupled* multipoint boundary value problem, see figure 3.3. Local control constraints,

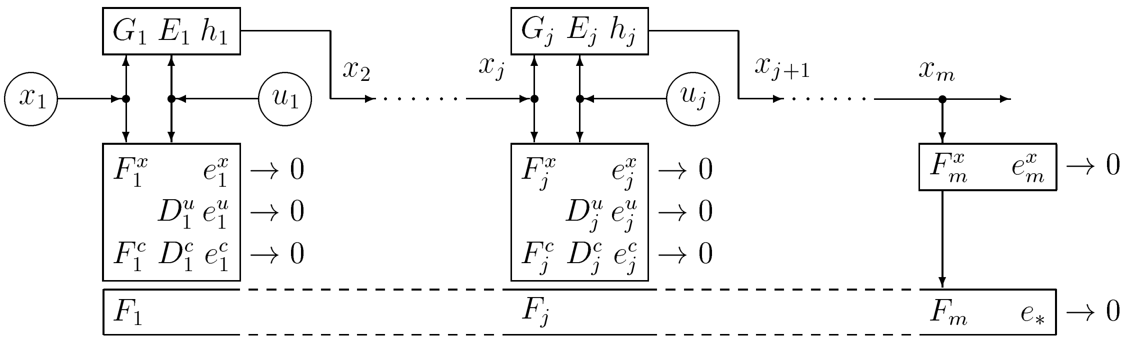
Figure 3.3: Chain structure of a decoupled MPBVP



$D_j^u u_j + e_j^u = 0$, behave exactly like initial conditions. They fix independent variables locally, thereby reducing the number of degrees of freedom without any backward effect. Interior state constraints, $F_j^x x_j + e_j^x = 0$, also act as initial conditions when seen from the tail of the chain. Seen from the head, however, they behave exactly like final conditions, determining independent variables in one or more previous stages. Locally coupled constraints on states and controls, which arise from state restrictions in the following stage, combine local elimination and backward effect in a similar way. (Details will be given below.) Basically the recursion is similar to the previously described TPBVP case, only the definitions of sets $\mathcal{S}_j$ and $\mathcal{C}_j$ involve additional local restrictions.

The general case of a globally coupled multipoint boundary value problem is shown in figure 3.4. Including global constraints alters the control-theoretical interpretation of the recursion (the dynamic programming principle does not apply any more), and, in connection with this, our global regularity assumption must be replaced by slightly enforced requirements that will be discussed below. However, the solution algorithm remains essentially unchanged, while the new global constraints are recursively transformed into initial conditions and then taken into account in the root system.

Figure 3.4: Chain structure of a globally coupled MPBVP



Each stage of the recursion involves six basic steps. Each step reduces the local system size by direct or formal elimination of some variables. Here we give a brief overview of these steps before describing them in detail.
Step 1 (PH). Elimination of local control constraints.
Step 2 (PH). Elimination of local state constraints.
Step 3 (PH). Elimination of control part of locally coupled constraints.
Step 4 (PH). Elimination of state part of locally coupled constraints.
Step 5 (PH). Elimination of continuity conditions.
Step 6 (SC). Elimination of control minimization conditions.

Steps 1 and 2 treat decoupled local constraints, which may arise from boundary conditions or (within an active-set strategy) from path constraints. Steps 3 and 4 treat locally coupled constraints, which result as attainability conditions from the elimination of state constraints in the following stage. Step 5 cuts off the following stage by eliminating the continuity conditions. Altogether, these steps comprise a hierarchical projection onto the local feasible submanifold $\mathcal{S}_j$. Finally, the Schur complement step 6 eliminates the local control minimization conditions.

3.2.2 The recursion—step by step

In stage j of the recursion, a j-stage KKT system $\Omega_j z_j + a_j = 0$ will be obtained from a $(j+1)$-stage system. Before the recursion starts, an artificial stage $m+1$ is formally appended to the original KKT system (3.3). Some other definitions are also introduced, most of them for the handling of globally coupled constraints.

Let us first define $\lambda_0 := 0$, with the same dimension as x_1, and formally introduce degenerate matrices $(G_m\ E_m) := 0$ and vectors $\lambda_m := 0$, $h_m := 0$, and $x_{m+1} := 0$, with dimension zero. This is needed to make the equations for stages 1 and m in the recursive formulation look exactly the same as in all other stages.

Next, we introduce stage-wise local representations of the global constraints by defining vectors $e_0, \dots, e_m$ which satisfy

$$\sum_{j=1}^{k} \begin{pmatrix} F_j & D_j \end{pmatrix} \begin{pmatrix} x_j \\ u_j \end{pmatrix} + e_k = 0. \tag{3.39}$$

This yields $e_0 = 0$ and the recursive relation

$$\begin{pmatrix} F_j & D_j \end{pmatrix} \begin{pmatrix} x_j \\ u_j \end{pmatrix} + e_j - e_{j-1} = 0, \qquad j = 1(1)m. \tag{3.40}$$

Of course, $e_m = e_*$ is given initially, and (3.39) is the original global constraint in the case $k = m$.

When recursion step $j+1$ is finished, certain matrices $S_{j+1}, R_{j+1}, Q_{j+1}$ and vectors $\alpha_{j+1}, \beta_{j+1}$ have been generated which replace the tail stages $j+1, \dots, m$. These data form an indefinite augmentation

$$\begin{bmatrix} S_{j+1} & {}^tR_{j+1} \\ R_{j+1} & -Q_{j+1} \end{bmatrix} \begin{bmatrix} x^3_{j+1} \\ \mu_* \end{bmatrix} + \begin{bmatrix} \alpha_{j+1} - \lambda^3_j \\ \beta_{j+1} - e_j \end{bmatrix} = 0 \tag{3.41}$$

to the as yet unchanged head stages $1, \dots, j$ of the KKT system. Here x^3_{j+1}, λ^3_j denote certain components of x_{j+1}, λ_{j+1}, respectively (see step 6 below), and S_{j+1}, Q_{j+1} are positive semi-definite matrices. The initial augmentation is defined by $S_{m+1} := 0$, $R_{m+1} := 0$, $Q_{m+1} := 0$, $\alpha_{m+1} := 0$, and $\beta_{m+1} := e_*$, so that (3.41) is always satisfied for $j = m$. Again degenerate quantities appear: S_{m+1} has zero rows and columns.

Let us now consider stage j of the backward recursion, $j = m(-1)1$. The $(m+1)$-stage system $\Omega_{m+1} z_{m+1} + a_{m+1} = 0$ has already been reduced to a $(j+1)$-stage system $\Omega_{j+1} z_{j+1} + a_{j+1} = 0$. Using the above definitions and including stage-coupling terms in the right hand side, stage j together with the augmentation can be written

$$\left[\begin{array}{cc|ccccc} H_j & {}^tJ_j & {}^tG_j & {}^tF^x_j & & {}^tF^c_j & {}^tF_j \\ J_j & K_j & {}^tE_j & & {}^tD^u_j & {}^tD^c_j & {}^tD_j \\ \hline G_j & E_j & & & & & \\ F^x_j & & & & & & \\ & D^u_j & & & & & \\ F^c_j & D^c_j & & & & & \\ F_j & D_j & & & & & \end{array}\right] \left[\begin{array}{c} x_j \\ u_j \\ \hline \lambda_j \\ \mu^x_j \\ \mu^u_j \\ \mu^c_j \\ \mu_* \end{array}\right] + \left[\begin{array}{c} f_j - \lambda_{j-1} \\ d_j \\ \hline h_j - x_{j+1} \\ e^x_j \\ e^u_j \\ e^c_j \\ e_j - e_{j-1} \end{array}\right] = 0, \tag{3.42}$$

$$\begin{bmatrix} S_{j+1} & {}^tR_{j+1} \\ R_{j+1} & -Q_{j+1} \end{bmatrix} \begin{bmatrix} x^3_{j+1} \\ \mu_* \end{bmatrix} + \begin{bmatrix} \alpha_{j+1} - \lambda^3_j \\ \beta_{j+1} - e_j \end{bmatrix} = 0. \tag{3.43}$$

All previous stages $i < j$ have exactly the same form as (3.42). e_j plays the role of a coupling term between (3.42,3.43), and actual computations will be performed on β_{j+1} which carries the relevant information on the right hand side of globally coupled constraints.

Recursion step 1 (PH). *Elimination of the local control constraints by a projection onto* $\ker(D_j^u)$.
The constraint matrix D_j^u is factorized and the KKT system is transformed accordingly. This yields a partitioning $u_j \to (u_j^1, u_j^*)$, where u_j^1 is determined immediately from the constraint equation. Its multiplier μ_j^u is formally eliminated using the corresponding minimization condition, and is actually computed in the solution phase. Finally, u_j is obtained from components u_j^1, u_j^*.

Table 3.3: Step 1 of recursive KKT solver

Factorization	Transformation	Solution
$D_j^u =: (L_j^{u1}\ \ 0)U_j^1$	$u_j^1 := -(L_j^{u1})^{-1}e_j^u$	$\mu_j^u := -({}^tL_j^{u1})^{-1}[$
$\begin{pmatrix} J_j^1 \\ J_j^* \end{pmatrix} := ({}^tU_j^1)^{-1}J_j$	$f_j^* := {}^tJ_j^1 u_j^1 + f_j$	$J_j^1 x_j +$
$\begin{pmatrix} K_j^{11} & {}^tK_j^{*1} \\ K_j^{*1} & K_j^{**} \end{pmatrix} := ({}^tU_j^1)^{-1}K_j(U_j^1)^{-1}$	$\begin{pmatrix} \hat{d}_j^1 \\ \hat{d}_j^* \end{pmatrix} := ({}^tU_j^1)^{-1}d_j$	$\boxed{u_j := (U_j^1)^{-1}}\begin{pmatrix} u_j^1 \\ u_j^* \end{pmatrix}$
	$d_j^1 := K_j^{11}u_j^1 + \hat{d}_j^1$	
	$d_j^* := K_j^{*1}u_j^1 + \hat{d}_j^*$	${}^tK_j^{*1}u_j^* +$
$(E_j^1\ \ E_j^*) := E_j(U_j^1)^{-1}$	$h_j^* := E_j^1 u_j^1 + h_j$	${}^tE_j^1\lambda_j +$
$(D_j^{c1}\ \ D_j^{c*}) := D_j^c(U_j^1)^{-1}$	$e_j^{c*} := D_j^{c1}u_j^1 + e_j^c$	${}^tD_j^{c1}\mu_j^c +$
$(D_j^1\ \ D_j^*) := D_j(U_j^1)^{-1}$	$e_j^* := D_j^1 u_j^1 + e_j$	${}^tD_j^1\mu_* + d_j^1]$

Table 3.3 describes each phase of step 1 in full detail. As in sections 3.1.2 and 3.1.3, corresponding operations are lined up horizontally, blocks in the solution phase are transposed as compared to factorization and transformation phases, and the operations of the solution phase proceed in reverse order. Vectors entering the solution phase are intermediate results that have been computed in recursion step 2 or earlier steps/stages of the algorithm. The right-open box around u_j indicates that this operation is not part of the summation involved in computing μ_j^u. Vectors β_{j+1} and e_j undergo identical transformations in recursion steps 1–4. Hence, we define $\beta_{j+1}^* := D_j^1 u_j^1 + \beta_{j+1}$ in analogy to $e_j^* = D_j^1 u_j^1 + e_j$.

The first part of step 1 is equivalent to steps (1–3b) of the PH algorithm, where the Hessian H consists of blocks $H_j, J_j, {}^tJ_j, K_j$, and the constraint matrix is $C = (0\ D_j^u)$. The last part of step 1 is a projection of additional local constraint matrix blocks E_j, D_j^c, D_j. But the complete KKT system also contains all the blocks in previous stages $i < j$; these are not at all affected by the local transformation. PH step 4 is therefore replaced by the remaining part of the recursion. The augmentation remains also unchanged until step 5; only the difference $\beta_{j+1} - e_j$ is formally replaced with equivalent expressions. Recall from section 3.1.3 that either an orthogonal or a general factorization may be applied to D_j^u, leading to an orthogonal or a general projection step.

After step 1 the local KKT system is reduced to

$$\left[\begin{array}{cc|cccc} H_j & {}^tJ_j^* & {}^tG_j & {}^tF_j^x & {}^tF_j^c & {}^tF_j \\ J_j^* & K_j^{**} & {}^tE_j^* & & {}^tD_j^{c*} & {}^tD_j^* \\ \hline G_j & E_j^* & & & & \\ F_j^x & & & & & \\ F_j^c & D_j^{c*} & & & & \\ F_j & D_j^* & & & & \end{array}\right] \left[\begin{array}{c} x_j \\ u_j^* \\ \hline \lambda_j \\ \mu_j^x \\ \mu_j^c \\ \mu_* \end{array}\right] + \left[\begin{array}{c} f_j^* - \lambda_{j-1} \\ d_j^* \\ \hline h_j^* - x_{j+1} \\ e_j^x \\ e_j^{c*} \\ e_j^* - e_{j-1} \end{array}\right] = 0, \tag{3.44}$$

$$\begin{bmatrix} S_{j+1} & {}^tR_{j+1} \\ R_{j+1} & -Q_{j+1} \end{bmatrix} \begin{bmatrix} x_{j+1}^3 \\ \mu_* \end{bmatrix} + \begin{bmatrix} \alpha_{j+1} - \lambda_j^3 \\ \beta_{j+1}^* - e_j^* \end{bmatrix} = 0. \tag{3.45}$$

Recursion step 2 (PH). *Elimination of the local state constraints by a projection onto* $\ker(F_j^x)$.
In complete analogy to step 1, the constraint matrix F_j^x is factorized and the system is transformed accordingly. This yields a partitioning $x_j \to (x_j^1, x_j^*)$, where x_j^1 is determined immediately from the constraint equation. Its multiplier μ_j^x is formally eliminated using the corresponding minimization condition, and is actually computed in the solution phase. Finally, x_j is obtained from components x_j^1, x_j^*. These operations are given in table 3.4. In addition we define $\beta_{j+1}^{**} := F_j^1 x_j^1 + \beta_{j+1}^*$ in analogy to $e_j^{**} = F_j^1 x_j^1 + e_j^*$. Again either an orthogonal or a general projection may be used.

Table 3.4: Step 2a of recursive KKT solver

Factorization	Transformation	Solution
$F_j^x =: (L_j^{x1}\ 0)V_j^1$	$x_j^1 := -(L_j^{x1})^{-1}e_j^x$	$\mu_j^x := -({}^tL_j^{x1})^{-1}[$
$\begin{pmatrix} H_j^{11} & {}^tH_j^{*1} \\ H_j^{*1} & H_j^{**} \end{pmatrix} := ({}^tV_j^1)^{-1}H_j(V_j^1)^{-1}$	$\begin{pmatrix} \hat{f}_j^{*1} \\ \hat{f}_j^{**} \end{pmatrix} := ({}^tV_j^1)^{-1}f_j^*$	$\boxed{x_j := (V_j^1)^{-1}\begin{pmatrix} x_j^1 \\ x_j^* \end{pmatrix}}$
	$f_j^{*1} := H_j^{11}x_j^1 + \hat{f}_j^{*1}$	
	$f_j^{**} := H_j^{*1}x_j^1 + \hat{f}_j^{**}$	${}^tH_j^{*1}x_j^* +$
$(J_j^{*1}\ J_j^{**}) := J_j^*(V_j^1)^{-1}$	$d_j^{**} := J_j^{*1}x_j^1 + d_j^*$	${}^tJ_j^{*1}u_j^* +$
$(G_j^1\ G_j^*) := G_j(V_j^1)^{-1}$	$\hat{h}_j^* := G_j^1 x_j^1 + h_j^*$	${}^tG_j^1\lambda_j +$
$(F_j^{c1}\ F_j^{c*}) := F_j^c(V_j^1)^{-1}$	$e_j^{c**} := F_j^{c1}x_j^1 + e_j^{c*}$	${}^tF_j^{c1}\mu_j^c +$
$(F_j^1\ F_j^*) := F_j(V_j^1)^{-1}$	$e_j^{**} := F_j^1 x_j^1 + e_j^*$	${}^tF_j^1\mu_* + \tilde{f}_j^{*1}]$

Unlike step 1, this projection has a side effect on the previous stage as explained in section 3.2.1. After elimination of x_{j+1}^1 and x_j^1 the continuity conditions at stage j read

$$G_j^* x_j^* + E_j^* u_j^* + \hat{h}_j^* = x_{j+1} = (V_{j+1}^1)^{-1}\begin{pmatrix} x_{j+1}^1 \\ x_{j+1}^* \end{pmatrix}. \tag{3.46}$$

Multiplication by V^1_{j+1} and inserting x^1_{j+1} (which is already known) yields partitionings of the multiplier $\lambda_j \to (\lambda^1_j, \lambda^*_j)$ and transformed continuity conditions according to table 3.5,

$$\begin{pmatrix} G^{*1}_j & E^{*1}_j \\ G^{**}_j & E^{**}_j \end{pmatrix} \begin{pmatrix} x^*_j \\ u^*_j \end{pmatrix} + \begin{pmatrix} h^{*1}_j \\ h^{**}_j \end{pmatrix} = \begin{pmatrix} 0 \\ x^*_{j+1} \end{pmatrix}. \tag{3.47}$$

The rows corresponding to fixed components x^1_{j+1} are now decoupled from stage $j+1$. They are converted to locally coupled constraints representing conditions for *attainability* of $\mathcal{S}_{j+1}$. Rows associated with x^*_{j+1} are the remaining continuity conditions.

Table 3.5: Step 2b of recursive KKT solver

Factorization	Transformation	Solution
$\begin{pmatrix} G^{*1}_j & E^{*1}_j \\ G^{**}_j & E^{**}_j \end{pmatrix} := V^1_{j+1}(G^*_j \;\; E^*_j)$	$\begin{pmatrix} \hat{h}^{*1}_j \\ h^{**}_j \end{pmatrix} := V^1_{j+1}\hat{h}^*_j$	$\lambda_j := {}^tV^1_{j+1}\begin{pmatrix} \lambda^1_j \\ \lambda^*_j \end{pmatrix}$
	$h^{*1}_j := -x^1_{j+1} + \hat{h}^{*1}_j$	$\tilde{f}^{*1}_{j+1} := -\lambda^1_j + f^{*1}_{j+1}$

It may be noted that all the operations of steps 1 and 2 are independent with respect to the m stages. Therefore, these steps may be performed as m-parallel preprocessing (factorization and transformation phases) or postprocessing (solution phase) to the "real" recursion (steps 3–6). Step 1 does not require any information exchange, whereas processors $j+1$ and j must transmit data between steps 2a and 2b. The solution phase needs μ_* on each processor.

Recursion step 3 (PH). *Elimination of the control part of locally coupled constraints by a projection onto* $\ker(A_j)$.
Further components of x_j and u_j will be eliminated in steps 3 and 4 by means of locally coupled constraints. Before this can be done, some preparations must take place.

Table 3.6: Step 3a of recursive KKT solver

Factorization	Transformation	Solution
$\begin{pmatrix} G^2_j & E^2_j \\ G^3_j & E^3_j \end{pmatrix} := V^2_{j+1}(G^{**}_j \;\; E^{**}_j)$	$\begin{pmatrix} \hat{h}^2_j \\ h^3_j \end{pmatrix} := V^2_{j+1}\hat{h}^{**}_j$	$\lambda^*_j := {}^tV^2_{j+1}\begin{pmatrix} \lambda^2_j \\ \lambda^3_j \end{pmatrix}$
	$h^2_j := -x^2_{j+1} + \hat{h}^2_j$	$\hat{f}^2_{j+1} := -\lambda^2_j + \tilde{f}^2_{j+1}$
$(C_j \;\; A_j) := \begin{pmatrix} G^{*1}_j & E^{*1}_j \\ G^2_j & E^2_j \\ F^{c*}_j & D^{c*}_j \end{pmatrix}$	$c_j := \begin{pmatrix} h^{*1}_j \\ h^2_j \\ e^{c**}_j \end{pmatrix}$	$\begin{pmatrix} \lambda^1_j \\ \lambda^2_j \\ \mu^c_j \end{pmatrix} := \nu_j$

The splitting effect of step 2 also occurs in step 4, when eliminating additional components x^2_{j+1} converts corresponding rows of the continuity conditions (see below). Table 3.6

gives the operations associated with this splitting. Additionally, matrices A_j, C_j and vectors c_j, ν_j are defined to combine original locally coupled constraints (if present) with converted continuity conditions. This produces the local system

$$\left[\begin{array}{cc|ccc} H_j^{**} & {}^tJ_j^{**} & {}^tG_j^3 & {}^tC_j & {}^tF_j^* \\ J_j^{**} & K_j^{**} & {}^tE_j^3 & {}^tA_j & {}^tD_j^* \\ \hline G_j^3 & E_j^3 & & & \\ C_j & A_j & & & \\ F_j^* & D_j^* & & & \end{array}\right] \left[\begin{array}{c} x_j^* \\ u_j^* \\ \hline \lambda_j^3 \\ \nu_j \\ \mu_* \end{array}\right] + \left[\begin{array}{c} f_j^{**} - \lambda_{j-1}^* \\ d_j^{**} \\ \hline h_j^3 - x_{j+1}^3 \\ c_j \\ e_j^{**} - e_{j-1} \end{array}\right] = 0, \tag{3.48}$$

$$\begin{bmatrix} S_{j+1} & {}^tR_{j+1} \\ R_{j+1} & -Q_{j+1} \end{bmatrix} \begin{bmatrix} x_{j+1}^3 \\ \mu_* \end{bmatrix} + \begin{bmatrix} \alpha_{j+1} - \lambda_j^3 \\ \beta_{j+1}^{**} - e_j^{**} \end{bmatrix} = 0. \tag{3.49}$$

Some care must be taken in exploiting the constraint equation

$$(C_j \; A_j) \begin{pmatrix} x_j^* \\ u_j^* \end{pmatrix} + c_j = 0. \tag{3.50}$$

We have to treat x_j^* and u_j^* separately in order to maintain the block-sparse structure. However, x_j^*, u_j^* are coupled by (3.50), and an orthogonal projection onto $\ker(C_j \; A_j)$ would mix components of these vectors. As a consequence, components of both x_j^*, u_j^* would appear in the preceding continuity conditions (instead of x_j^3 only), and components of λ_{j-1}^* would appear both in the state and control minimization conditions.

Table 3.7: Step 3b of recursive KKT solver

Factorization	Transformation	Solution
$A_j =: {}^tP_j \begin{pmatrix} L_j^{u2} & 0 \\ A_j^x & \varepsilon \end{pmatrix} U_j^2$		
$\begin{pmatrix} C_j^u \\ C_j^x \end{pmatrix} := P_j C_j$	$\begin{pmatrix} c_j^u \\ c_j^x \end{pmatrix} := P_j c_j$	$\nu_j := {}^tP_j \begin{pmatrix} \nu_j^u \\ \nu_j^x \end{pmatrix}$
$\begin{pmatrix} J_j^2 \\ J_j^3 \end{pmatrix} := ({}^tU_j^2)^{-1} J_j^{**}$	$\begin{pmatrix} d_j^2 \\ d_j^3 \end{pmatrix} := ({}^tU_j^2)^{-1} d_j^{**}$	$u_j^* := (U_j^2)^{-1} \begin{pmatrix} u_j^2 \\ u_j^3 \end{pmatrix}$
$\begin{pmatrix} K_j^{22} & {}^tK_j^{32} \\ K_j^{32} & K_j^{33} \end{pmatrix} := ({}^tU_j^2)^{-1} K_j^{**} (U_j^2)^{-1}$		
$(E_j^{32} \; E_j^{33}) := E_j^3 (U_j^2)^{-1}$		
$(D_j^{*2} \; D_j^{*3}) := D_j^* (U_j^2)^{-1}$		

Since we wish to eliminate as many control variables as possible, we begin by factorizing A_j. Assume first that A_j has maximal rank. Three cases are possible.

(1) A_j has less rows than columns. Hence, *all* the constraints are needed to eliminate *some* of the control variables u_j^2 after factorizing $A_j = {}^tP_j(L_j^{u2}\ 0)U_j^2$, and some other components u_j^3 are left undetermined. (Here P_j is a permutation matrix used for pivoting.)

(2) A_j has as many rows as columns. Hence, *all* the constraints determine u_j^* *completely* after factorizing $A_j = {}^tP_jL_j^{u2}U_j^2$.

(3) A_j has more rows than columns. Hence, *some* of the constraints determine u_j^* *completely* after factorizing

$$A_j = {}^tP_j \begin{pmatrix} L_j^{u2} \\ A_j^x \end{pmatrix} U_j^2. \tag{3.51}$$

Certain rows of A_j constituting A_j^x are left over to fix state components x_j^2 in step 4.

In any of these cases A_j might be rank-deficient or ill-conditioned. This is likely to happen, e.g., when higher order state constraints are present. Consider again the PTP robot trajectory optimization problem from chapter 1. All the final positions and velocities are fixed (by conditions of orders 2 and 1, respectively), so that all the continuity conditions at stage $m-1$ are converted to attainability conditions. Hence, $A_{m-1} = E_{m-1}$, and entries in A_{m-1} are of magnitude $\mathcal{O}(\Delta\tau_{m-1}^2)$ or $\mathcal{O}(\Delta\tau_{m-1})$, depending on whether they belong to position or velocity differential equations.

For treating the rank-deficient or ill-conditioned case, a general factorization with pivoting and rank decision is computed, see table 3.7. Here the dimensions are chosen so that L_j^{u2} is well-conditioned and ε can be neglected. Rows and columns of the local KKT system are partitioned according to the choice of P_j and U_j^2, yielding partitionings $u_j^* \to (u_j^2, u_j^3)$ and $\nu_j \to (\nu_j^u, \nu_j^x)$.

Table 3.8: Step 3c of recursive KKT solver

Factorization	Transformation	Solution
$\bar{C}_j^u := -(L_j^{u2})^{-1}C_j^u$	$\bar{c}_j^u := -(L_j^{u2})^{-1}c_j^u$	$\nu_j^u := -({}^tL_j^{u2})^{-1}[$
$W := K_j^{22}\bar{C}_j^u$	$w := K_j^{22}\bar{c}_j^u + d_j^2$	$K_j^{22}u_j^2 +$
$\bar{H}_j := {}^t\bar{C}_j^u W +$	$\bar{f}_j := {}^t\bar{C}_j^u w +$	$\boxed{u_j^2 := \bar{C}_j^u x_j^* + \bar{c}_j^u}$
$({}^tJ_j^2\bar{C}_j^u + {}^t\bar{C}_j^u J_j^2) + H_j^{**}$	${}^tJ_j^2\bar{c}_j^u + f_j^{**}$	$J_j^2 x_j^* +$
$\bar{J}_j^3 := K_j^{32}\bar{C}_j^u + J_j^3$	$\bar{d}_j^3 := K_j^{32}\bar{c}_j^u + d_j^3$	${}^tK_j^{32}u_j^3 +$
$\bar{G}_j^3 := E_j^{32}\bar{C}_j^u + G_j^3$	$\bar{h}_j^3 := E_j^{32}\bar{c}_j^u + h_j^3$	${}^tE_j^{32}\lambda_j^3 +$
$\bar{C}_j^x := A_j^x\bar{C}_j^u + C_j^x$	$\bar{c}_j^x := A_j^x\bar{c}_j^u + c_j^x$	${}^tA_j^x\nu_j^x +$
$\bar{F}_j^* := D_j^{*2}\bar{C}_j^u + F_j^*$	$\bar{e}_j := D_j^{*2}\bar{c}_j^u + e_j^{**}$	${}^tD_j^{*2}\mu_* + d_j^2]$

Step 3 is *not* similar to step 1, since unknown state components x_j^* are present in the constraint equation (3.50). Hence, u_j^* is only formally eliminated which makes the projection considerably more complicated, see table 3.8. (Here W and w denote temporary "workspace" variables.)

After step 3 the reduced KKT system has exactly the same form as after step 1, where local state constraints (if present) remain from partially eliminated coupled constraints:

$$
\left[\begin{array}{cc|ccc}
\bar{H}_j & {}^t\bar{J}^3_j & {}^t\bar{G}^3_j & {}^t\bar{C}^x_j & {}^t\bar{F}^*_j \\
\bar{J}^3_j & \bar{K}^{33}_j & {}^tE^{33}_j & & {}^tD^{*3}_j \\
\hline
\bar{G}^3_j & E^{33}_j & & & \\
\bar{C}^x_j & & & & \\
\bar{F}^*_j & D^{*3}_j & & &
\end{array}\right]
\left[\begin{array}{c}
x^*_j \\ u^3_j \\ \hline \lambda^3_j \\ \nu^x_j \\ \mu_*
\end{array}\right]
+
\left[\begin{array}{c}
\bar{f}_j - \lambda^*_{j-1} \\ \bar{d}^3_j \\ \hline \bar{h}^3_j - x^3_{j+1} \\ \bar{c}^x_j \\ \bar{e}_j - e_{j-1}
\end{array}\right]
= 0, \tag{3.52}
$$

$$
\begin{bmatrix} S_{j+1} & {}^tR_{j+1} \\ R_{j+1} & -Q_{j+1} \end{bmatrix}
\begin{bmatrix} x^3_{j+1} \\ \mu_* \end{bmatrix}
+
\begin{bmatrix} \alpha_{j+1} - \lambda^3_j \\ \bar{\beta}_{j+1} - \bar{e}_j \end{bmatrix}
= 0. \tag{3.53}
$$

Recursion step 4 (PH). *Elimination of the state part of locally coupled constraints by a projection onto* $\ker(\bar{C}^x_j)$.
Step 4 is completely similar to step 2. The constraint matrix $\bar{C}^x_j$ is factorized and the system is transformed accordingly. This yields a partitioning $x^*_j \to (x^2_j, x^3_j)$, where x^2_j is determined immediately from the constraint equation. Its multiplier ν^x_j is formally eliminated using the corresponding minimization condition, and is actually computed in the solution phase. These operations are given in table 3.9. The splitting of continuity conditions due to step 4 is performed in step 3a, see table 3.6.

Table 3.9: Step 4 of recursive KKT solver

Factorization	Transformation	Solution
$\bar{C}^x_j =: (L^{x2}_j\ 0)V^2_j$	$x^2_j := -(L^{x2}_j)^{-1}\bar{c}^x_j$	$\nu^x_j := -({}^tL^{x2}_j)^{-1}[$
$\begin{pmatrix} H^{22}_j & {}^tH^{32}_j \\ H^{32}_j & H^{33}_j \end{pmatrix} := ({}^tV^2_j)^{-1}\bar{H}_j(V^2_j)^{-1}$	$\begin{pmatrix} f^2_j \\ f^3_j \end{pmatrix} := ({}^tV^2_j)^{-1}\bar{f}_j$	$\boxed{x^*_j := (V^2_j)^{-1}\begin{pmatrix} x^2_j \\ x^3_j \end{pmatrix}}$
	$\tilde{f}^2_j := H^{22}_j x^2_j + f^2_j$	
	$\tilde{f}^3_j := H^{32}_j x^2_j + f^3_j$	${}^tH^{32}_j x^3_j +$
$(J^{32}_j\ J^{33}_j) := \bar{J}^3_j(V^2_j)^{-1}$	$\tilde{d}^3_j := J^{32}_j x^2_j + \bar{d}^3_j$	${}^tJ^{32}_j u^3_j +$
$(G^{32}_j\ G^{33}_j) := \bar{G}^3_j(V^2_j)^{-1}$	$\tilde{h}^3_j := G^{32}_j x^2_j + \bar{h}^3_j$	${}^tG^{32}_j \lambda^3_j +$
$(F^{*2}_j\ F^{*3}_j) := \bar{F}^*_j(V^2_j)^{-1}$	$\tilde{e}_j := F^{*2}_j x^2_j + \bar{e}_j$	${}^tF^{*2}_j \mu_* + \hat{f}^2_j]$

At this point all the local constraints have been eliminated along with possible backward effects from higher stages. The local KKT system consists of minimization conditions, continuity conditions, global constraints and augmentation only:

$$\left[\begin{array}{cc|cc} H_j^{33} & {}^tJ_j^{33} & {}^tG_j^{33} & {}^tF_j^{*3} \\ J_j^{33} & K_j^{33} & {}^tE_j^{33} & {}^tD_j^{*3} \\ \hline G_j^{33} & E_j^{33} & & \\ F_j^{*3} & D_j^{*3} & & \end{array}\right] \left[\begin{array}{c} x_j^3 \\ u_j^3 \\ \hline \lambda_j^3 \\ \mu_* \end{array}\right] + \left[\begin{array}{c} \tilde{f}_j^3 - \lambda_{j-1}^3 \\ \tilde{d}_j^3 \\ \hline \tilde{h}_j^3 - x_{j+1}^3 \\ \tilde{e}_j - e_{j-1} \end{array}\right] = 0, \tag{3.54}$$

$$\left[\begin{array}{c|c} S_{j+1} & {}^tR_{j+1} \\ \hline R_{j+1} & -Q_{j+1} \end{array}\right] \left[\begin{array}{c} x_{j+1}^3 \\ \hline \mu_* \end{array}\right] + \left[\begin{array}{c} \alpha_{j+1} - \lambda_j^3 \\ \hline \tilde{\beta}_{j+1} - \tilde{e}_j \end{array}\right] = 0. \tag{3.55}$$

These two equation systems can be combined by inserting $\tilde{e}_j$ from (3.55) into (3.54) and rearranging the coupling terms, which yields

$$\left[\begin{array}{ccc|cc} H_j^{33} & {}^tJ_j^{33} & 0 & {}^tG_j^{33} & {}^tF_j^{*3} \\ J_j^{33} & K_j^{33} & 0 & {}^tE_j^{33} & {}^tD_j^{*3} \\ 0 & 0 & S_{j+1} & -I & {}^tR_{j+1} \\ \hline G_j^{33} & E_j^{33} & -I & 0 & 0 \\ F_j^{*3} & D_j^{*3} & R_{j+1} & 0 & -Q_{j+1} \end{array}\right] \left[\begin{array}{c} x_j^3 \\ u_j^3 \\ x_{j+1}^3 \\ \hline \lambda_j^3 \\ \mu_* \end{array}\right] + \left[\begin{array}{c} \tilde{f}_j^3 - \lambda_{j-1}^3 \\ \tilde{d}_j^3 \\ \alpha_{j+1} \\ \hline \tilde{h}_j^3 \\ \tilde{\beta}_{j+1} - e_{j-1} \end{array}\right] = 0. \tag{3.56}$$

Recursion step 5 (PH). *Elimination of the continuity conditions by a projection onto* $\ker(G_j^{33}\ E_j^{33}\ -I)$.
Here the augmentation remaining from stage $j+1$ is cut off. This is achieved by solving for x_{j+1}^3, λ_j^3 and resubstituting the resulting expressions into (3.56). Thus, x_{j+1}^3, λ_j^3 are formally eliminated; their actual values are obtained in the solution phase, see table 3.10.

Table 3.10: Step 5 of recursive KKT solver

Factorization	Transformation	Solution
$(Z_j^{\mu x}\ Z_j^{\mu u}) :=$ $(F_j^{*3}\ D_j^{*3}) + R_{j+1}(G_j^{33}\ E_j^{33})$	$z_j^\mu := R_{j+1}\tilde{h}_j^3 + \tilde{\beta}_{j+1}$	$\lambda_j^3 := {}^tR_{j+1}\mu_* + S_{j+1}x_{j+1}^3 + \alpha_{j+1}$
$W := S_{j+1}(G_j^{33}\ E_j^{33})$	$w := S_{j+1}\tilde{h}_j^3 + \alpha_{j+1}$	
$\begin{pmatrix} Z_j^{xx} & {}^tZ_j^{ux} \\ Z_j^{ux} & Z_j^{uu} \end{pmatrix} :=$ $\begin{pmatrix} H_j^{33} & {}^tJ_j^{33} \\ J_j^{33} & K_j^{33} \end{pmatrix} + \begin{pmatrix} {}^tG_j^{33} \\ {}^tE_j^{33} \end{pmatrix} W$	$\begin{pmatrix} z_j^x \\ z_j^u \end{pmatrix} := \begin{pmatrix} {}^tG_j^{33} \\ {}^tE_j^{33} \end{pmatrix} w + \begin{pmatrix} \tilde{f}_j^3 \\ \tilde{d}_j^3 \end{pmatrix}$	$x_{j+1}^3 := (G_j^{33}\ E_j^{33}) \begin{pmatrix} x_j^3 \\ u_j^3 \end{pmatrix} + \tilde{h}_j^3$

Solving for x_{j+1}^3, λ_j^3 means that $-I$ is chosen as a block pivot from $(G_j^{33}\ E_j^{33}\ -I)$ in the elimination process. As a consequence, step 5 is a *non*-orthogonal projection. Its projection operator is given by

$$\bar{P}_j = \begin{pmatrix} I & & \\ & I & \\ G_j^{33} & E_j^{33} & 0 \end{pmatrix} \tag{3.57}$$

on the subspace containing $(x_j^3, u_j^3, x_{j+1}^3)$; it is the identity operator on the complement.

The remaining KKT system in stage j is dense after this projection:

$$\left[\begin{array}{cc|c} Z_j^{xx} & {}^tZ_j^{ux} & {}^tZ_j^{\mu x} \\ Z_j^{ux} & Z_j^{uu} & {}^tZ_j^{\mu u} \\ \hline Z_j^{\mu x} & Z_j^{\mu u} & -Q_{j+1} \end{array}\right] \left[\begin{array}{c} x_j^3 \\ u_j^3 \\ \hline \mu_* \end{array}\right] + \left[\begin{array}{c} z_j^x - \lambda_{j-1}^3 \\ z_j^u \\ \hline z_j^\mu - e_{j-1} \end{array}\right] = 0. \tag{3.58}$$

Recursion step 6 (SC). *Elimination of the projected control minimization conditions by a Schur complement computation.*
Obviously, no projection is possible in (3.58). For further reduction we must apply the SC method using a positive definite (or negative definite) diagonal block as a block pivot. Since the first and the third equation both couple to stage $j-1$, we have to take Z_j^{uu}. Hereby u_j^3 is formally eliminated, see table 3.11. (The block pivot Z_j^{uu} must of course be assumed to be positive definite, see section 3.2.4 below.)

Table 3.11: Step 6 of recursive KKT solver

Factorization	Transformation	Solution
$Z_j^{uu} =: (L_j^{u3})({}^tL_j^{u3})$		
$\bar{Z}_j^{ux} := (L_j^{u3})^{-1} Z_j^{ux}$	$\bar{z}_j^u := -(L_j^{u3})^{-1} z_j^u$	$u_j^3 := -({}^tL_j^{u3})^{-1}[$
$\bar{Z}_j^{\mu u} := Z_j^{\mu u}({}^tL_j^{u3})^{-1}$		
$S_j := -{}^t\bar{Z}_j^{ux}\bar{Z}_j^{ux} + Z_j^{xx}$	$\alpha_j := {}^t\bar{Z}_j^{ux}\bar{z}_j^u + z_j^x$	$\bar{Z}_j^{ux} x_j^3 +$
$R_j := -\bar{Z}_j^{\mu u}\bar{Z}_j^{ux} + Z_j^{\mu x}$	$\beta_j := \bar{Z}_j^{\mu u}\bar{z}_j^u + z_j^\mu$	${}^t\bar{Z}_j^{\mu u}\mu_* - \bar{z}_j^u]$
$Q_j := \bar{Z}_j^{\mu u}\,{}^t\bar{Z}_j^{\mu u} + Q_{j+1}$		

Step 6 completes stage j of the recursion, leaving the projected state minimization condition and global constraints

$$\begin{bmatrix} S_j & {}^tR_j \\ R_j & -Q_j \end{bmatrix} \begin{bmatrix} x_j^3 \\ \mu_* \end{bmatrix} + \begin{bmatrix} \alpha_j - \lambda_{j-1}^3 \\ \beta_j - e_{j-1} \end{bmatrix} = 0. \tag{3.59}$$

This indefinite system couples to stage $j-1$ where it will become the new augmentation.

Recursion step 0 (SC). *Solution of the root system.*
After the final stage $j = 1$, vectors λ_0^3, e_0 vanish and no other equations need to be satisfied. The indefinite root system

$$\begin{bmatrix} S_1 & {}^tR_1 \\ R_1 & -Q_1 \end{bmatrix} \begin{bmatrix} x_1^3 \\ \mu_* \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \beta_1 \end{bmatrix} = 0 \tag{3.60}$$

is then solved directly, yielding initial states x_1^3 and the global multiplier μ_* for the solution phase of the recursion.

3.2.3 Remarks on the recursive algorithm

- An additional SC step 7 could be included in the recursion in order to exploit a possible substructure of the matrix of global constraints (like, e.g., a staircase or step-pyramid structure). This is of special interest when the KKT solver is applied in the context of multibody dynamics, see section 5.3.3.2.

- We have presented each phase of the algorithm as a single recursion consisting of six (or seven) steps in each stage. There are, however, other possible implementations.

  As indicated above, steps 1 and 2 can be extracted as parallel preprocessing (in factorization and transformation phases) or postprocessing (in the solution phase). Furthermore, separate recursions can be formed from steps 3, 4 and steps 5, 6 (and possibly 7), respectively. This would split each phase of the algorithm into one parallel and two recursive modules.

  On the other hand, factorization and transformation phases may be interleaved to form a single backward recursion (or three modules according to the previous observation).

  Finally, the treatment of global constraints can be extracted from each phase to be performed in an independent recursion (or three independent modules).

- When all the constraints are decoupled, then the dynamic programming principle applies as explained in section 3.2.1. The recursive algorithm can be seen to be a true projected Hessian method (see below), and the factorization does not produce any fill-in in the KKT matrix.

  In the presence of global constraints, local optimality according to the dynamic programming principle does not necessarily apply, the algorithm is not a true PH method, and fill-in is accumulated in form of the matrix Q_1. The latter is part of the Schur complement of control minimization blocks Z_j^{uu}, or more precisely, the block associated with projected global constraints.

- Our recursion is closely related to the *multistage backward sweep* algorithm due to McReynolds [64], cf. also Bryson and Ho [25, sections 6.10,7.7]. This algorithm appears as a special case of our recursion, permitting two-point boundary conditions of the form

$$F_1 x_1 + F_m x_m + e_* = \begin{pmatrix} I \\ 0 \end{pmatrix} x_1 + \begin{pmatrix} 0 \\ F_m^m \end{pmatrix} x_m + \begin{pmatrix} e_*^1 \\ e_*^m \end{pmatrix} = 0, \tag{3.61}$$

i.e., fixed initial states $x_1 = -e_*^1$ and any final constraints. Each stage of the backward sweep is equivalent to simplified versions of steps 5 and 6 in our algorithm, therefore we have adopted the notation of [25, section 6.10] in these steps.

Although boundary conditions (3.61) are clearly separated, the final conditions are treated like globally coupled constraints: they are recursively *swept back* through all m stages, yielding equivalent initial conditions. A generalized backward sweep algorithm (i.e., steps 5, 6 of our algorithm) could treat any combination of local and global constraints in that way. However, neglecting the structure is much less efficient, requires more restrictive regularity assumptions and produces more fill-in. Even the $\mathcal{O}(m)$ behavior will be destroyed if there are $\mathcal{O}(m)$ local constraints, which is likely to happen in an active set strategy.

3.2.4 Regularity assumptions

In the following we will discuss the prerequisites that are necessary for the recursive factorization. Regularity assumptions will first be studied locally for each single step of the factorization phase. A local control-theoretical interpretation is always possible, since the algorithm knows the precise m-stage block-sparse structure of the discrete linear-quadratic control problem. If all the necessary conditions hold, it follows that the constraint qualification is satisfied (see section 1.1.3.2), and hence the interpretation carries over to the nonlinear discrete problem (NLP) of chapter 2. Finally, regularity assumptions and interpretation will be formulated from a global point of view.

First we assume that the local and total number of constraints are always less or equal to the local and total number of variables, respectively. Otherwise the QP would have conflicting constraints (or possibly redundant ones or both, depending on the actual right hand side which is not checked during factorization).

Step 1. Factorization step 1 requires full rank of D_j^u, which is equivalent to linear independence of the rows of this constraint matrix. In other words, the QP must not have conflicting (or redundant) local control constraints in any stage.

Step 2. Similarly, step 2 requires full rank of F_j^x, i.e., the QP must not have conflicting (or redundant) local state constraints in any stage.

Steps 3, 4. These steps require full rank of $(C_j \; A_j)$. This means that the feasible manifold $\mathcal{S}_j$ is non-empty, which is equivalent to *attainability* of $\mathcal{S}_{j+1}$. Considering step 3 first, we observe that the kernel $\ker(A_j)$ and range $\operatorname{im}(A_j)$ are determined in the factorization. The dimension of the latter, $\dim \operatorname{im}(A_j) = \operatorname{rank}(L_j^{u2})$ (which may be zero), gives the number of attainability conditions that can be satisfied by suitable choice of the control u_j. This is a measure of local *controllability*. In step 4, $\bar{C}_j^x$ must have full rank, i.e., independent rows. This condition ensures that the remaining attainability conditions can actually be satisfied by suitable choice of the state x_j, which means $\mathcal{S}_j \neq \emptyset$.

Step 5. The non-orthogonal projection of step 5 does not require *any* regularity condition. As a simple consequence of the problem formulation, the block pivot $-I$ is always present. This means that state components x^3_{j+1} (which are not fixed by any local constraints) are simply determined by preceding variables x^3_j, u^3_j.

Step 6. As mentioned before, the Schur complement step 6 requires positive-definiteness of the diagonal block $Z^{uu}_j = K^{33}_j + {}^tE^{33}_j S_{j+1} E^{33}_j$. This is equivalent to local *uniqueness* of the optimal control u_j. (Of course, if Z^{uu}_j is indefinite but non-singular, the KKT system still has a unique solution. Such a solution, however, is a saddle point or maximum of the QP rather than a minimum.) Observe that u^3_j is minimized with respect to a sum of two Hessian terms. The first term, K^{33}_j, gives the direct contribution of u^3_j to the cost function. The second one, ${}^tE^{33}_j S_{j+1} E^{33}_j$, represents the amount that u^3_j contributes indirectly through x^3_{j+1} and the complete tail of the chain.

Step 0. Solution of the root system requires that its matrix

$$\Omega_1 = \begin{pmatrix} S_1 & {}^tR_1 \\ R_1 & -Q_1 \end{pmatrix} \tag{3.62}$$

is non-singular, where S_1, Q_1 must be positive semi-definite. Since the root system carries information on the entire m-stage KKT system, and in particular on the global constraints, a local interpretation of this condition is not possible. For a global interpretation consider a QP without any boundary conditions, or equivalently, the QP resulting from the recursion after steps 1–4. Each stage of the associated KKT system has the form (3.54), i.e., we minimize states and controls subject to continuity conditions and global constraints. Now, generating the root system recursively by steps 5 and 6 can be shown to be equivalent to the following two operations.

(1) Projection onto the kernel of *all* continuity conditions by eliminating $x_2, \ldots, x_m$.

(2) Formal elimination of $u_{m-1}, \ldots, u_1$ (i.e., in reverse order) by forming Schur complements of the associated diagonal blocks of the projected Hessian.

In fact, the Hessian blocks that are factorized in (2) are exactly the matrices Z^{uu}_j. However, the global projection (1) (known as *condensing*, see [69, 20]) produces a *completely dense* projected Hessian. Here the fill-in consists of certain intermediate results most of which are not even generated in the recursive approach.

Hence, Ω_1 is non-singular if and only if the matrix of global constraints has full rank (even after projection on all the local ones—including continuity conditions), and the Hessian is positive definite on the kernel of *all* (local and global) constraints.

Formulation of the global regularity assumptions is now very easy. The recursive algorithm requires:

(A) The full set of constraints is linearly independent.

(B) The Hessian is positive definite on the kernel of all constraints.

(C) The part of the Hessian associated with control variables is positive definite on the kernel of local constraints.

Conditions (A,B) are simply the sufficient conditions for a unique minimum. Condition (C) is slightly more restrictive, but empty in the absence of global constraints. In the latter case the recursion is a true PH algorithm.

3.2.5 Elementary factors

We will not go into detail here, but only point out the overall structure of the KKT matrix factorization. Four levels have to be considered. On the top level, the symmetric factorization is simply

$$\Omega = \Lambda\Pi\Lambda^T. \tag{3.63}$$

On the next level, these factors are further decomposed according to stages 1–m,

$$\Lambda = \Lambda_m \dots \Lambda_1, \tag{3.64}$$
$$\Pi = \Pi_m \dots \Pi_1. \tag{3.65}$$

On the level of stages, the decomposition involves one factor for each step,

$$\Lambda_j = \Lambda_j^1 \dots \Lambda_j^6, \tag{3.66}$$
$$\Pi_j = \Pi_j^1 \dots \Pi_j^6. \tag{3.67}$$

On the level of recursion steps, these matrices are finally decomposed into elementary factors:

$$\Lambda_j^1 = \Lambda_j^{11}\Lambda_j^{12}, \tag{3.68}$$
$$\Lambda_j^2 = \Lambda_j^{21}\Lambda_j^{22}\Lambda_j^{23}\Lambda_j^{24}, \tag{3.69}$$
$$\Lambda_j^3 = \Lambda_j^{31}\Lambda_j^{32}\Lambda_j^{33}\Lambda_j^{34}\Lambda_j^{35}\Lambda_j^{36}, \tag{3.70}$$
$$\Lambda_j^4 = \Lambda_j^{41}\Lambda_j^{42}, \tag{3.71}$$
$$\Lambda_j^5 = \Lambda_j^{51}\Lambda_j^{52}, \tag{3.72}$$
$$\Lambda_j^6 = \Lambda_j^{61}\Lambda_j^{62}. \tag{3.73}$$

The factors Π_j^i are not decomposed any further, except for Π_1^1 which represents the root system. Types and nontrivial blocks of the elementary factors are not given here; they can be deduced from tables 3.3–3.11. Note that multiplication by Π^{-1} is split up between transformation and solution phases rather than being performed in a separate phase.

3.2.6 A naming scheme for structured factorizations

In order to distinguish the recursive algorithm from several other $\mathcal{O}(m)$ algorithms that will be developed below, we use the notation

$$(\mathrm{PH})_2^m[(\mathrm{PH})_3(\mathrm{SC})]_m(\mathrm{SC}) \tag{3.74}$$

to describe its structure in short terms. Here (PH) and (SC) denote projected Hessian or Schur complement steps, respectively, superscript m indicates that the operations are performed independently in m stages (possibly in parallel), and subscript k indicates that k operations of the same type are performed in sequence. According to this, we read (3.74) as follows: Perform two PH steps in each stage (eliminating local control and state constraints), then perform m times a sequence of three PH steps and one SC step (the body of the recursion, steps 3–6), and conclude the algorithm by one SC step (solving the root system). Of course, the order of operations is reversed in the solution phase. When we wish to emphasize that steps 1 and 2 are actually included in the recursion, we may use the notation $[(\mathrm{PH})_5(\mathrm{SC})]_m(\mathrm{SC})$.

3.3 Alternative $\mathcal{O}(m)$ algorithms for separated constraints

A rather important special case is the situation that boundary conditions are linearly decoupled. That is, no global constraints are present in (3.34), and hence the constraint matrix F is block-diagonal. A variety of $\mathcal{O}(m)$ KKT solvers will be developed for this case. We drop the distinction between state and control variables and simplify partitionings (3.32–3.34). In the simplified notation we replace Hessian blocks, variable and gradient vectors by

$$H_j \leftarrow \begin{pmatrix} H_j & {}^tJ_j \\ J_j & K_j \end{pmatrix}, \qquad x_j \leftarrow \begin{pmatrix} x_j \\ u_j \end{pmatrix}, \qquad f_j \leftarrow \begin{pmatrix} f_j \\ d_j \end{pmatrix}, \tag{3.75}$$

transition and coupling matrices by

$$G_j \leftarrow (G_j \ E_j), \qquad P_{j+1} \leftarrow (-I \ \ 0), \tag{3.76}$$

and finally constraint matrices, vectors, and multipliers by

$$F_j \leftarrow \begin{pmatrix} F_j^x & \\ & D_j^u \\ F_j^c & D_j^c \end{pmatrix}, \qquad e_j \leftarrow \begin{pmatrix} e_j^x \\ e_j^u \\ e_j^c \end{pmatrix}, \qquad \mu_j \leftarrow \begin{pmatrix} \mu_j^x \\ \mu_j^u \\ \mu_j^c \end{pmatrix}. \tag{3.77}$$

In the following we will not rely any more on the special substructure in $(-I \ \ 0)$; any full-rank coupling matrix P_{j+1} is admissible.

3.3.1 Band solver

It is easy to see that the nonzero blocks of the KKT matrix Ω may be arranged along the diagonal so that Ω has a natural block-banded structure. The decoupled m-stage block-sparse KKT system then reads

$$\begin{bmatrix} H_1 & {}^tF_1 & {}^tG_1 & & & & & & & \\ F_1 & & & & & & & & & \\ G_1 & & & P_2 & & & & & & \\ & & {}^tP_2 & H_2 & {}^tF_2 & {}^tG_2 & & & & \\ & & & F_2 & & & & & & \\ & & & G_2 & & & & & & \\ & & & & & & \ddots & P_m & & \\ & & & & & & {}^tP_m & H_m & {}^tF_m & {}^tG_m \\ & & & & & & & F_m & & \\ & & & & & & & G_m & & \end{bmatrix} \begin{bmatrix} x_1 \\ \mu_1 \\ \lambda_1 \\ x_2 \\ \mu_2 \\ \lambda_2 \\ \vdots \\ x_m \\ \mu_m \\ \lambda_m \end{bmatrix} + \begin{bmatrix} f_1 \\ e_1 \\ h_1 \\ f_2 \\ e_2 \\ h_2 \\ \vdots \\ f_m \\ e_m \\ h_m \end{bmatrix} = 0. \tag{3.78}$$

As in the previous section, we assume that a unique solution exists. The KKT system can be treated by any standard band solver (LU factorization with partial pivoting) such as, e.g., the LAPACK band solver implemented in routines DGBTRx. (According to the LAPACK naming scheme, letters 'GB' denote matrix type 'general band', see [4].) Note that the symmetry of Ω can *not* be exploited since pivoting is necessary due to the indefiniteness of Ω. The band solver will be referred to as $(\text{GB})_m$.

3.3.2 Structured PH methods

As a consequence of the decoupling, exactly one possibility exists to reduce the size of system 3.78 by a projection without destroying the structure of Ω. It is the projection onto the kernel of the block-diagonal constraint matrix F. Here we develop a family of structured solution algorithms that begin with this projection. The operations of the initial PH step are given in table 3.12.

Obviously, the projection is suitable for parallel processing on m processors. Data transmission occurs only in the transformation and solution phases. In a sequential implementation, the transformation phase runs backward and the solution phase runs forward.

The PH step yields a KKT system which is still indefinite and banded,

$$\begin{bmatrix} H_1^{22} & {}^tG_1^2 & & & & & & \\ G_1^2 & & P_2^2 & & & & & \\ & {}^tP_2^2 & H_2^{22} & {}^tG_2^2 & & & & \\ & & G_2^2 & & & & & \\ & & & & \ddots & P_m^2 & & \\ & & & & {}^tP_m^2 & H_m^{22} & {}^tG_m^2 \\ & & & & & G_m^2 & \end{bmatrix} \begin{bmatrix} x_1^2 \\ \lambda_1 \\ x_2^2 \\ \lambda_2 \\ \vdots \\ x_m^2 \\ \lambda_m \end{bmatrix} + \begin{bmatrix} f_1^2 \\ h_1^2 \\ f_2^2 \\ h_2^2 \\ \vdots \\ f_m^2 \\ h_m^2 \end{bmatrix} = 0. \tag{3.79}$$

Table 3.12: Step 1 of structured PH methods

Factorization	Transformation	Solution
$F_j =: (L_j^1 \ 0)U_j$	$x_j^1 := -(L_j^1)^{-1}e_j$	$\mu_j := -({}^tL_j^1)^{-1}[$
$\begin{pmatrix} H_j^{11} & {}^tH_j^{21} \\ H_j^{21} & H_j^{22} \end{pmatrix} := {}^tU_j^{-1}H_jU_j^{-1}$	$\begin{pmatrix} \tilde{f}_j^1 \\ \tilde{f}_j^2 \end{pmatrix} := {}^tU_j^{-1}f_j$	$\boxed{x_j := U_j^{-1}} \begin{pmatrix} x_j^1 \\ x_j^2 \end{pmatrix}$
	$f_j^1 := H_j^{11}x_j^1 + \tilde{f}_j^1$	
	$f_j^2 := H_j^{21}x_j^1 + \tilde{f}_j^2$	${}^tH_j^{21}x_j^2 +$
$(G_j^1 \ G_j^2) := G_jU_j^{-1}$	$h_j^2 := G_j^1x_j^1 +$	${}^tG_j^1\lambda_j +$
$(P_j^1 \ P_j^2) := P_jU_j^{-1}$	$P_{j+1}^1x_{j+1}^1 + h_j$	${}^tP_j^1\lambda_{j-1} + f_j^1]$

The first member of our family simply applies a band solver to this reduced system. According to the naming scheme introduced above, the resulting method will be denoted by $(\mathrm{PH})^m(\mathrm{GB})_m$.

Further structure-preserving reductions are possible since all the diagonal blocks H_j^{22} of the Hessian are positive definite by assumption. A subsequent SC step can be performed to eliminate the states x_j^2, see table 3.13. Operations may again be performed in parallel, where data transmission is necessary in all three phases.

Table 3.13: Step 2 of structured PH methods

Factorization	Transformation	Solution
$H_j^{22} =: (L_j^2)({}^tL_j^2)$	$\bar{h}_j := -h_j^2$	$x_j^2 := -\bar{x}_j$
$\bar{G}_j := G_j^2({}^tL_j^2)^{-1}$		
$\bar{P}_j := P_j^2({}^tL_j^2)^{-1}$	$\bar{f}_j := (L_j^2)^{-1}f_j^2$	$\bar{x}_j := ({}^tL_j^2)^{-1}[$
$A_j := \bar{G}_j{}^t\bar{G}_j + \bar{P}_{j+1}{}^t\bar{P}_{j+1}$	$a_j := \bar{G}_j\bar{f}_j +$	${}^t\bar{G}_j\lambda_j +$
$B_j := \bar{G}_j{}^t\bar{P}_j$	${}^t\bar{P}_{j+1}\bar{f}_{j+1} + \bar{h}_j$	${}^t\bar{P}_j\lambda_{j-1} + \bar{f}_j]$

Only the multipliers of the continuity conditions are now left in the reduced system which has a block-tridiagonal structure. The system matrix is positive definite due to a sign change that was included in the previous SC step,

$$\begin{bmatrix} A_1 & {}^tB_2 & & \\ B_2 & A_2 & & \\ & & \ddots & {}^tB_m \\ & & B_m & A_m \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_m \end{bmatrix} + \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = 0. \tag{3.80}$$

One of three methods may be chosen for the solution of this system, leading to the other three members of our family. Method $(\mathrm{PH})^m(\mathrm{SC})^m(\mathrm{GB})_m$ applies a general band solver in the final step, $(\mathrm{PH})^m(\mathrm{SC})^m(\mathrm{PB})_m$ applies a symmetric band solver, and method $(\mathrm{PH})^m(\mathrm{SC})^m(\mathrm{TD})_m$ uses a block-tridiagonal solver. Here 'PB' and 'TD' indicate matrix types 'positive definite band' and 'block-tridiagonal', respectively. The associated solvers are based on Cholesky factorizations which exploit the banded or block-tridiagonal structure.

Note that the last row and column of blocks in equations (3.78–3.80) are only formally included for notational convenience. Actually G_m, λ_m, h_m and the derived quantities G_m^2, h_m^2 and A_m, B_m, a_m have all dimension zero, and the block-tridiagonal system (3.80) has actually only $m-1$ stages.

3.3.3 Structured SC methods

We conclude this chapter with the presentation of a family of structured algorithms that apply a Schur complement step first. These methods require positive-definiteness of all the Hessian blocks H_j. The initial SC step is given in table 3.14.

The resulting system is positive definite (due to a sign change) and banded,

$$\begin{bmatrix} D_1 & {}^tE_1 & & & & & \\ E_1 & \tilde{A}_1 & {}^t\tilde{B}_2 & {}^tC_2 & & & \\ & \tilde{B}_2 & D_2 & {}^tE_2 & & & \\ & C_2 & E_2 & \tilde{A}_2 & & & \\ & & & & \ddots & {}^t\tilde{B}_m & {}^tC_m \\ & & & & \tilde{B}_m & D_m & {}^tE_m \\ & & & & C_m & E_m & \tilde{A}_m \end{bmatrix} \begin{bmatrix} \mu_1 \\ \lambda_1 \\ \mu_2 \\ \lambda_2 \\ \vdots \\ \mu_m \\ \lambda_m \end{bmatrix} + \begin{bmatrix} b_1 \\ \tilde{a}_1 \\ b_2 \\ \tilde{a}_2 \\ \vdots \\ b_m \\ \tilde{a}_m \end{bmatrix} = 0. \tag{3.81}$$

It can be treated by a general or symmetric band solver, giving members $(\mathrm{SC})^m(\mathrm{GB})_m$ and $(\mathrm{SC})^m(\mathrm{PB})_m$ of our second family.

Since all the constraint matrices F_j have full rank, their Schur complements D_j are positive definite, and a second SC step can be applied to eliminate the corresponding block rows and columns. This transformation yields exactly system (3.80), and hence the final solution steps are identical to the structured PH methods above. Thus, we arrive at methods $(\mathrm{SC})^m(\mathrm{SC})^m(\mathrm{GB})_m$, $(\mathrm{SC})^m(\mathrm{SC})^m(\mathrm{PB})_m$, and $(\mathrm{SC})^m(\mathrm{SC})^m(\mathrm{TD})_m$.

Table 3.14: Step 1 of structured SC methods

Factorization	Transformation	Solution
$H_j =: L_j {}^tL_j$		$x_j := -\bar{x}_j$
$\bar{F}_j := F_j {}^tL_j^{-1}$	$\bar{e}_j := -e_j$	
$\bar{G}_j := G_j {}^tL_j^{-1}$	$\bar{h}_j := -h_j$	
$\bar{P}_j := P_j {}^tL_j^{-1}$	$\bar{f}_j := L_j^{-1} f_j$	$\bar{x}_j := {}^tL_j^{-1}[$
$\tilde{B}_j := \bar{F}_j {}^t\bar{P}_j$		
$D_j := \bar{F}_j {}^t\bar{F}_j$	$b_j := \bar{F}_j \bar{f}_j + \bar{e}_j$	${}^t\bar{F}_j \mu_j +$
$E_j := \bar{G}_j {}^t\bar{F}_j$		
$C_j := \bar{G}_j {}^t\bar{P}_j$	$\tilde{a}_j := \bar{G}_j \bar{f}_j +$	${}^t\bar{G}_j \lambda_j +$
$\tilde{A}_j := \bar{G}_j {}^t\bar{G}_j + \bar{P}_{j+1} {}^t\bar{P}_{j+1}$	$\bar{P}_{j+1} \bar{f}_{j+1} + \bar{h}_j$	${}^t\bar{P}_j \lambda_{j-1} + \bar{f}_j]$

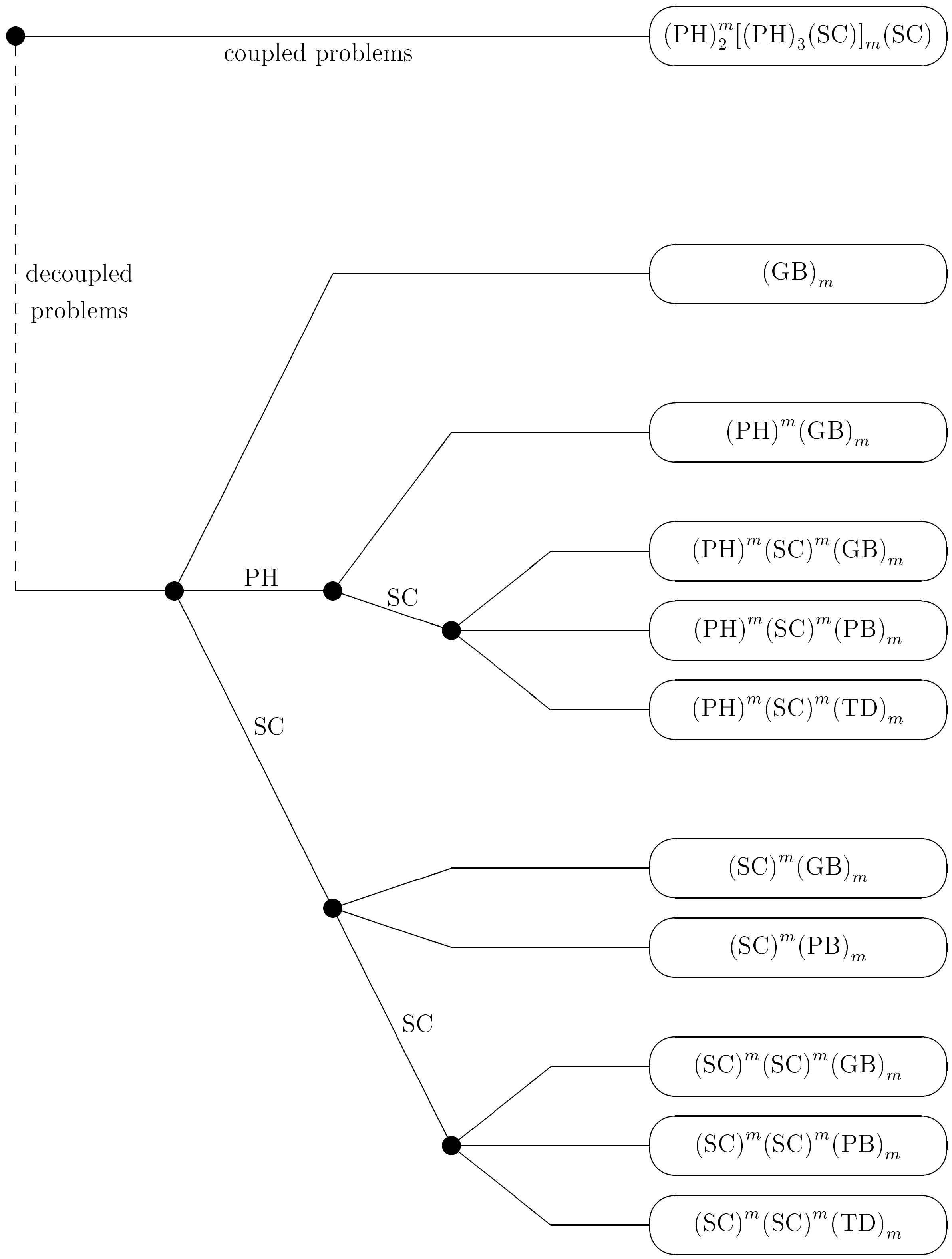
3.4 Summary

In this chapter we have developed a recursive solution algorithm for the general multistage KKT system, and a whole family of structured algorithms for the special case of decoupled constraints. All these algorithms have linear complexity $\mathcal{O}(m)$; a complete overview is given in figure 3.5.

The family of SC algorithms is given for completeness reasons only. Each of its members except $(\mathrm{SC})^m(\mathrm{PB})_m$ has a corresponding algorithm in the PH family that differs only in the initial elimination step. The latter is, however, about equally efficient but more stable, and therefore preferable. It can be expected that the algorithms show gradually increasing speed and decreasing stability in the order $(\mathrm{GB})_m$, $(\mathrm{PH})^m(\mathrm{GB})_m$, $(\mathrm{PH})^m(\mathrm{SC})^m(\mathrm{GB})_m$, $(\mathrm{PH})^m(\mathrm{SC})^m(\mathrm{PB})_m$, $(\mathrm{PH})^m(\mathrm{SC})^m(\mathrm{TD})_m$, $(\mathrm{PH})_2^m[(\mathrm{PH})_3(\mathrm{SC})]_m(\mathrm{SC})$. But numerical results in chapter 5 indicate that even $(\mathrm{GB})_m$ is not much more accurate than the recursive solver, so that no actual need for the other algorithms can be seen yet.

Full or partial exploitation of the m-stage block-sparse structure makes these algorithms not only very efficient, it also has another important consequence. As we have seen in the case of the recursive algorithm, regularity of the KKT matrix can be deduced from certain properties of the optimal control problem. Vice versa, if one of the structured solvers finds the KKT matrix singular, one can exactly "locate" the rank-deficiency and give a higher-level interpretation. The singularity can be traced back through the QP and discrete NLP, so that as much information as possible is available to find the reason in the original optimal control problem. Therefore these structured algorithms are very well suited for analyzing the control problem when difficulties arise. This is not true for even the best general-purpose sparse solvers.

Figure 3.5: The complete zoo of $\mathcal{O}(m)$ KKT solvers



Chapter 4

A structured interior point method for the QP

The concept of interior point or barrier methods for constrained optimization has been introduced by Frisch [39] and theoretically developed by Fiacco and McCormick [35]. After the appearance of Karmarkar's paper [51] barrier methods for linear programs have been developed by many authors, and extended to convex quadratic programs shortly thereafter. See, e.g., [66, 95, 44, 41, 89, 90, 87, 88]. A coarse classification of barrier methods (both for LP and QP cases) includes the primal, primal-dual, and predictor-corrector methods.

In general we suggest a primal-dual barrier method, as described below, for QP solution within our structured approach to optimal control problems. However, an active set strategy with warm start may be competitive, especially when a good estimate of the correct active set is known and only a few exchange steps are expected. Note that large numbers of binding box constraints reduce the effort of recursive KKT solution, since such constraints are pre-eliminated in recursion steps 1 and 2, and changes of the active set can be handled efficiently by a Schur complement method [42]. Improving a given solution by mesh refinement may therefore be very efficient with an active set strategy.

In the following we outline an interior point method for the general QP (2.29–2.31). No assumptions on the internal structure of the problem matrices are made, and the barrier approach is developed on the given coarse structural level. We will see that the m-stage block-sparse QP structure in the optimal control context is automatically preserved. Finally the treatment of inequalities is refined, again without making any assumptions on the structure of the matrices involved.

4.1 The QP with inequalities

Let us consider the general QP (2.29–2.31), but drop superscript k and replace Δy by y (we are not interested in the higher level SQP iteration at the moment):

$$[\mathrm{QP}]\qquad \begin{array}{rrcl} & \frac{1}{2}y^T H y + J_1 y & = & \min\limits_{y} \\ \text{subject to} & J_2 y + F_2 & = & 0, \\ & J_3 y + F_3 & \geq & 0. \end{array} \qquad \begin{array}{r}(4.1)\\(4.2)\\(4.3)\end{array}$$

Throughout this chapter we assume that the QP has a unique, stable minimum according to theorem 7. The Lagrangian of [QP] is

$$L(y, w_2, w_3) = \frac{1}{2}y^T H y + J_1 y - w_2^T(J_2 y + F_2) - w_3^T(J_3 y + F_3), \tag{4.4}$$

and its dual problem reads

$$[\mathrm{D}_0]\qquad \begin{array}{rrcl} & -\frac{1}{2}y^T H y - F_2^T w_2 - F_3^T w_3 & = & \max\limits_{y,w_2,w_3} \\ \text{subject to} & Hy + J_1^T - J_2^T w_2 - J_3^T w_3 & = & 0, \\ & w_3 & \geq & 0. \end{array} \qquad \begin{array}{r}(4.5)\\(4.6)\\(4.7)\end{array}$$

(Here we have inserted (4.6) into the Lagrangian to get the simplified version in (4.5).) As a first step towards the barrier formulation we introduce nonnegative slack variables $s \geq 0$ and reformulate the primal QP:

$$[\mathrm{P}_0]\qquad \begin{array}{rrcl} & \frac{1}{2}y^T H y + J_1 y & = & \min\limits_{y} \\ \text{subject to} & J_2 y + F_2 & = & 0, \\ & -s + J_3 y + F_3 & = & 0, \\ & s & \geq & 0. \end{array} \qquad \begin{array}{r}(4.8)\\(4.9)\\(4.10)\\(4.11)\end{array}$$

The corresponding dual problem with slacks is not identical but of course equivalent to $[\mathrm{D}_0]$. Since the exact form is unimportant here, we regard $[\mathrm{D}_0]$ as the dual of $[\mathrm{P}_0]$.

Assume that (y, s) and (y, w_2, w_3) are primal and dual feasible, satisfying (4.9–4.11) and (4.6,4.7), respectively. It is well-known from duality theory that the objective of $[\mathrm{P}_0]$ is then greater or equal to the objective of $[\mathrm{D}_0]$, and their difference is given as the *duality gap* $s^T w_3 \geq 0$. At the optimal solution the duality gap vanishes and *complementary slackness* holds, $s^T w_3 = 0$.

4.2 Barrier formulation of the QP

Considering strictly positive slacks $s > 0$ only, we replace nonnegativity constraints by a logarithmic barrier potential term with potential parameter $\mu > 0$, thus converting $[\mathrm{P}_0]$ to the unconstrained problem

$$[\mathrm{P}_\mu] \qquad \begin{array}{rcll} \frac{1}{2}y^T H y + J_1 y - \mu \sum_i \ln s_i & = & \min\limits_{y,s} & (4.12) \\ \text{subject to} \qquad J_2 y + F_2 & = & 0, & (4.13) \\ -s + J_3 y + F_3 & = & 0, & (4.14) \end{array}$$

with Lagrangian

$$\begin{aligned} L(y, s, w_2, w_3; \mu) \;\; = \;\; & \frac{1}{2}y^T H y + J_1 y - \mu \sum\nolimits_i \ln s_i \\ & - w_2^T (J_2 y + F_2) - w_3^T (J_3 y + F_3 - s). \end{aligned} \tag{4.15}$$

The barrier problem $[\mathrm{P}_\mu]$ has a unique minimum (y_μ, s_μ) for each parameter value $\mu > 0$. The set of primal solutions (y_μ, s_μ) is called the *central path*; it depends analytically on μ. As $\mu \to 0$, the central path approaches the solution of the original problem $[\mathrm{P}_0]$. In addition, if the polytope defined by $J_3 y + F_3 \geq 0$ is bounded, then y_μ converges to the *analytic center* of this polytope for $\mu \to \infty$.

4.3 Necessary conditions

Derivatives of the Lagrangian with respect to each of the variables y, s, w_2, w_3 yield the first order optimality conditions for problem $[\mathrm{P}_\mu]$,

$$Hy + J_1^T - J_2^T w_2 - J_3^T w_3 = 0, \tag{4.16}$$

$$-\mu S^{-1} e + w_3 = 0, \tag{4.17}$$

$$J_2 y + F_2 = 0, \tag{4.18}$$

$$-s + J_3 y + F_3 = 0. \tag{4.19}$$

As usual in this context, $e = (1, \ldots, 1)$ denotes the vector of all ones, and $S = \mathrm{diag}(\{s_i\})$ is the diagonal matrix composed from the components of vector s. When the name of a vector is capitalized without explanation, this will always refer to the corresponding diagonal matrix in the following.

Equations (4.16) and (4.18,4.19) are linear; they represent dual and primal feasibility, respectively. The *μ-complementarity* condition (4.17) is characteristic for the barrier approach. It is the only nonlinear condition and the only one that involves the barrier parameter μ. Actually it represents a relaxation of the complementarity condition, $s^T w_3 = 0$ or $s_i w_{3i} = 0 \; \forall i$. This becomes immediately clear when we replace (4.17) by the equivalent form

$$SW_3 e = \mu e \tag{4.17'}$$

or $s_i w_{3i} = \mu \; \forall i$. When (y, s, w_2, w_3) satisfy the necessary conditions for any parameter μ, then (y, s) are primal feasible, (y, w_2, w_3) are dual feasible, and the duality gap is given by $s^T w_3 = e^T SW_3 e = n_3 \mu > 0$. For $\mu \to 0$ the original complementarity condition is retained.

Due to the nonlinearity of the μ-complementarity condition (4.17′) one cannot solve the system of necessary conditions directly. Instead, one or more Newton steps are taken, a new barrier parameter μ is chosen, and the procedure is repeated until a suitable convergence criterion is satisfied. The Newton increment $(\delta y, \delta s, \delta w_2, \delta w_3)$ is defined as solution of the KKT system

$$\left[\begin{array}{cc|cc} H & & J_2^T & J_3^T \\ & \Theta & & -I \\ \hline J_2 & & & \\ J_3 & -I & & \end{array}\right] \left[\begin{array}{c} \delta y \\ \delta s \\ \hline -\delta w_2 \\ -\delta w_3 \end{array}\right] + \left[\begin{array}{l} Hy + J_1^T - J_2^T w_2 - J_3^T w_3 \\ \vartheta \\ \hline J_2 y + F_2 \\ J_3 y + F_3 - s \end{array}\right] = 0, \tag{4.20}$$

where $\Theta = S^{-1}W_3$ and $\vartheta = w_3 - \mu S^{-1}e$. (If the original μ-complementarity condition is linearized instead of (4.17′), one must define $\Theta = \mu S^{-2}$). In order to solve the KKT system we eliminate δs and $-\delta w_3$ using the fourth and the second equation,

$$\begin{aligned} \delta s &= J_3 \delta y + (J_3 y + F_3 - s), && (4.21) \\ -\delta w_3 &= \Theta \delta s + \vartheta. && (4.22) \end{aligned}$$

Furthermore we define $\tau = \vartheta + \Theta(J_3 y + F_3 - s)$ and

$$\begin{aligned} \hat{H} &= H + J_3^T \Theta J_3, && (4.23) \\ \hat{w}_3 &= w_3 - \tau. && (4.24) \end{aligned}$$

This reduces the KKT system to

$$\begin{bmatrix} \hat{H} & J_2^T \\ J_2 & \end{bmatrix} \begin{bmatrix} \delta y \\ -\delta w_2 \end{bmatrix} + \begin{bmatrix} Hy + J_1^T - J_2^T w_2 - J_3^T \hat{w}_3 \\ J_2 y + F_2 \end{bmatrix} = 0. \tag{4.25}$$

System (4.25) has exactly the same overall structure as the system of optimality conditions for a QP *without* inequality constraints. The only difference due to the inequalities is the augmentation $J_3^T \Theta J_3$ of the Hessian, and of course the corresponding term $-J_3^T \hat{w}_3$ in the right hand side.

Let us now consider the case where [QP] has the m-stage block-sparse substructure that results from a control problem discretization according to theorem 9. This structure is preserved under the previously described transformation, and hence system (4.25) can be solved by the recursive algorithm developed in chapter 3:

Theorem 10 *The augmented Hessian $\hat{H} = H + J_3^T \Theta J_3$ has precisely the same block-diagonal structure as H.*

Proof. Observe that Θ is a diagonal matrix, $\Theta = S^{-1}W_3$. Hence the assertion is an immediate consequence of the block-diagonal structures of H and J_3 as established in theorem 9. ∎

4.4 A refined barrier approach

In the following we present a refined version of the previously described interior point approach. In specific, upper and lower bounds and ranges will be distinguished among the inequality constraints in order to allow for a better exploitation of their substructure. This approach does not only lead to improved numerical algorithms, it also shows a closer connection to other common formulations (which are usually developed from an LP viewpoint). Our formulation is based on the approach of Vanderbei et al. [89, 88] who treat several LP and QP types. While maintaining their high degrees of decoupling and symmetry, we avoid typical LP conventions such as making all variables nonnegative or treating equality and inequality constraints alike. A comparison with our previous coarse approach and with the formulation in [89, 88] concludes this chapter.

Assume that we are given lower and upper bounds as well as lower and upper ranges,

$$b_l \le \quad y \le b_u, \tag{4.26}$$

$$r_l \le By \le r_u, \tag{4.27}$$

where the limits may include $\pm\infty$ in the case of unconstrained components. Variables are not necessarily nonnegative (i.e., $b_l \neq 0$ in general), and we do not subsume equality constraints here (i.e., $b_l < b_u$ and $r_l < r_u$). All the equality constraints are included in (4.2), and inequalities (4.3) now have a substructure

$$J_3 y + F_3 = \begin{pmatrix} I \\ -I \\ B \\ -B \end{pmatrix} y + \begin{pmatrix} -b_l \\ b_u \\ -r_l \\ r_u \end{pmatrix} \ge 0, \tag{4.28}$$

yielding similar partitionings

$$s = (s_l, s_u, t_l, t_u), \qquad \vartheta = (\phi_l, \phi_u, \psi_l, \psi_u), \tag{4.29}$$

$$w_3 = (u_l, u_u, v_l, v_u), \qquad \Theta = \operatorname{diag}(\Phi_l, \Phi_u, \Psi_l, \Psi_u). \tag{4.30}$$

When we set $A = J_2$, the KKT matrix of (4.20) becomes

$$\left[\begin{array}{ccccc|ccccc}
H & & & & & A^T & I & -I & B^T & -B^T \\
& \Phi_l & & & & & -I & & & \\
& & \Phi_u & & & & & -I & & \\
& & & \Psi_l & & & & & -I & \\
& & & & \Psi_u & & & & & -I \\
\hline
A & & & & & & & & & \\
I & -I & & & & & & & & \\
-I & & -I & & & & & & & \\
B & & & -I & & & & & & \\
-B & & & & -I & & & & &
\end{array}\right], \tag{4.31}$$

and the augmented Hessian of the previous “coarse formulation” reads

$$\hat{H} = H + (\Phi_l + \Phi_u) + B^T(\Psi_l + \Psi_u)B. \tag{4.32}$$

Note that infinite bounds or ranges have infinite slacks, the reciprocals of which produce zero diagonal elements in the corresponding Φ or Ψ matrix. Thus, as should be expected, free components do not contribute to $\hat{H}$.

In (4.32) the diagonal term due to the bounds and the general Schur complement term due to the ranges are both added to the Hessian. The first of them does not cause any difficulties, but adding $B^T(\Psi_l + \Psi_u)B$ makes the solution process for the KKT system potentially unstable. This Schur complement term might even destroy the sparseness of H in the general case (when no assumptions are made about the specific sparse structure of matrices A, B, H). Therefore we will not form system (4.25), but prefer a larger KKT system with B left unchanged. However, B should not appear twice (for lower *and* upper ranges) as in (4.31). Hence, after introducing slacks, we eliminate By from the equation for upper ranges,

$$\begin{aligned} By - r_l - t_l &= 0, && (4.33)\\ r_u - t_u - r_l - t_l &= 0. && (4.34) \end{aligned}$$

The bound equations are transformed similarly for symmetry reasons.

In the following we set $z = w_2$ and denote the right hand side vectors of the KKT system by

$$\begin{aligned} h &= Hy + J_1^T - A^T z - u_l - B^T v_l, && (4.35)\\ a &= Ay + F_2, && (4.36) \end{aligned}$$

and

$$\begin{aligned} \beta_l &= y - b_l - s_l, \qquad & \beta_u &= b_u - s_u - b_l - s_l, && (4.37)\\ \rho_l &= By - r_l - t_l, \qquad & \rho_u &= r_u - t_u - r_l - t_l. && (4.38) \end{aligned}$$

This yields the modified KKT system

$$\left[\begin{array}{ccccc|ccccc} H & & & & & A^T & I & & B^T & \\ & \bar{\Phi}_l & & & & & -I & -I & & \\ & & \Phi_u & & & & & -I & & \\ & & & \bar{\Psi}_l & & & & & -I & -I \\ & & & & \Psi_u & & & & & -I \\ \hline A & & & & & & & & & \\ I & -I & & & & & & & & \\ & -I & -I & & & & & & & \\ B & & & -I & & & & & & \\ & & & -I & -I & & & & & \end{array}\right] \left[\begin{array}{c} \delta y \\ \delta s_l \\ \delta s_u \\ \delta t_l \\ \delta t_u \\ \hline -\delta z \\ -\delta u_l \\ -\delta u_u \\ -\delta v_l \\ -\delta v_u \end{array}\right] + \left[\begin{array}{c} h \\ \bar{\phi}_l \\ \phi_u \\ \bar{\psi}_l \\ \psi_u \\ \hline a \\ \beta_l \\ \beta_u \\ \rho_l \\ \rho_u \end{array}\right] = 0, \tag{4.39}$$

where

$$\bar{\Phi}_l = S_l^{-1}(U_l + U_u), \qquad \bar{\phi}_l = -\mu S_l^{-1} e + (u_l + u_u), \tag{4.40}$$
$$\bar{\Psi}_l = T_l^{-1}(V_l + V_u), \qquad \bar{\psi}_l = -\mu T_l^{-1} e + (v_l + v_u), \tag{4.41}$$
$$\Phi_u = S_u^{-1} U_u, \qquad \phi_u = -\mu S_u^{-1} e + u_u, \tag{4.42}$$
$$\Psi_u = T_u^{-1} V_u, \qquad \psi_u = -\mu T_u^{-1} e + v_u. \tag{4.43}$$

Obviously the partial reformulation of the bound and range equations leads to a coupling of the associated dual variables in the μ-complementarity condition.

We can now combine lower and upper bounds and ranges, respectively, in order to obtain a smaller set of equations. Elimination of

$$\delta s_u = -\delta s_l + \beta_u, \qquad -\delta u_u = \Phi_u \delta s_u + \phi_u, \tag{4.44}$$
$$\delta t_u = -\delta t_l + \rho_u, \qquad -\delta v_u = \Psi_u \delta t_u + \psi_u \tag{4.45}$$

reduces the system to

$$\left[\begin{array}{ccc|ccc} H & & & A^T & I & B^T \\ & \Phi & & & -I & \\ & & \Psi & & & -I \\ \hline A & & & & & \\ I & -I & & & & \\ B & & -I & & & \end{array}\right] \left[\begin{array}{c} \delta y \\ \delta s_l \\ \delta t_l \\ \hline -\delta z \\ -\delta u_l \\ -\delta v_l \end{array}\right] + \left[\begin{array}{c} h \\ \phi \\ \psi \\ \hline a \\ \beta_l \\ \rho_l \end{array}\right] = 0, \tag{4.46}$$

where

$$\Phi = \bar{\Phi}_l + \Phi_u, \qquad \phi = \bar{\phi}_l - \phi_u - \Phi_u \beta_u, \tag{4.47}$$
$$\Psi = \bar{\Psi}_l + \Psi_u, \qquad \psi = \bar{\psi}_l - \psi_u - \Psi_u \rho_u. \tag{4.48}$$

Finally, elimination of

$$\delta s_l = \delta y + \beta_l, \qquad -\delta u_l = \Phi \delta s_l + \phi, \tag{4.49}$$
$$\delta t_l = \Psi^{-1}(-\delta v_l - \psi) \tag{4.50}$$

yields

$$\left[\begin{array}{c|cc} H + \Phi & A^T & B^T \\ \hline A & & \\ B & & -\Psi^{-1} \end{array}\right] \left[\begin{array}{c} \delta y \\ \hline -\delta z \\ -\delta v_l \end{array}\right] + \left[\begin{array}{c} h + \Phi \beta_l + \phi \\ \hline a \\ \rho_l + \Psi^{-1} \psi \end{array}\right] = 0. \tag{4.51}$$

This refined approach follows our general strategy for attacking nonlinear problems. It starts out with introducing lots of slack variables in order to get a highly decoupled (nonlinear) system of necessary conditions. This leads immediately to the very large but extremely sparse KKT system (4.39). The size of this *linear* system is then reduced by very simple and stable eliminations; only diagonal matrices are used as block pivots. As much symmetry as possible is maintained during the elimination process; only the decoupling of lower and upper bounds and ranges causes a slight asymmetry. Finally we arrive at the smallest possible KKT system that can be derived from (4.39) without destroying any problem-specific structure. It has the original size of [QP] only, as determined by matrices A, B, and H. Contributions from bounds and ranges are concentrated in the nonnegative matrices Φ and Ψ^{-1}. They change only the diagonal of the KKT system (with the effect of spreading its eigenvalues). At this point a problem-specific indefinite solver can be applied, and internal structures of A, B, H come into play.

A comparison to [89, 88] shows a minor and a major difference. First, our formulation has an even higher degree of symmetry since arbitrary lower bounds b_l are permitted and the details in computing Φ, Ψ and ϕ, ψ are slightly different. The major difference is that equalities and inequalities, as represented by matrices A and B, are kept separate in our approach. There is a good reason for doing this. Equality constraints, such as boundary conditions or discretized differential equations in the optimal control context, are given a higher priority than inequality constraints. Consequently, when solving (4.51), the equalities are eliminated first if possible. Only the relevant (projected) part of the Schur complement $B^T \Psi B$ need to be added then to the Hessian.

Considering the m-stage block-sparse QP structure in the optimal control context again, we could still transform system (4.51) to a system like (4.25) by symmetric elimination of δv_l, which adds the Schur complement of $-\Psi^{-1}$ to the Hessian:

$$\begin{bmatrix} H + \Phi + B^T \Psi B & A^T \\ A & \end{bmatrix} \begin{bmatrix} \delta y \\ -\delta z \end{bmatrix} + \begin{bmatrix} h + (\Phi \beta_l + \phi) + B^T (\Psi \rho_l + \psi) \\ a \end{bmatrix} = 0. \tag{4.52}$$

On the other hand, the recursive algorithm presented in chapter 3 can easily be generalized to treat system (4.51) directly. Projection steps 1–4 in each stage are also applied to the current local block of B, and the resulting local Schur complement of Ψ^{-1} is formed before elimination of the local controls in step 6. Such an extension of the algorithm will be presented in a subsequent paper.

Chapter 5

Numerical results

In this chapter the results of numerical run-time and accuracy tests are reported. First we describe briefly the actual implementation of the recursive multi-stage KKT solver, which we call MSKKT, and give information on the testing environment and conditions. Then the tests for KKT systems in optimal control are described in detail, and KKT systems in multibody dynamics are presented as a further application.

5.1 Implementation and test environment

Several versions of the code MSKKT have been implemented in Fortran 77; they are all based on the BLAS [62, 32, 31] and LAPACK [4] subroutine libraries which are public domain and provide an unofficial standard for linear algebra computations. A compact storage scheme is used for the block-sparse KKT matrix. Its structure is given by an array of dimensions, and only the individual non-zero blocks are stored. As usual, the matrix data are overwritten during the factorization process, and right hand side data are overwritten during transformation and solution phases. The three phases are implemented as separate subroutines. Orthogonal projections are applied in PH steps 1,2 and 4, i.e., local constraint matrices are decomposed into orthogonal and triangular factors. The four different versions of MSKKT allow a choice between blocked or unblocked LAPACK subroutines (the former being more efficient for very large local dimensions), and between solving for one or multiple right hand side vectors.

Numerical computations have been performed in 64-bit IEEE floating point arithmetic on a Silicon Graphics workstation Iris Indigo under the IRIX operating system Release 4.0.5F. The workstation is equipped with a 50 MHz R4000/R4010 processor, 2×8 KB program and data cache, 1 MB combined secondary cache, and 80 MB main memory. All the test programs have been implemented in Fortran 77 and compiled by the MIPS Fortran compiler Release 3.10.1 with optimization flag `-O2`. LAPACK version 1.0 (from `netlib`) has been compiled by Release 3.9.1 of the same compiler using flags `-O2 -static -mp`. The BLAS library is provided by Silicon Graphics in executable form, with level 2 and 3 subroutines optimized in order to take advantage of the hardware cache structure.

5.2 KKT systems in optimal control

5.2.1 General information

5.2.1.1 Test matrices and right hand sides

A large number of artificial test problems has been used for testing the recursive KKT solver. In all the test runs pseudo-random blocks of the KKT matrix were created using the LAPACK test matrix generator DLATMS. The singular values (eigenvalues in the case of symmetric blocks) and certain other properties of the pseudo-random matrices generated by DLATMS can be influenced through control parameters. These parameters were set to the following values.

`DIST = 'U'`	(uniform $(0,1)$-distribution ...)
`MODE = 5`	(... of logarithms of singular values)
`COND = 50.0`	(condition number)
`DMAX = 1.0`	(maximum singular value)

The Hessian blocks

$$\begin{pmatrix} H_j & {}^tJ_j \\ J_j & K_j \end{pmatrix} \tag{5.1}$$

were always generated as a single matrix, not in the partitioned form. When the number of rows of matrix $(F_j\ D_j)$ or $(G_j\ E_j)$ was greater than the number of columns of each of their respective individual blocks, these were also generated as a single matrix. Otherwise the component blocks were created individually, as were the local constraint matrices F_j^x, D_j^u.

The right hand side of each KKT system was set to the negative sum of columns of the KKT matrix. Hence, the exact solution always was the vector $e = (1, \ldots, 1)$.

5.2.1.2 Problem classes

Nine different classes of test problems (a1–c3) have been obtained as combinations of three types of boundary conditions and three types of discrete dynamic systems. These classes resemble the algebraic structures that arise in typical classes of ODE boundary value problems; results for each of them will be discussed below in separate sections.

The chosen types of boundary conditions can be described as follows (in the notation of section 1.1.2):

(a) Decoupled two-point boundary conditions.
Type (a) KKT systems model completely separated initial and final constraints,

$$r(x_1, \ldots, x_k) = r^0(x_1, x_k) = \begin{pmatrix} r_1^0(x_1) \\ r_k^0(x_k) \end{pmatrix}. \tag{5.2}$$

The numbers of initial and final conditions are identical to the number of state variables, $r_1^0(x_1), r_k^0(x_k) \in I\!R^{n_x}$, so that x_1 and x_k are completely fixed.

(b) Coupled two-point boundary conditions.
Type (b) KKT systems resemble linearly coupled initial and final constraints,

$$r(x_1, \ldots, x_k) = r^1(x_1, x_k) = r_1^1(x_1) + r_k^1(x_k). \tag{5.3}$$

The number of boundary conditions is equal to the number of state variables, $r^1(x_1, x_k) \in \mathbb{R}^{n_x}$. A typical example for problem class (b) are periodic problems.

(c) Multipoint boundary conditions.
Type (c) KKT systems model general constraints which couple the state variables across all stages,

$$r(x_1, \ldots, x_k) = r^1(x_1, \ldots, x_k) = r_1^1(x_1) + \cdots + r_k^1(x_k). \tag{5.4}$$

The number of boundary conditions is again equal to the number of state variables, $r(x_1, \ldots, x_k) \in \mathbb{R}^{n_x}$.

The chosen types of discrete dynamic systems differ in the relative number of local state and control variables:

(1) First order systems.
These KKT systems have as many state variables as control variables, resembling the typical structure of BVP associated with first order ODE.

(2) Second order systems.
These KKT systems have twice as many state variables as control variables, modeling the typical structure of BVP associated with second order ODE (after reduction to first order).

(3) Robot optimization systems.
These KKT systems have 13 state variables (including one system parameter) and 6 control variables. Such systems arise in minimum time path planning of a 6-joint robot. The system parameter is the free terminal time, cf. the example in chapter 1. Type (3) is a specific example of order 2 systems. It has been included in the test set in order to get an impression of absolute computation times.

5.2.1.3 Problem sizes

For each of the nine problem classes a considerable number of different problem sizes has been run. The number of stages (or grid size) is started at $m = 4$, and then repeatedly doubled until $m = 8192$ is reached. The local number of state variables is independently varied between $n_x = 2$ and $n_x = 512$, also in powers of 2. However, no problem requiring more than 68 MB of workspace was attempted, so that time measurements are not biased due to swapping. The number of state variables is of course fixed for robot optimization systems.

5.2.1.4 Compared sparse solvers

Each of the test problems has been attempted by several solvers.

(1) MSKKT: the unblocked/single right hand side version.

(2) MSKKT: the blocked/single right hand side version.
In most cases there have been found no significant differences between solvers (1) and (2). Therefore results will only be described for solver (1) in the following.

(3) MA28: a general purpose sparse solver [33].
This solver has been chosen for comparison because it is an extensively tested and widely used code. MA28 is part of the commercial Harwell Subroutine Library where it was superseded by the improved code MA48; now it is also available in the public domain software library `netlib`.

Before a matrix is actually factorized, its sparse structure is examined and stored in certain index arrays. Therefore the user interface provides two factorization subroutines. MA28AD combines analyze and factorization phases, whereas MA28BD factors a matrix with the same sparse structure as a previously analyzed matrix. Subroutine MA28CD, finally, applies a given factorization to solve for a right hand side. Below we will use the name MA28A if the analyze phase is included, MA28B if it is not, and MA28 if this is irrelevant (as in accuracy comparisons). In all our tests the MA28A input parameter that controls bias towards numeric or sparsity pivoting has been set to its recommended value `u=0.1`.

(4) BAND: the LAPACK band solver.
This state of the art band solver has been applied to all decoupled TPBVP's (type (a) problems); it is of course not useful on more general boundary conditions. The band solver provides a subroutine DGBTRF for the factorization phase and a subroutine DGBTRS for the combined transformation and solution phases.

5.2.1.5 Runtime measurements

CPU time measurements have been performed individually for each phase using the built-in 100 Hz hardware clock, with a resolution of 0.01 seconds. For the transformation and solution phases, which required less than 0.01 seconds on most of the smaller test problems, the resolution was enhanced as follows. After solving once (in order to get correct solution accuracies), each of these phases was performed repeatedly until more than one second of CPU time was spent, and the total time was then simply divided by the number of runs. This technique could not be applied to the factorization phase, since repeated factorization would often cause error exits due to near-singular matrices.

In order to prevent external influences, all the test runs were performed over night, the machine was disconnected from the network, and no one was logged in during time measurements. In addition, the processing of any crontab files by the clock daemon was stopped.

5.2.1.6 Accuracy measurements

For accuracy measurements we use averaged p-norms ($p = 1, 2, \infty$) on $I\!R^N$:

$$\|x\|_{N,p} = \left(\frac{1}{N}\sum |x_i|^p\right)^{1/p}, \qquad 1 \le p < \infty, \tag{5.5}$$

$$\|x\|_{N,\infty} = \max\{|x_i|\} \equiv \|x\|_\infty. \tag{5.6}$$

In these norms we have $\|e\|_{N,p} = 1$ for all N, p. Hence, if y is the numerical solution of the KKT system and $x = y - e$ is the absolute error, these norms provide measurements of the *relative* error with respect to the corresponding non-averaged p-norm:

$$\|x\|_{N,p} = \|y - e\|_p / \|e\|_p = \|x\|_p / N^{1/p}, \tag{5.7}$$

$$\|x\|_{N,\infty} = \|y - e\|_\infty / \|e\|_\infty = \|x\|_\infty. \tag{5.8}$$

Moreover, one can easily show that $N^{\frac{1}{q}-\frac{1}{p}}\|x\|_{N,q} \le \|x\|_{N,p} \le \|x\|_{N,q} \le N^{\frac{1}{p}-\frac{1}{q}}\|x\|_{N,p}$ for all $x \in I\!R^N$ and $1 \le p \le q \le \infty$.

5.2.1.7 Illustrations

In the following sections we supply three types of plots to illustrate results of the run time and accuracy tests. Data points in these plots are marked by certain symbols and joined by straight lines; the marker symbols and line styles associated with each solver remain identical throughout this chapter, see figure 5.1.

Figure 5.1: Line styles and marker symbols for the plots

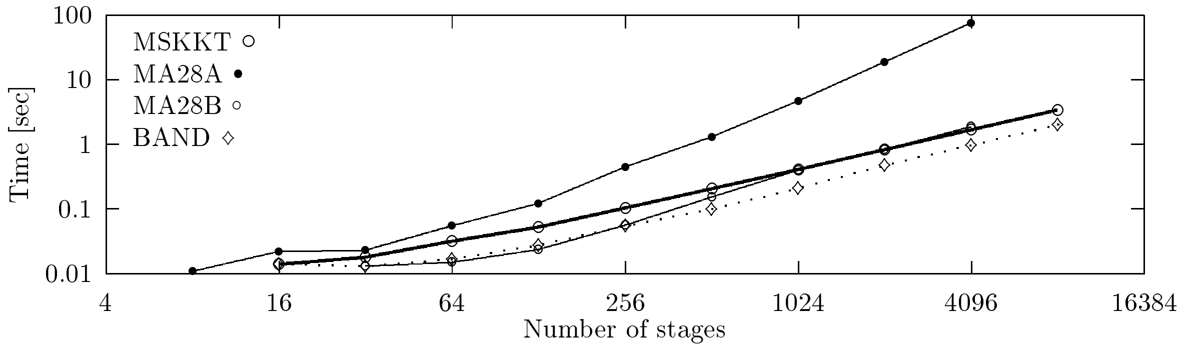
—○—	MSKKT	—•—	MA28A, MA28
··◇··	BAND	—○—	MA28B

The first type of plots shows the CPU time for each solver versus the number of stages, m, while the local number of states is fixed at some value n_x. The CPU times given are the total times for a complete KKT system solution. They include the factorization, transformation and solution phases for solvers MSKKT, BAND, and MA28B, and in addition the analyze phase in the case of MA28A. Comments on CPU times for the individual phases will be given in the text when appropriate.

Plots of the second type show speed-up factors of MSKKT versus each other solver, i.e., relative CPU times. They are defined as the quotients CPU(solver)/CPU(MSKKT). Hence MSKKT is faster than a competing solver if the speed-up factor is greater than 1, and slower else. Separate plots are provided for MA28A and the remaining solvers because their respective run times often differ by several orders of magnitude.

The third type of plots shows relative solution accuracies obtained by each solver. Accuracies are given in correct decimal digits as measured in the infinity-norm, i.e., the numbers are obtained according to the formula $\texttt{acc} = -\log_{10}\|\texttt{err}\|_\infty$. Results for the averaged 1- and 2-norms will be commented in the text when appropriate.

Figure 5.2: CPU times for decoupled 1st order TPBVP, $n_x = 2$



5.2.2 Decoupled two-point boundary value problems

In this section the test problems have the simplest type of constraints, namely decoupled two-point boundary conditions

$$\begin{pmatrix} F_1^x \Delta x_1 + e_1^x \\ F_m^x \Delta x_m + e_m^x \end{pmatrix} = 0. \tag{5.9}$$

F_1^x and F_m^x are full-rank dense random blocks here, and all the other constraint matrix blocks $F_j^x, D_j^u,\ F_j^c, D_j^c,\ F_j, D_j$ are not present in the KKT system. As pointed out in section (3.3), this leads to a natural block-banded structure of the KKT matrix, so that all the linear solvers can handle these problems.

5.2.2.1 First order systems

Here we consider test problems with as many control variables as state variables, $n_x = n_u$. The values chosen for illustrations are $n_x = 2, 8, 32$. Before the results are discussed in detail, we give a brief overview.

In figure 5.2 the CPU times of each solver are plotted versus the number of stages, while the number of state variables is fixed at $n_x = 2$. Figure 5.3 shows the speed-up factors of MSKKT versus BAND and MA28B for the same value $n_x = 2$; the speed-up versus MA28A is given in figure 5.4. Solution accuracies are finally displayed in figure 5.5.

Similar data for the cases $n_x = 8, 32$ are provided in figures 5.6–5.9 and 5.10–5.13, respectively. Note that the scaling of both axes is logarithmic on the CPU time plots, whereas the vertical axis has linear scaling in all other cases.

Let us first turn to figures 5.2–5.5 and consider the results for the smallest block size, $n_x = 2$. On this problem type BAND is about twice as fast as MSKKT regardless of the number of stages. Hence there is a slow-down rather than a speed-up when using MSKKT. On the other hand, MSKKT is always faster than MA28A; it becomes up to 45 times as fast when m is increased to 4096. A reason for the exceptional behavior of MA28A is found when the exact numerical data are examined. In contrast to the factorization, transformation and solution phases, where run times grow roughly linear with m for

Figure 5.3: Speed-up for decoupled 1st order TPBVP, $n_x = 2$

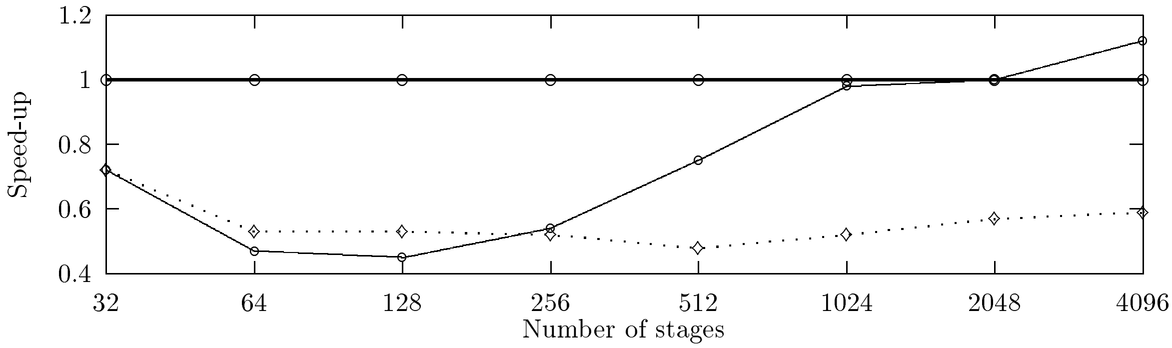
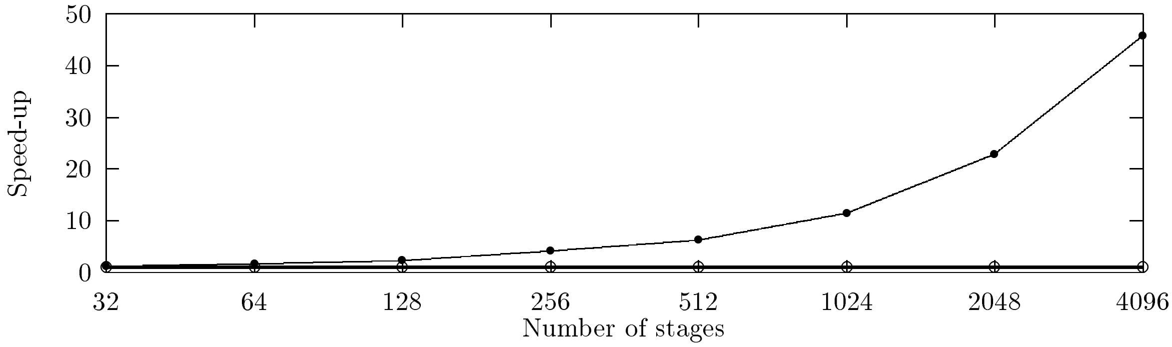


Figure 5.4: Speed-up for decoupled 1st order TPBVP, $n_x = 2$



all solvers, the analyze phase of MA28A clearly exhibits superlinear complexity $\mathcal{O}(m^\alpha)$. Typically the analyze time multiplies by about 4 when m is doubled, hence $\alpha \approx 2$. But in exactly two cases (see below) the time factor was observed to vary between 4 and 8, indicating a value of $\alpha > 2$. On the present problem the analyze phase requires up to 97.5 % of the total run time of MA28A. This can be seen in comparison to MA28B, the performance of which lies between BAND and MSKKT on average.

The expensive analyze phase is a serious drawback of MA28, even if the matrix structure needs to be analyzed only once (at the beginning of an SQP iteration) and MA28B is used in subsequent factorizations. The band solver, on the other hand, has a clear advantage on problems with a very small bandwidth, since index computations in MSKKT and MA28 lead to a relatively large overhead. In the case of MSKKT this is especially true for the transformation and solution phases. These take roughly 40 % of the total CPU time on the problem under consideration. The corresponding amounts for the band solver and MA28B are about 30 % and 10 %, respectively.

Note that only MSKKT and BAND can solve the problem with $m = 8192$ stages; MA28 needs too much workspace. Observe also that time measurements are not very accurate on small grids, where run times are in the neighborhood of 0.01 sec (the clock resolution). Some CPU times could not be measured at all on less than 32 stages here, see figure 5.2, and hence no speed-up factors are given in this range.

Figure 5.5: Accuracies for decoupled 1st order TPBVP, $n_x = 2$

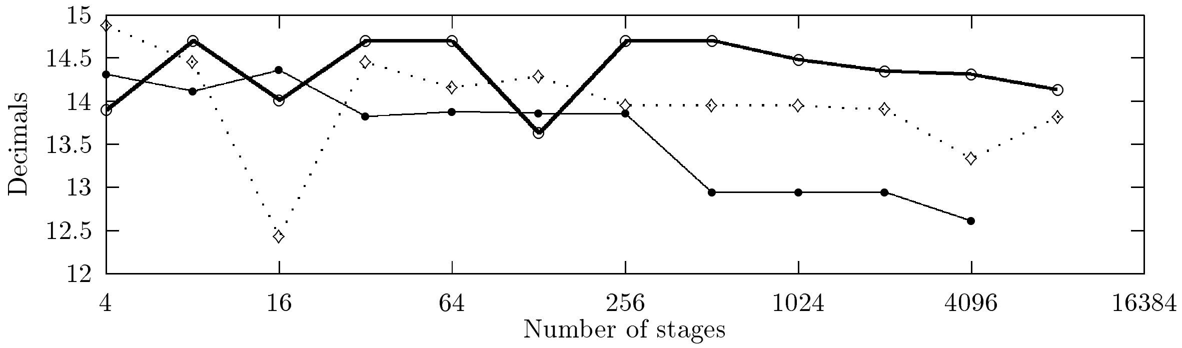
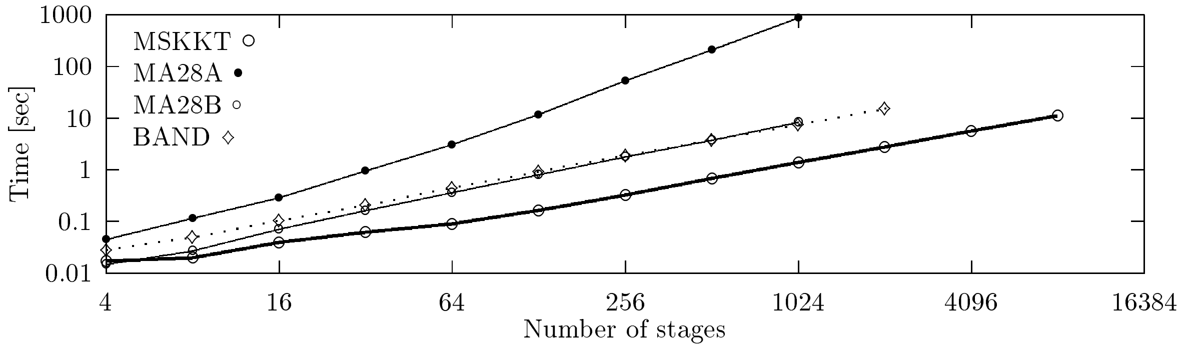


Figure 5.6: CPU times for decoupled 1st order TPBVP, $n_x = 8$



The solution accuracies achieved by each solver are all above 12 significant digits; they are roughly comparable for up to 256 stages, see figure 5.5. However, MSKKT shows slightly better results than BAND which is in turn more accurate than MA28, especially when the number of stages is increased.

Next we consider test problems with $n_x = 8$ state variables, see figures 5.6–5.9. Here MSKKT clearly is the fastest solver. It runs up to 600 times as fast as MA28A at $m = 1024$. MA28B and BAND perform roughly equal; they are about 2 times slower than MSKKT on small grids and 5 to 6 times slower on fine grids, see figure 5.7. The relative CPU times of the transformation and solution phases for solvers MSKKT, BAND, and MA28B drop to 30 %, 10 %, and 5 %, respectively.

Note that MSKKT shows a significant advantage in storage efficiency: the problem can still be solved for 8192 stages, whereas the limits for BAND and MA28 are 2048 and 1024 stages, respectively.

On this block size, $n_x = 8$, the band solver obtains more accurate results than both MSKKT and MA28, which are comparable on average, see figure 5.9. But obviously the accuracy of MSKKT is essentially independent of the grid size. This strange behavior can be explained as follows. For MSKKT, the order of operations on the block level is completely fixed according to the m-stage block-sparse structure. This fixed order leads to a characteristic accuracy pattern in the solution components. (More details will be

Figure 5.7: Speed-up for decoupled 1st order TPBVP, $n_x = 8$

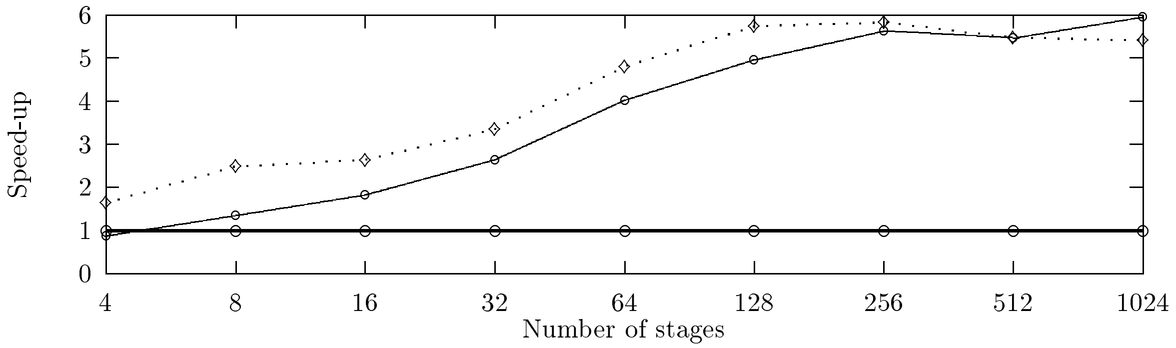


Figure 5.8: Speed-up for decoupled 1st order TPBVP, $n_x = 8$

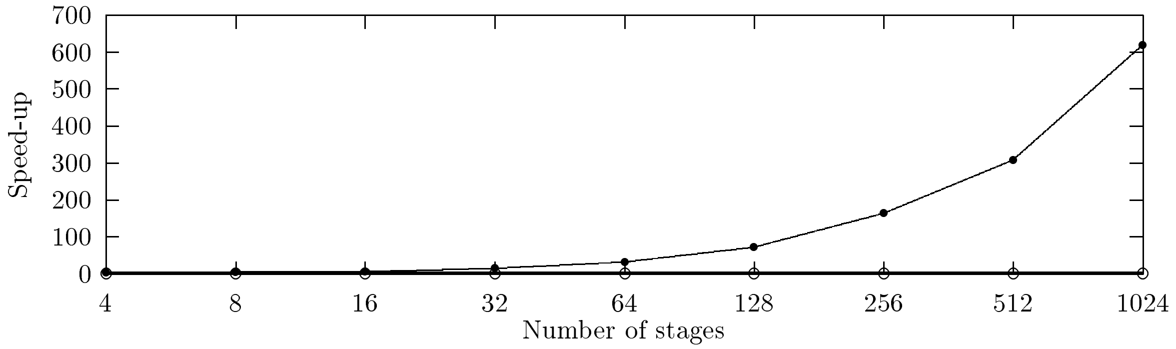
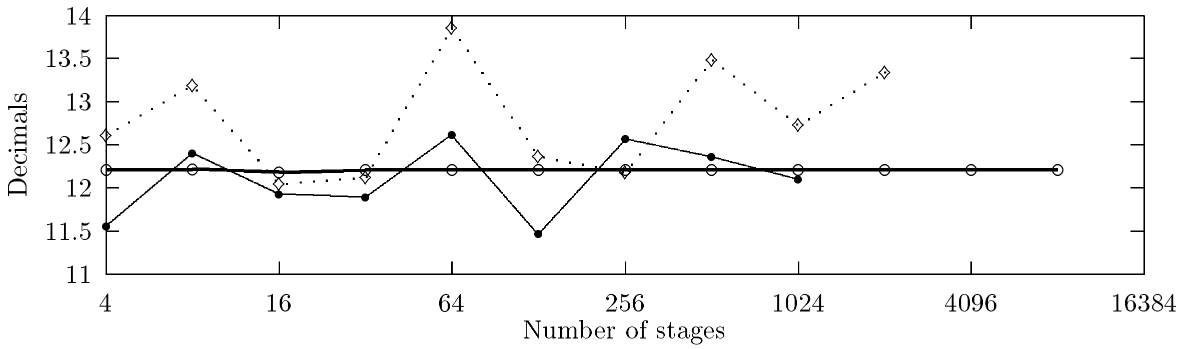


Figure 5.9: Accuracies for decoupled 1st order TPBVP, $n_x = 8$



given below.) The chosen error criterion measures only the least accurate component which in this case happens to appear in the multiplier of terminal constraints, μ^x_m, for all m. Now, the $2m$-stage test problem for block size n_x is generated by *pre*pending m new stages to the m-stage problem for the same block size (after removing the initial constraint block F^x_1). Hence all these test problems have identical matrices in the tail stages. This is precisely what makes the error in μ^x_m independent of the grid size: it is evidently dominated by effects due to the factorization of F^x_m rather than by error propagation through the head stages. A different behavior has been observed in the

Figure 5.10: CPU times for decoupled 1st order TPBVP, $n_x = 32$

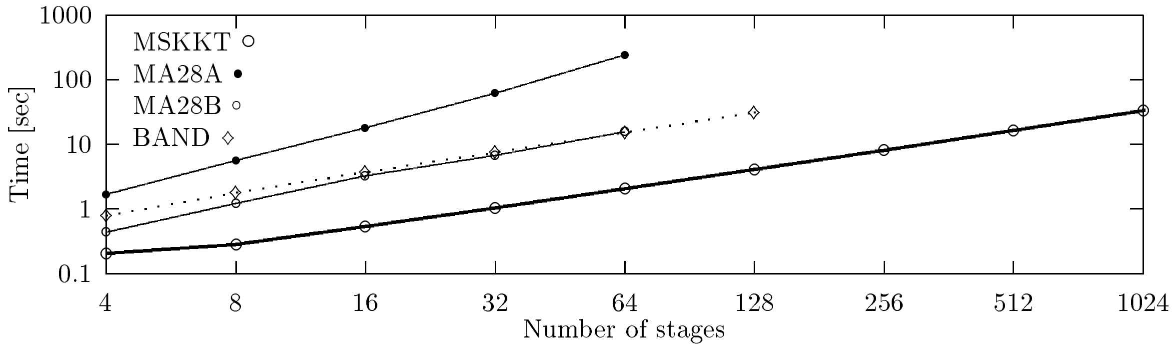


Figure 5.11: Speed-up for decoupled 1st order TPBVP, $n_x = 32$

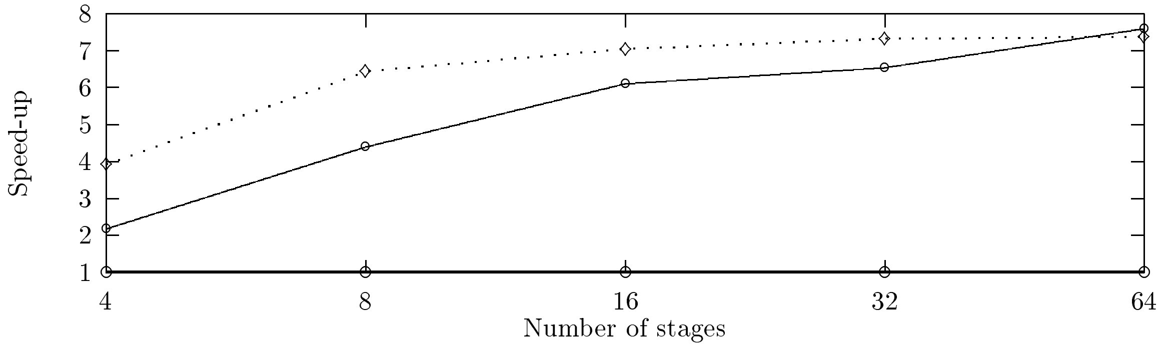
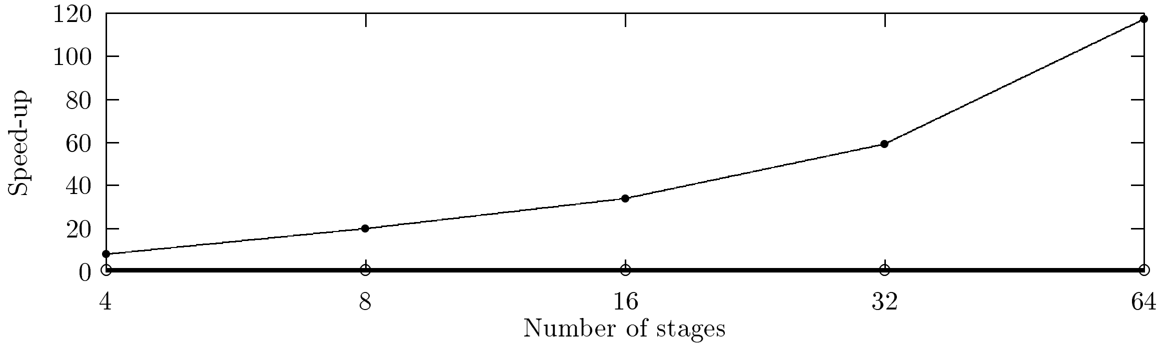


Figure 5.12: Speed-up for decoupled 1st order TPBVP, $n_x = 32$



previous example, $n_x = 2$, since the largest error in μ_m^x often was exceeded by the error in other solution components.

Finally we consider the case $n_x = 32$ which shows very similar pictures, see figures 5.10–5.13. MA28B is now a little bit faster than the band solver, and the speed-up of MSKKT versus both competitors increases slightly. So does the advantage in storage efficiency. The largest problems that can be handled by MA28 and BAND have 64 and 128 stages, respectively. MSKKT is successful on problems that are up to 16 times and 8 times as big; its limit is only reached at 1024 stages.

Figure 5.13: Accuracies for decoupled 1st order TPBVP, $n_x = 32$

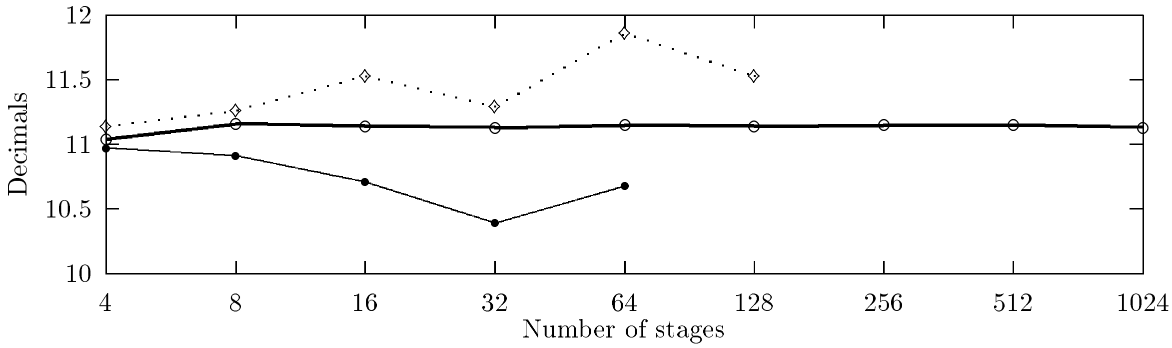
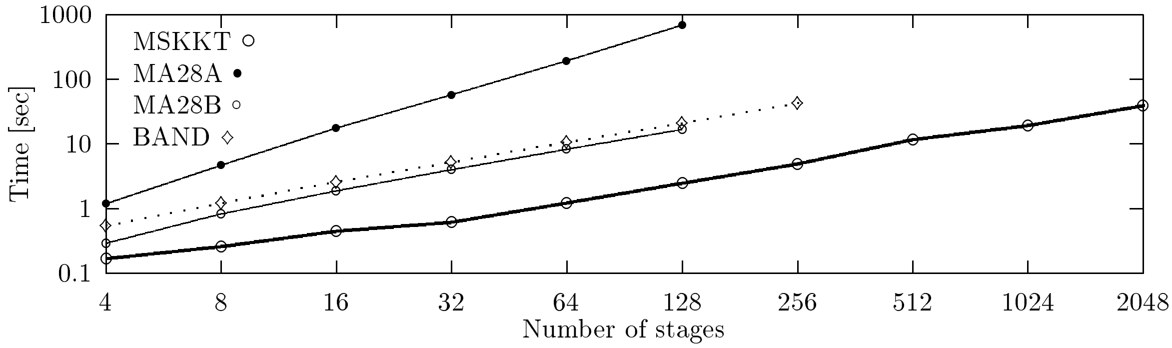


Figure 5.14: CPU times for decoupled 2nd order TPBVP, $n_x = 32$



Solution accuracies are again roughly comparable here (they differ by ±1 digit only, see figure 5.13), but the band solver is always better than MSKKT, and MSKKT is always more accurate than MA28.

Results for other values of n_x are qualitatively similar to the ones described here. One also observes that the relative amount of CPU time spent in the transformation and solution phases decreases with increasing block size n_x when the number of stages is kept constant. Asymptotically this time grows as $\mathcal{O}(n_x^2)$ only, whereas the factorization time grows as $\mathcal{O}(n_x^3)$. In the last case shown these percentages are about 10 %, 5 % and 2 % for MSKKT, BAND and MA28, respectively. As should be expected, the solution accuracy of each solver also decreases with increasing n_x, simply because more variables are introduced into the system.

5.2.2.2 Second order systems

Here we consider test problems with twice as many state variables as control variables, $n_x = 2n_u$. From now on we pick only *one* specific number n_x for illustrations. Anything special that happens for different values will be described in the text.

Results in this section are given for the value $n_x = 32$, see figures 5.14–5.17. The runtime behavior of all solvers is rather similar to the case of first order systems, and the same holds true for differences in storage efficiency, see figures 5.10 and 5.14.

Figure 5.15: Speed-up for decoupled 2nd order TPBVP, $n_x = 32$

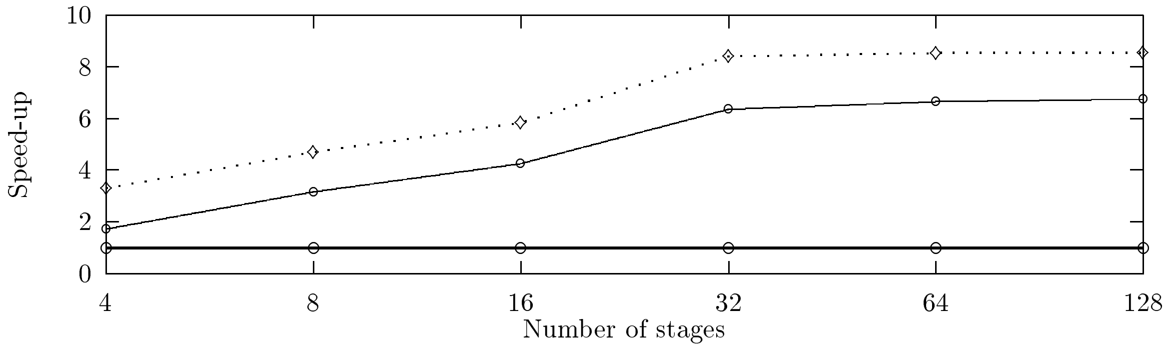


Figure 5.16: Speed-up for decoupled 2nd order TPBVP, $n_x = 32$

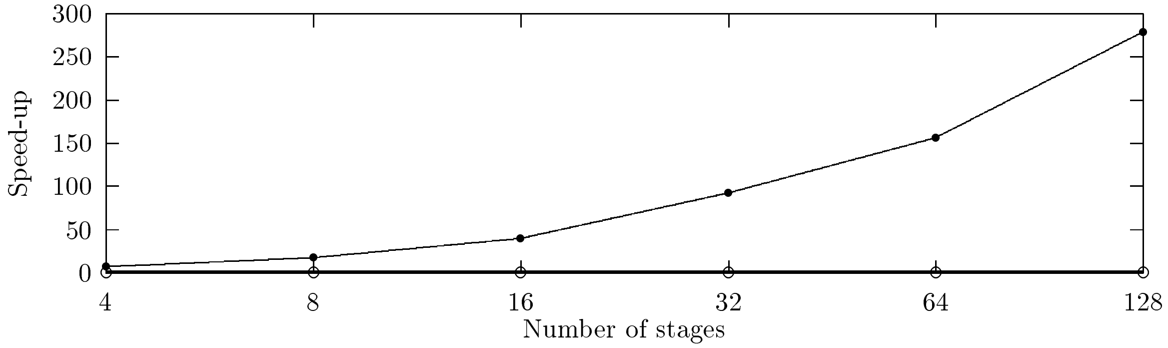
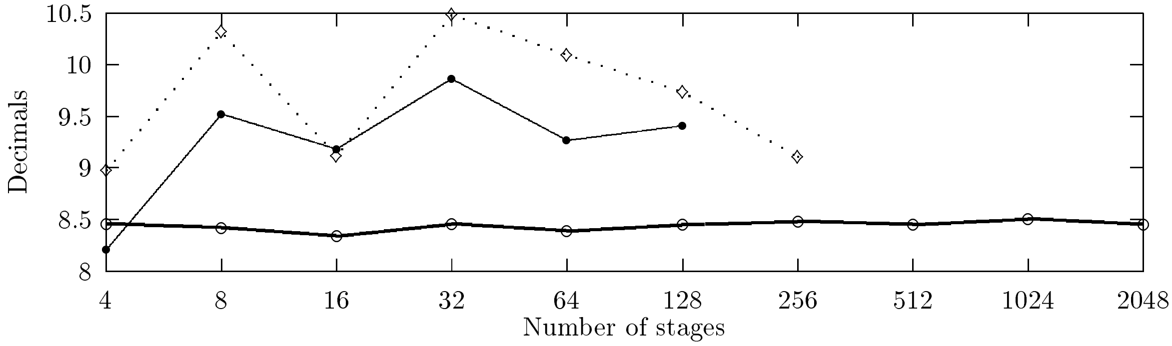


Figure 5.17: Accuracies for decoupled 2nd order TPBVP, $n_x = 32$



But a look at figure 5.17 shows that MSKKT achieves only an exceptionally low accuracy of about 8.5 digits in this example. BAND and MA28 are similarly inaccurate at $m = 4$, whereas one or two more digits are achieved when the number of stages is greater. This is again a consequence of the differing pivoting strategies. Since the final constraint matrix F^x_m is always chosen as the first block pivot by MSKKT, the accuracy of μ^x_m can never be increased above a certain level. MA28 and BAND, on the other hand, may avoid this specific large error when the system is large enough to allow for a better pivoting.

Figure 5.18: CPU times for decoupled robot TPBVP

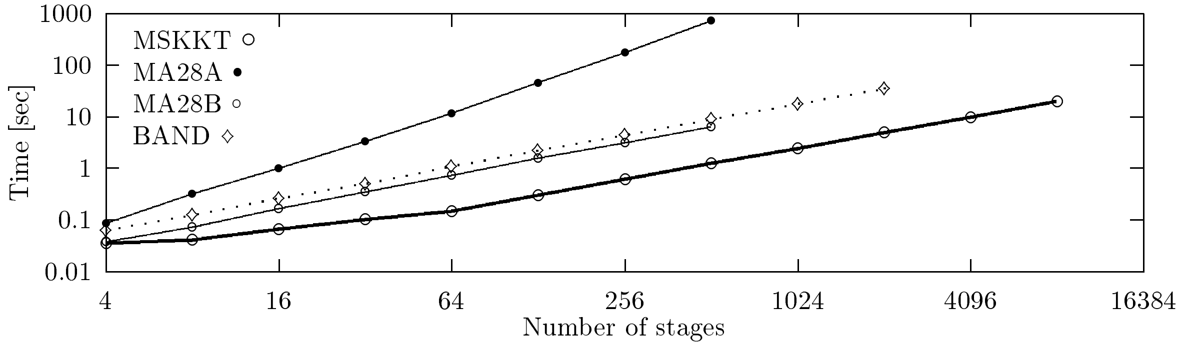
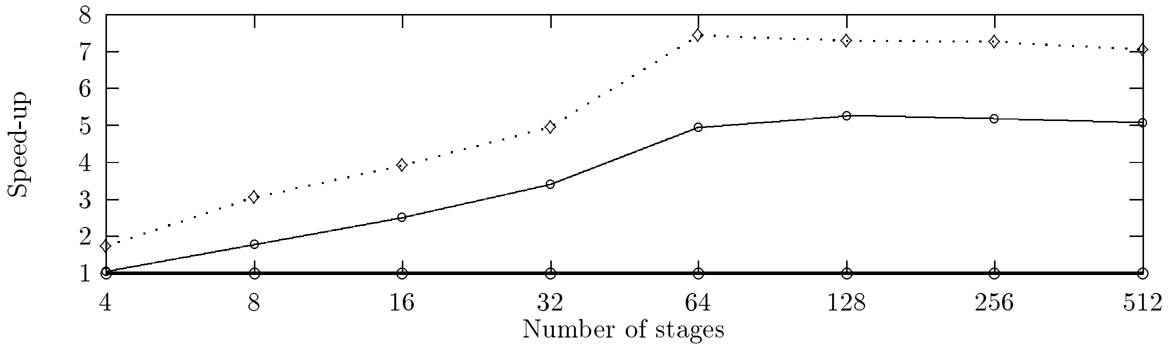


Figure 5.19: Speed-up for decoupled robot TPBVP



However, the distribution of errors among the individual solution components shows that the situation is actually better than indicated in figure 5.17. Only components in μ_m^x are less accurate than 9 digits, and an accuracy below 10 digits is only observed in the multipliers λ_{m-1} and μ_m^x. When 12 digits are considered, "low accuracy" components can be found in μ_m^x and stages $m-6$ through $m-1$ only, and for a level of 14 digits the "low accuracy" region includes multipliers μ_1^x, μ_m^x and stages $m-11$ through $m-1$. State and control variables are never less accurate than 11 and 10 digits, respectively.

These observations are strictly valid for the whole set of test problems represented in figure 5.17. But they also indicate a general tendency. Solution accuracies are usually high for the state components and low for the multipliers; the control variables lie in between. The highest accuracies are achieved for locally fixed states and for variables in the following stages, the lowest ones are found in the corresponding multipliers and in preceding stages. That is, the built-in high-level pivoting strategy of MSKKT sacrifices the accuracy of dual variables for a higher accuracy of primal variables, which reflects their respective importance in the SQP iteration perfectly.

5.2.2.3 Robot optimization systems

Results for this problem type are similar to the previous examples, see figures 5.18–5.21. As in most cases of the previous sections, MSKKT clearly outperforms its competitors.

Figure 5.20: Speed-up for decoupled robot TPBVP

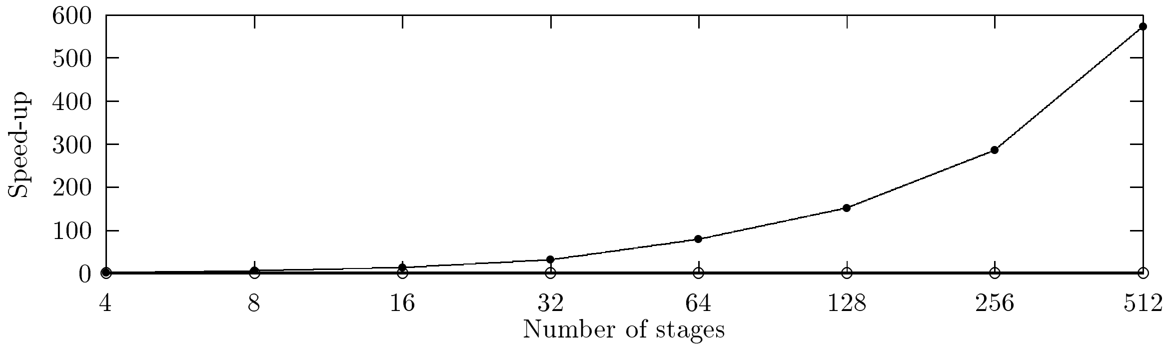
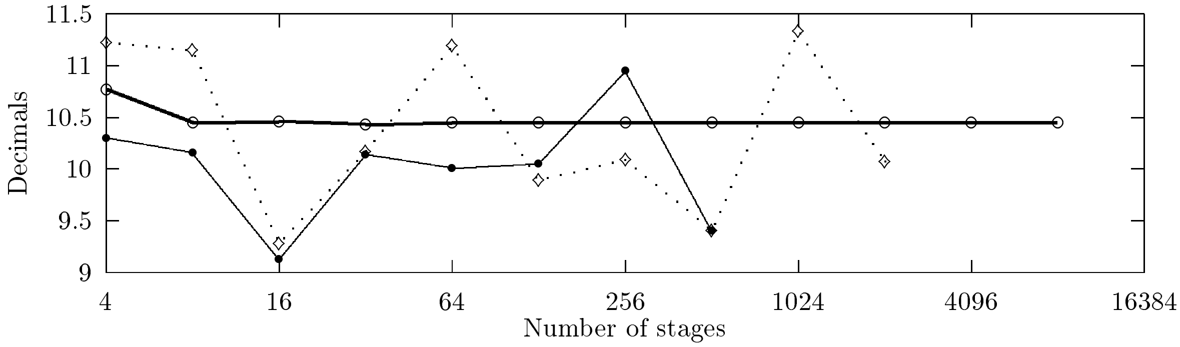


Figure 5.21: Accuracies for decoupled robot TPBVP



Its runtime advantage varies between factors of 3 and 600 for MA28A, between 1 and 5 for MA28B, and between 2 and 7 for the band solver. Even if accuracies are compared, MSKKT is better than MA28 on average and about as good as the band solver, see figure 5.21. Its accuracy level is again constant.

Let us now take a look at absolute run-times. We consider a coarse grid (16 stages), a fine grid (64 stages), and a very fine grid (256 stages). MA28A needs 1.00 seconds, 11.8 seconds, and 3 minutes (176 seconds) to solve these problems. MA28B omits the analyze phase and is finished after 0.17, 0.73, and 3.18 seconds. The band solver is a little slower, taking 0.26, 1.10 and 4.48 seconds, respectively. MSKKT, finally, needs only 0.07, 0.15, and 0.62 seconds. Each of the problems is solved in significantly less time than 0.83 seconds, the time that MA28A spends on just analyzing the smallest of them. MSKKT completes even the 4096-stage problem within 10 seconds, which is not enough time for MA28A to analyze the 64-stage problem.

We conclude the section of decoupled problems by some remarks on solution accuracies in the 1- and 2-norms. These norms measure the average error and average quadratic error over all solution components, respectively, rather than only the largest individual error. Hence the absolute number of accurate digits appears always higher in these norms, but the majority of test runs shows that the relation between different solvers remains essentially unaffected.

Figure 5.22: CPU times for coupled 1st order TPBVP, $n_x = 16$

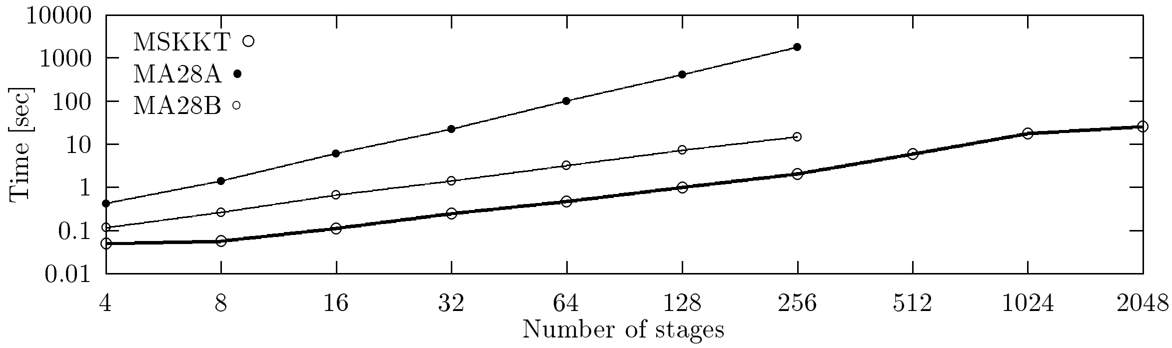
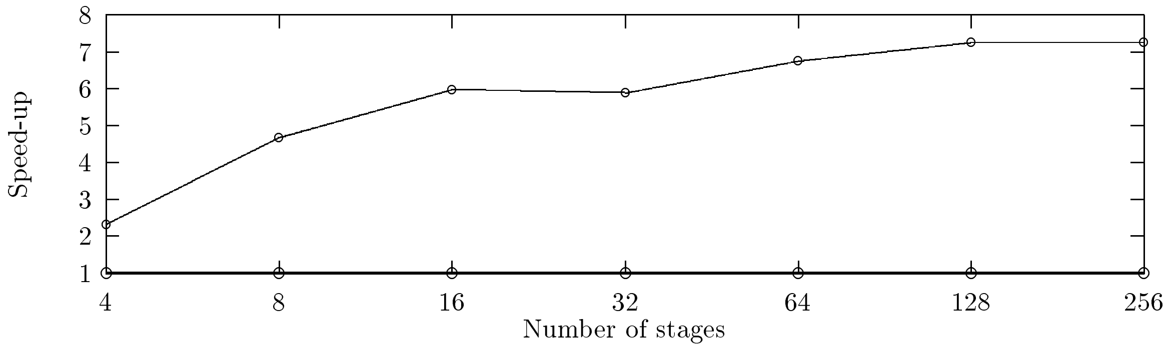


Figure 5.23: Speed-up for coupled 1st order TPBVP, $n_x = 16$



Still one interesting observation can be made. Accuracies in the 1- and 2-norms tend to *increase* with m for all solvers. This indicates that the number of "inaccurate" solution components is more or less independent of the number of stages, as has already been mentioned for MSKKT. The just described behavior occurs most clearly for the recursive solver (its accuracy often increases *monotonically* with m), and especially when applied to first order problems (type a1). On this problem type and in the case of robot optimization systems (type a3) one usually finds BAND and MSKKT more accurate than MA28, whereas MA28 and the band solver give clearly better results than MSKKT on most of the second order problems (type a2).

5.2.3 Coupled two-point boundary value problems

In this section the test problems have coupled initial and final conditions,

$$F_1 \Delta x_1 + F_m \Delta x_m + e_* = 0. \tag{5.10}$$

F_1 and F_m are full-rank dense random blocks here, the other global blocks F_j, D_j vanish, and local constraints are not present. These problems cannot be handled by the band solver due to the coupling of initial and final values. Hence, only our recursive solver and MA28 are compared.

Figure 5.24: Speed-up for coupled 1st order TPBVP, $n_x = 16$

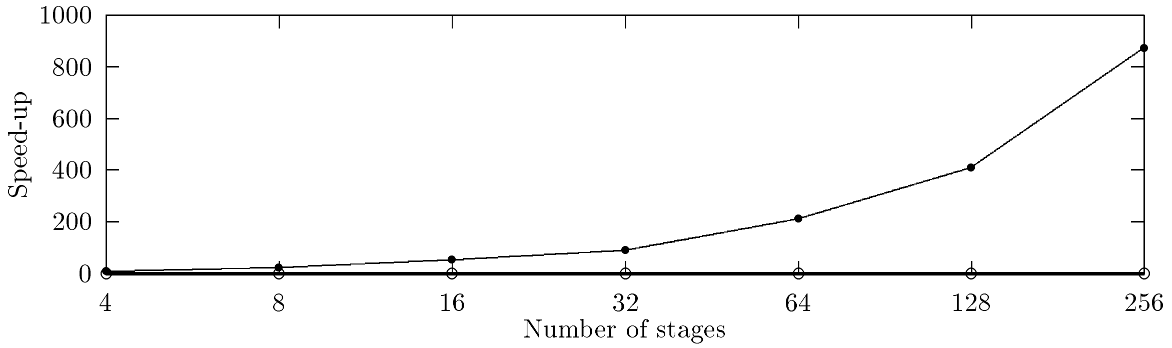


Figure 5.25: Accuracies for coupled 1st order TPBVP, $n_x = 16$

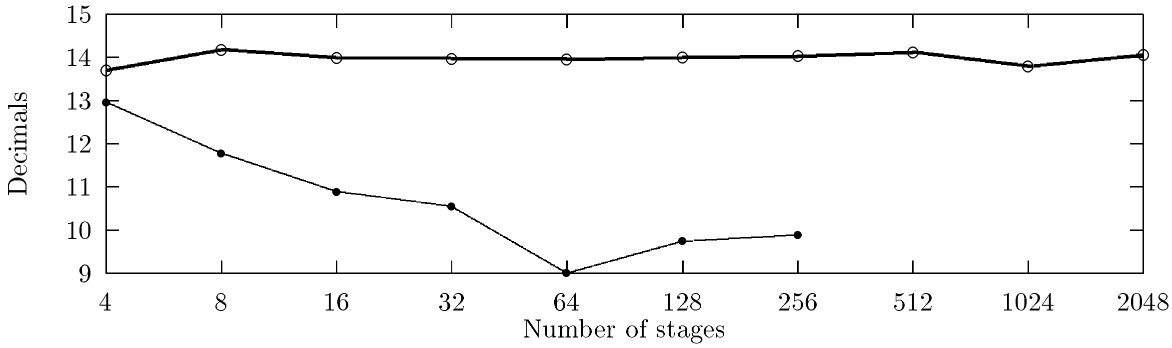
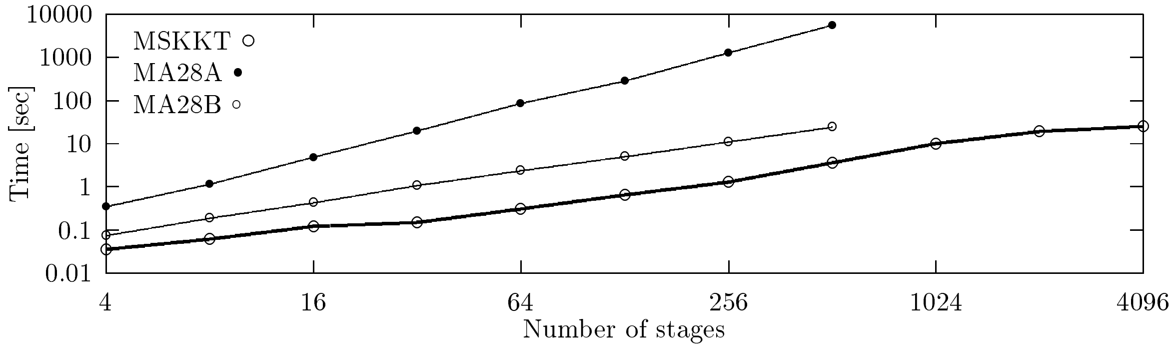


Figure 5.26: CPU times for coupled 2nd order TPBVP, $n_x = 16$



5.2.3.1 First order systems

Here we choose $n_x = n_u = 16$ as the number of state variables, see figures 5.22–5.25. The results are again rather positive for the recursive solver. MSKKT is between 2 and 7 times as fast as MA28B (a speed-up of 6 is already reached at 16 stages), and between 10 and 900 times as fast as MA28A. For instance, to solve the 256-stage system, MA28A needs nearly half an hour (1790 seconds), whereas MSKKT is finished after 2.05 seconds. Even the 2048-stage system takes less than half a minute (25.9 seconds).

Figure 5.27: Speed-up for coupled 2nd order TPBVP, $n_x = 16$

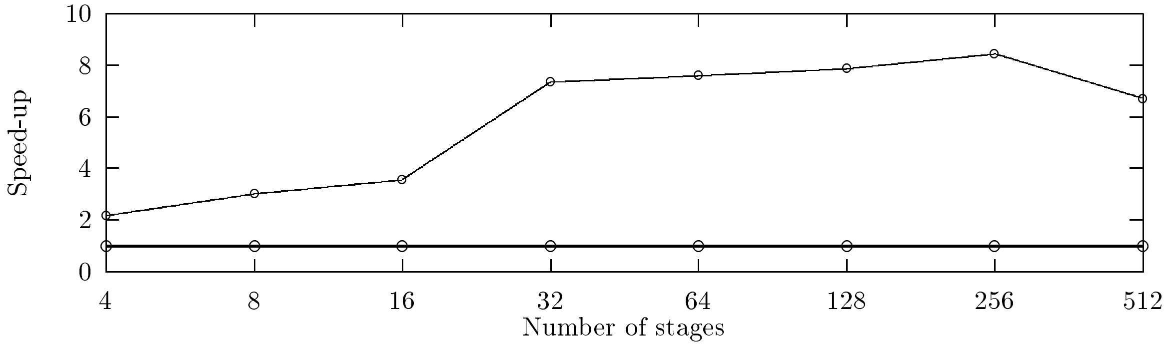
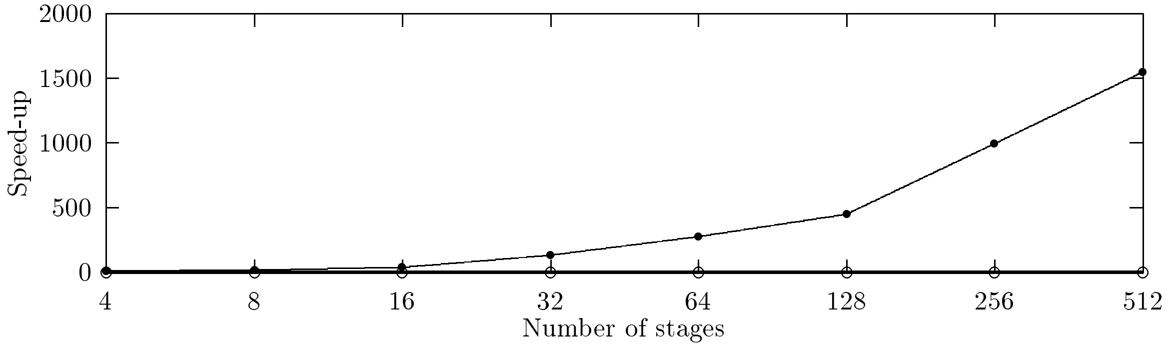


Figure 5.28: Speed-up for coupled 2nd order TPBVP, $n_x = 16$



Observe also that MSKKT obtains always between one and five more significant digits than MA28, a substantially higher accuracy! As in many of the decoupled cases, its accuracy is essentially independent of the number of stages.

An effect of the coupling on the runtime behavior of any solver could not be observed for most block sizes. However, the analyze phase of MA28A shows "super-quadratic" complexity in the single case $n_x = 32$. When the number of stages is doubled repeatedly starting at $m = 4$, the analyze time multiplies by 4.6, 3.2, 5.7, and 7.8, respectively. This is equivalent to an average factor of about 5, or a complexity exponent $\alpha \approx 2.3$.

5.2.3.2 Second order systems

Here $n_x = 2n_u = 16$ is again chosen as the number of state variables, see figures 5.26–5.29. The recursive solver is faster than MA28B by almost the same factors as in the previous problem. On small grids the speed-up is about 3, the factor 7 is reached at $m = 32$ stages and then retained or exceeded on larger grids. Although both solvers achieve similar accuracies of roughly 12 decimals for 4 and 8 stages, MA28 tends to become significantly less accurate as the number of stages increases.

The runtime behavior of the analyze phase is again worse than quadratic for $n_x = 32$. Upon increasing m the CPU time multiplies by 5.2, 5.3, 5.3, 3.9, and 7.4, respectively. This amounts to an average factor of 5.3, or a complexity exponent $\alpha \approx 2.4$.

Figure 5.29: Accuracies for coupled 2nd order TPBVP, $n_x = 16$

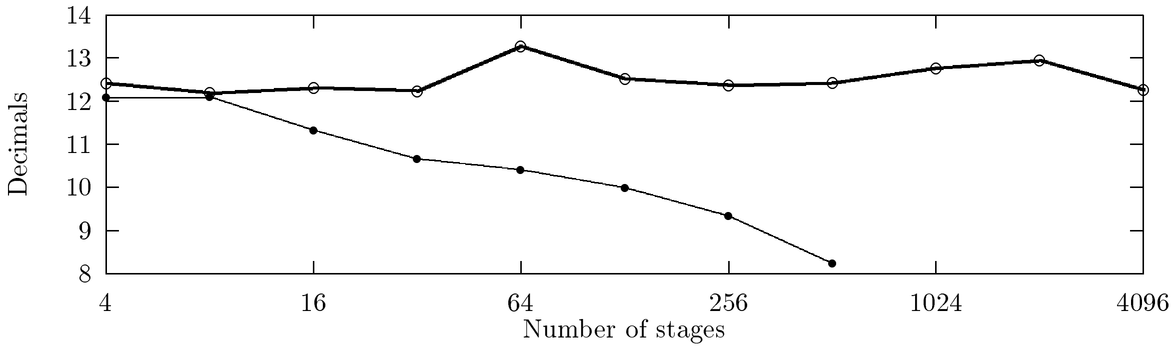


Figure 5.30: CPU times for coupled robot TPBVP

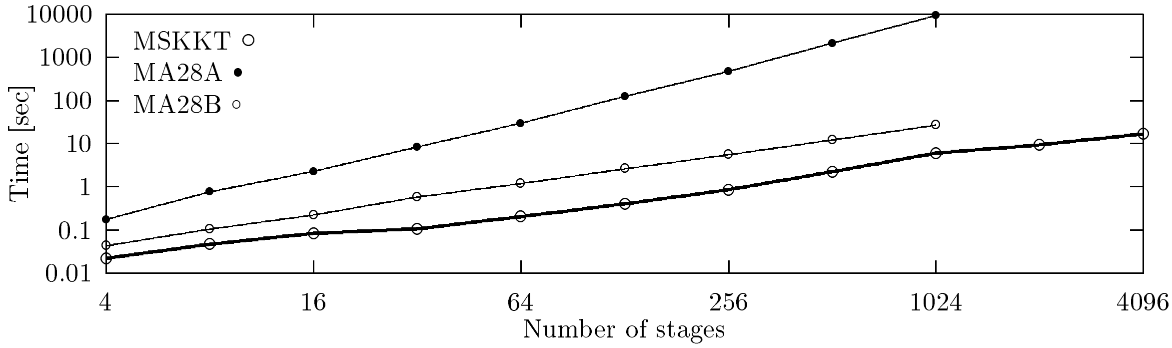
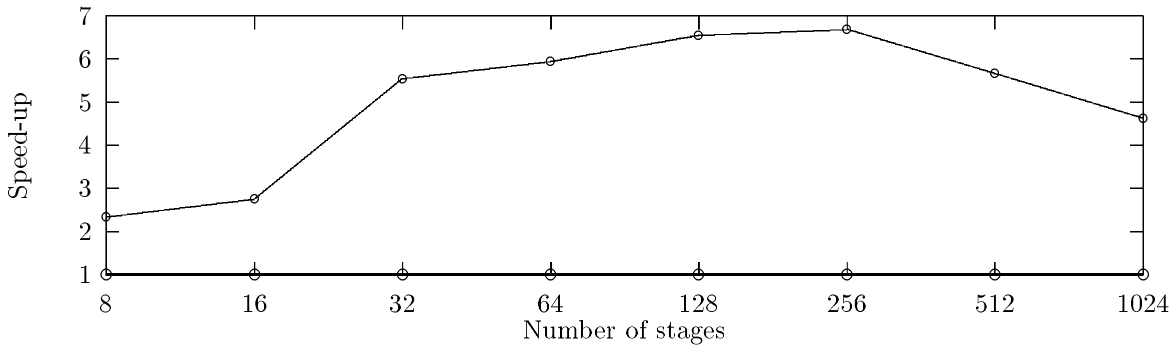


Figure 5.31: Speed-up for coupled robot TPBVP



5.2.3.3 Robot optimization systems

The results of the runs for coupled robot TPBVPs are shown in figures 5.30–5.33. MSKKT proves again more accurate and substantially faster than MA28 in all cases.

Let us again compare absolute run times on $m = 16, 64, 256$ stages. MA28A needs 2.26 seconds, half a minute (30.2 seconds), and 8 minutes (482 seconds) to solve these problems. MA28B takes only 0.23, 1.21 and 5.65 seconds, respectively, but MSKKT is finished after 0.08, 0.20, and 0.84 seconds. Again each problem is solved in significantly

Figure 5.32: Speed-up for coupled robot TPBVP

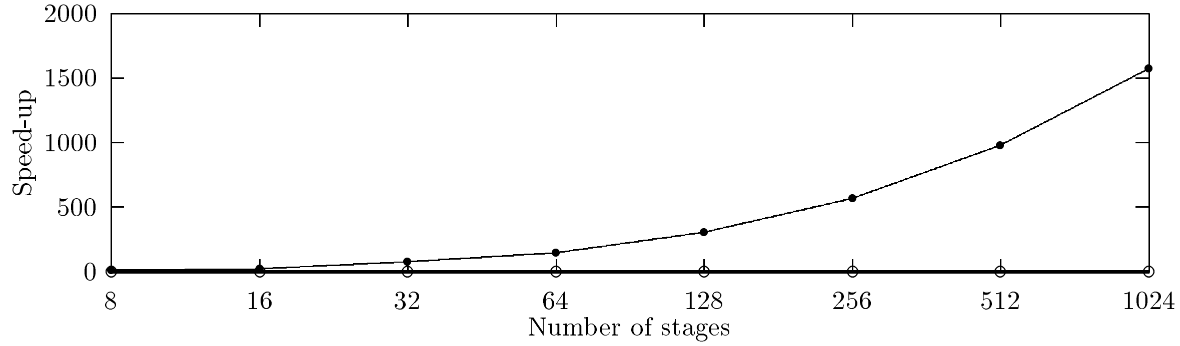
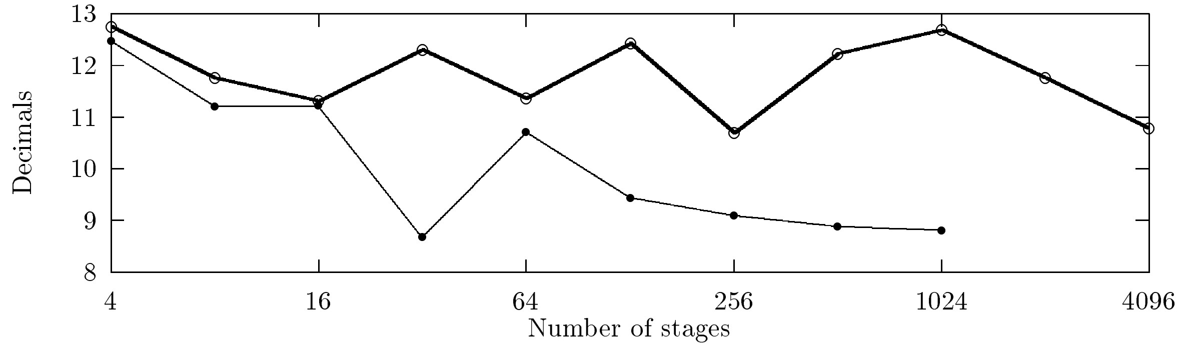


Figure 5.33: Accuracies for coupled robot TPBVP



less time than a second, whereas MA28A needs more than 2 seconds just to analyze the smallest one.

Although the accuracy plots in this section look similar to the ones in the previous section, there is a significant difference. The characteristic pattern in the error distribution of MSKKT cannot be observed any more. This pattern was caused by the specific treatment of decoupled constraints, which are now completely absent. In the coupled first order problems (type b1) a seemingly uniform distribution of errors over the whole grid is observed instead, whereas second order and robot optimization systems (types b2 and b3) show decreasing accuracies near the boundaries.

5.2.4 Multipoint boundary value problems

In this section the test problems have globally coupled constraints,

$$F_1 \Delta x_1 + \cdots + F_m \Delta x_m + e_* = 0. \tag{5.11}$$

The global blocks F_j are all full-rank dense random blocks, whereas none of the local blocks of the constraint matrix are present. Again, these problems cannot be handled by the band solver, and a comparison is only given for MA28 and the recursive solver.

Figure 5.34: CPU times for 1st order MPBVP, $n_x = 16$

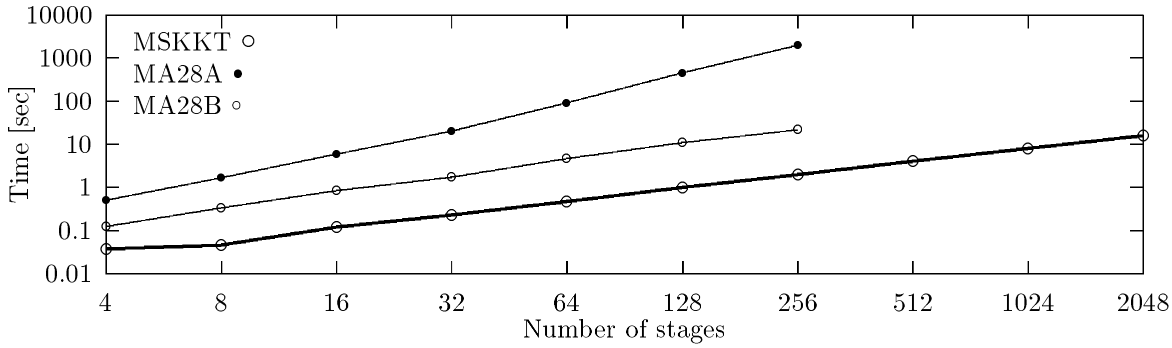


Figure 5.35: Speed-up for 1st order MPBVP, $n_x = 16$

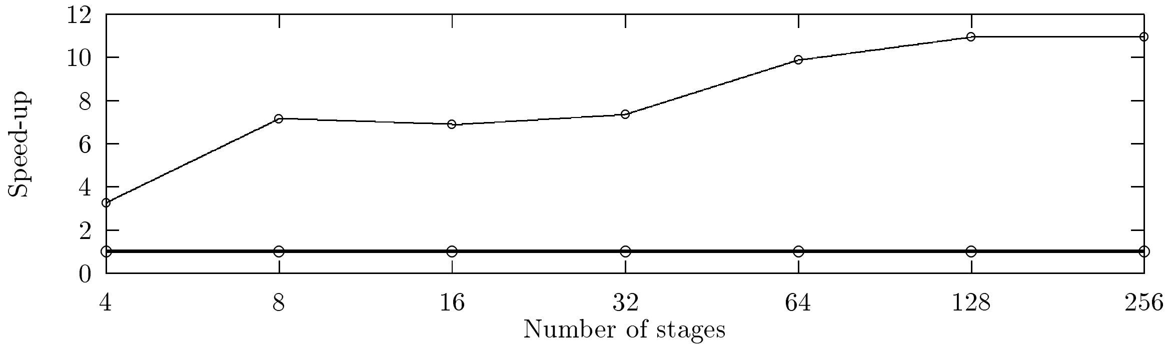
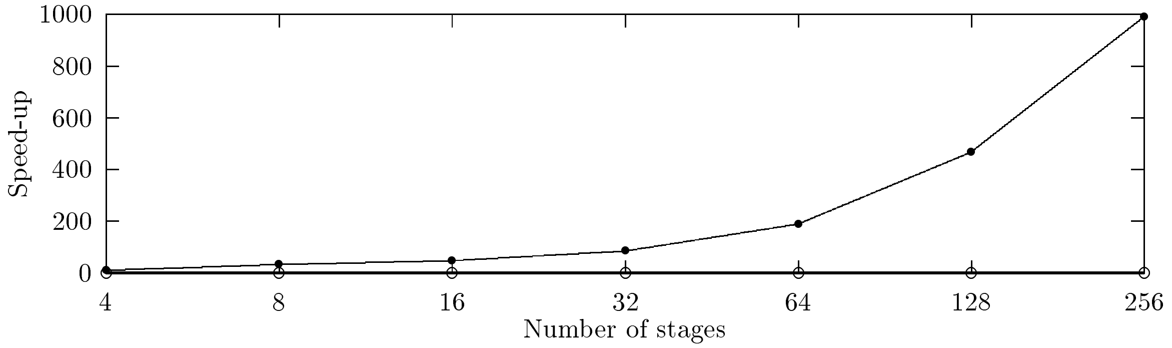


Figure 5.36: Speed-up for 1st order MPBVP, $n_x = 16$



5.2.4.1 First order systems

Here we choose the case $n_x = n_u = 16$ for illustrations; the results are given in figures 5.34–5.37. As expected, the global coupling does not show any effect on the runtime behavior of MSKKT. The same observation applies to MA28. Complexities of the analyze phase and remaining phases are still quadratic and linear, respectively. However, the relative efficiency of MSKKT increases as compared to MA28B. On small grids the speed-up is about 7, and the factor exceeds 10 when the number of stages is 64 or greater.

Figure 5.37: Accuracies for 1st order MPBVP, $n_x = 16$

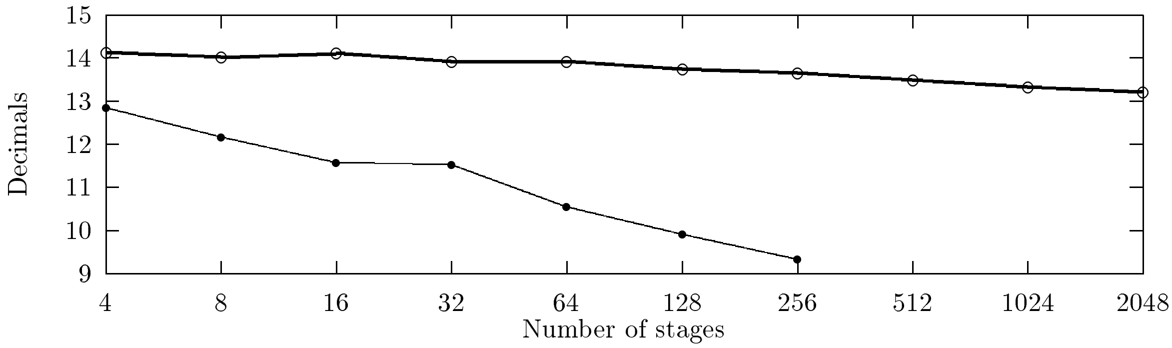


Figure 5.38: CPU times for 2nd order MPBVP, $n_x = 16$

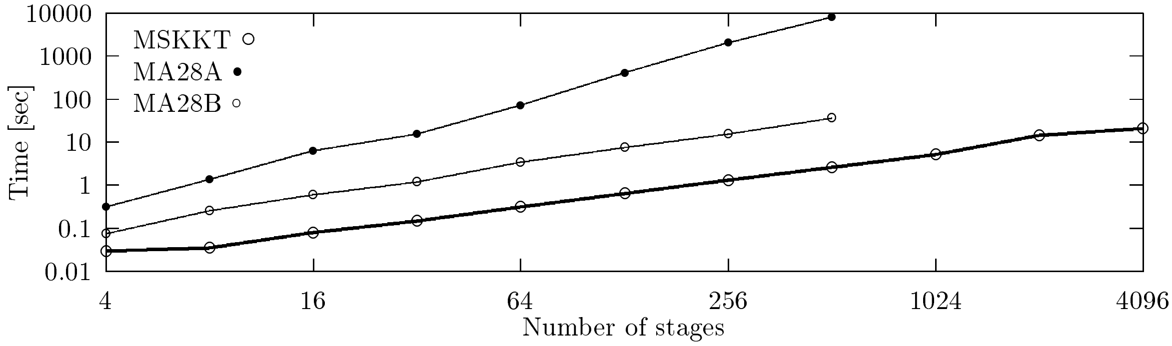
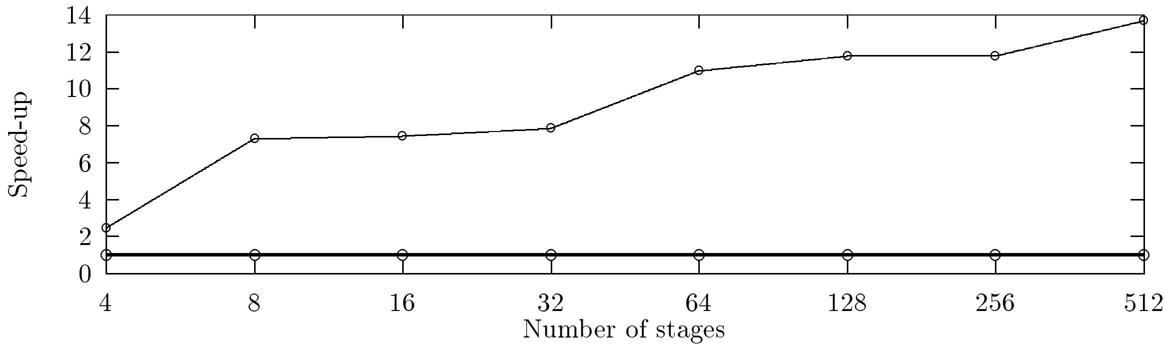


Figure 5.39: Speed-up for 2nd order MPBVP, $n_x = 16$



Apparently there also is no negative effect on the accuracy of MSKKT; it is rather high over the whole range of grid sizes. The accuracy of MA28, on the other hand, decreases rapidly when m is increased. Whereas 13 significant digits are achieved at $m = 4$, only one decimal less than with MSKKT, the difference is greater than 4 significant digits at $m = 256$. A final examination of the error distribution shows the same behavior as for problem type (b1) in the previous section: no specific pattern appears. This will remain true in the two final cases of second order and robot optimization MPBVP.

Figure 5.40: Speed-up for 2nd order MPBVP, $n_x = 16$

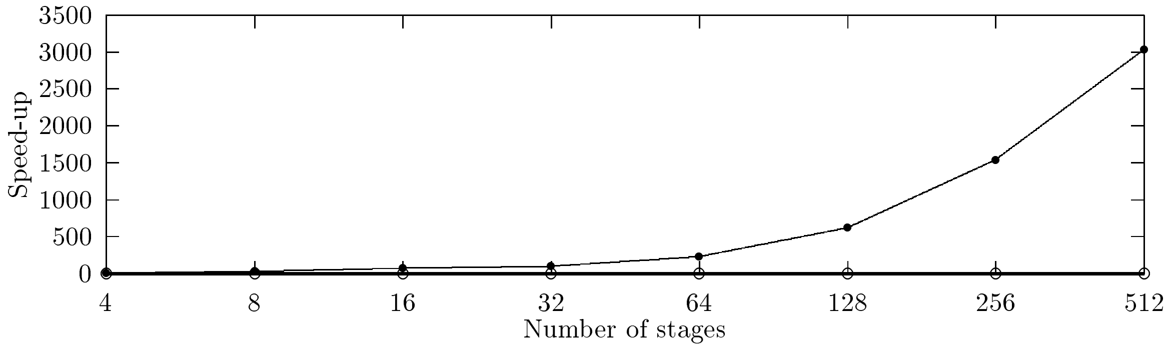
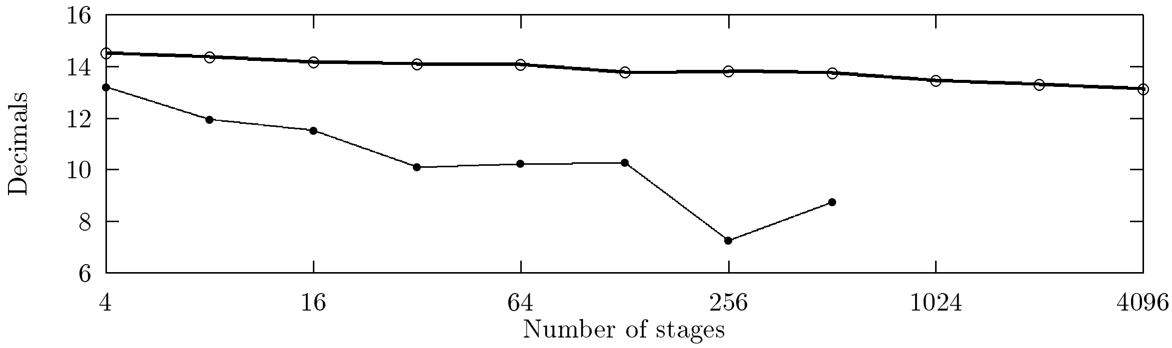


Figure 5.41: Accuracies for 2nd order MPBVP, $n_x = 16$



5.2.4.2 Second order systems

Again we choose $n_x = 2n_u = 16$ for illustrations; results are given in figures 5.38–5.41. The runtime behavior of both solvers is similar to the previous case, but the advantage of MSKKT over MA28 is even greater. On the 256-stage problem MSKKT was 1000 times as fast as MA28A in the previous case, here it is about 1500 times as fast. The corresponding speed-up factors versus MA28B are about 11 and 14, respectively. Similarly, the difference of accuracies also increases. Whereas MSKKT achieves more than 13 significant digits in all cases, the accuracy of MA28 drops below 9 digits for $m \geq 256$.

5.2.4.3 Robot optimization systems

The results for this final problem type are given in figures 5.42–5.45. Runtimes and accuracies of both solvers are very similar to the previous two problem types.

Let us again consider absolute runtimes for the problems with 16, 64, and 256 stages. MSKKT works 0.06, 0.19 and 0.85 seconds on these problems. MA28B needs between 6 and 10 times as much, taking 0.33, 1.74 and 8.81 seconds, respectively. MA28A, finally, solves the problems after 2.90 seconds, 35.7 seconds, and about 13 minutes (782 seconds). The analyze phase takes 2.57 seconds on the 16-stage problem, which is almost exactly three times as much as MSKKT needs to solve the 256-stage problem.

Figure 5.42: CPU times for robot MPBVP

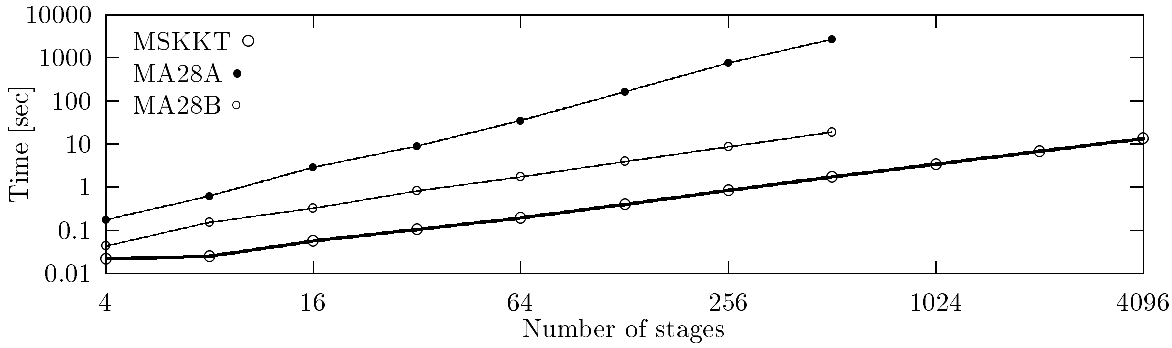


Figure 5.43: Speed-up for robot MPBVP

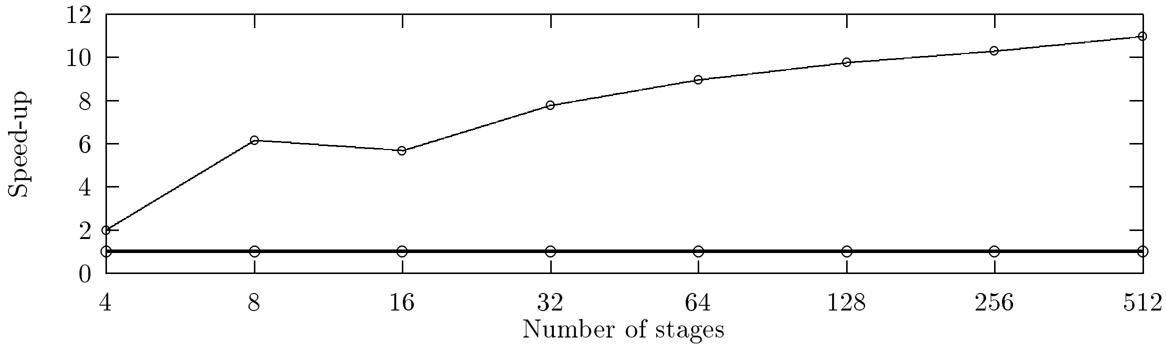
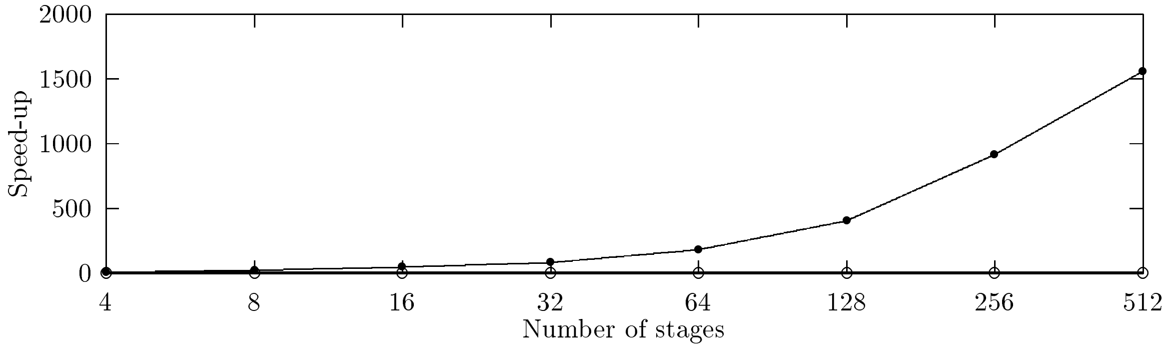


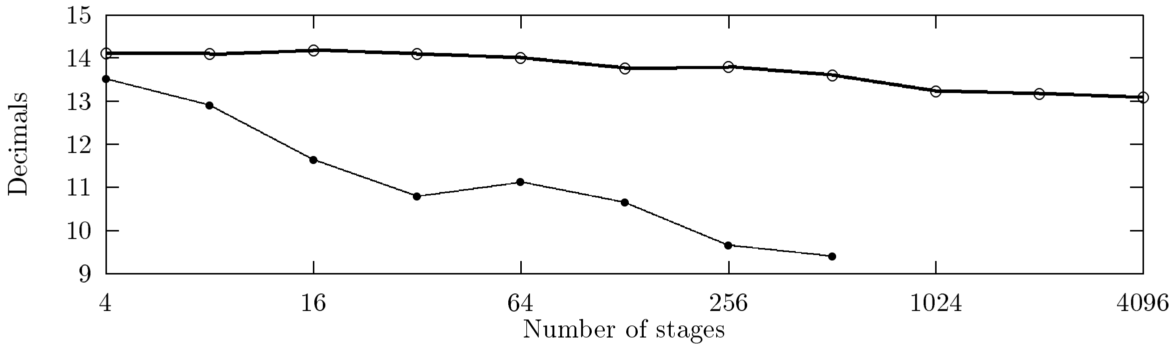
Figure 5.44: Speed-up for robot MPBVP



5.2.5 Summary

Results of the numerical tests show that the new recursive KKT solver MSKKT performs highly satisfactorily on the whole range of test problems. MSKKT is always the fastest among the tested solvers, except for very small problems. Moreover, it needs the smallest amount of storage by far and achieves surprisingly high solution accuracies. Only a few problems are solved with less than 10 digits in the least accurate component. An additional positive feature is the characteristic distribution of errors among the solution

Figure 5.45: Accuracies for robot MPBVP



components in the presence of local constraints. MSKKT tends to favor state and control components (primal variables), so that low accuracies are typically found in components of the Lagrange multipliers (dual variables). This behavior matches precisely the requirements in an SQP method. On coupled problems (without local constraints) there is no such specific pattern in the error distribution of MSKKT, but then the recursive algorithm shows always more accurate than MA28.

The LAPACK band solver proves almost competitive for the case of decoupled constraints. It is 2 to 10 times slower but often more accurate than MSKKT. Of course, the band solver is useless in the presence of coupled constraints.

The general sparse solver MA28 is seriously handicapped by the necessity of analyzing the sparsity pattern of the KKT matrix. If one could supply the structural information directly, then MA28 would run about 2 to 10 times slower than MSKKT, exactly as the band solver. The analyze phase, however, usually takes several dozen or hundred times as long as a complete solution by MSKKT. Still the accuracy of MA28 is often unsatisfactorily, especially on problems with coupled boundary conditions. Therefore MA28 is not considered competitive, even if only one initial structure analysis is needed in order to perform a complete SQP iteration.

5.3 KKT systems in multibody dynamics

It turns out that the inverse dynamics equations of certain multibody systems (MBS) have the same m-stage block-sparse structure as KKT systems in optimal control. Hence, the same solution algorithms may be applied in MBS simulation. For an overview on multibody system dynamics and further references see [75].

In the following we give a short introduction to multibody system dynamics in descriptor form. A chain of linear segments is then studied as an example. Finally, numerical results are briefly summarized for the simulation of chains with and without kinematic loops. A detailed presentation will be given in a forthcoming paper.

5.3.1 Multibody systems in descriptor form

Consider an idealized multibody system consisting of rigid bodies and point masses. We use redundant coordinates p for the position of the system, i.e., for describing the position and orientation of each individual body. (An example will be studied in the next section.) The set of feasible positions, which correspond to physically possible geometric configurations, is given by *holonomic* constraints

$$c(p) = 0. \tag{5.12}$$

We assume that this set forms a differentiable manifold, the configuration manifold $\mathcal{M}$. Although the configuration manifold is usually bounded in practice, we will consider it open here for simplicity.

Differentiation of the constraints with respect to time yields corresponding constraints for the velocity $\dot{p}$,

$$c'(p)\dot{p} = C(p)\dot{p} = 0, \tag{5.13}$$

and for the acceleration $\ddot{p}$,

$$C(p)\ddot{p} + \frac{\partial(C(p)\dot{p})}{\partial p}\dot{p} = C(p)\ddot{p} - \gamma(p, \dot{p}) = 0. \tag{5.14}$$

Here $C(p)$ is the constraint matrix; its kernel defines the tangent space to $\mathcal{M}$ at p, i.e., the local linearization of $\mathcal{M}$. In addition to this first order description the acceleration constraint involves curvature information represented in the term $\gamma(p, \dot{p})$.

The internal dynamic interactions of the multibody system are comprised in a single function, the Lagrangian

$$L(p, \dot{p}, \lambda) = T(p, \dot{p}) - V(p) + c(p)^T\lambda, \tag{5.15}$$

which involves the difference of kinetic and potential energy of the multibody system, and in addition the constraint potential term $c(p)^T\lambda$. Here the kinetic energy is a quadratic function of the velocity; it can always be written

$$T(p, \dot{p}) = \frac{1}{2}\dot{p}^T M(p)\dot{p}. \tag{5.16}$$

The mass matrix $M(p)$ describes the position-dependent inertia of the multibody system; it is always positive definite on the tangent space $\ker(C(p))$ of $\mathcal{M}$, provided that no massless bodies are present. Hence M defines a natural Riemannian metric on $\mathcal{M}$.

The motion of the multibody system is governed by the Euler-Lagrange equations. These are derived from the Lagrangian according to

$$\frac{d}{dt}\frac{\partial L}{\partial \dot{p}} - \frac{\partial L}{\partial p} = F^T, \tag{5.17}$$

where F denotes external forces. The Euler-Lagrange equations constitute a generalized form of Newton's force law,

$$M(p)\ddot{p} = f(t, p, \dot{p}) + C(p)^T\lambda, \tag{5.18}$$

where

$$
\begin{aligned}
f(t,p,\dot{p}) &= F(t) + \frac{1}{2}\left(\frac{\partial(\dot{p}^T M(p)\dot{p})}{\partial p}\right)^T - \frac{\partial(M(p)\dot{p})}{\partial p}\dot{p} - V'(p)^T, && (5.19)\\
&= F(t) - M(p)\Gamma(p)(\dot{p},\dot{p}) - V'(p)^T. && (5.20)
\end{aligned}
$$

Here the coupling term $C(p)^T\lambda$ describes constraint forces that appear in the joints connecting individual bodies. These forces are orthogonal to the manifold $\mathcal{M}$. The remaining forces are internal and external ones, where $\Gamma(p)$ denotes the Levi-Cività covariant derivation associated with $M(p)$. As is well-known, $\Gamma(p)(\dot{p},\dot{p})$ describes the quadratic gyroscopic accelerations, i.e., centrifugal and Coriolis accelerations.

In combination with the constraint equations on acceleration level, (5.14), the Euler-Lagrange equations constitute a DAE of index 1 for the differential variables p, v and algebraic variable λ,

$$
\begin{aligned}
\dot{p} &= v, && (5.21)\\
\begin{pmatrix} M(p) & C(p)^T \\ C(p) & \end{pmatrix}\begin{pmatrix} \dot{v} \\ -\lambda \end{pmatrix} &= \begin{pmatrix} f(t,p,v) \\ \gamma(p,v) \end{pmatrix}. && (5.22)
\end{aligned}
$$

The constraints on position level (5.12, index 3) and velocity level (5.13, index 2) are invariants of the DAE,

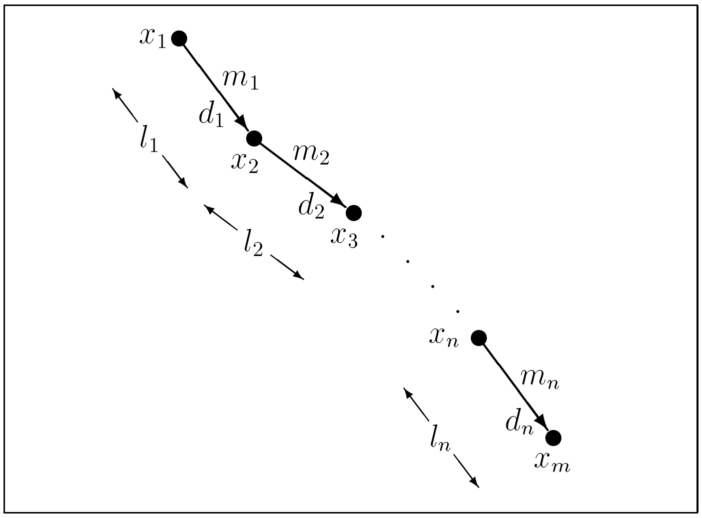
$$
\begin{aligned}
c(p) &= 0, && (5.23)\\
C(p)v &= 0. && (5.24)
\end{aligned}
$$

That is, they are automatically conserved along solution trajectories of (5.21,5.22) when the initial values (p_0, v_0) satisfy (5.23,5.24). Obviously, the linear system for $(\dot{v}, -\lambda)$ has the structure of a KKT system.

Various formulations for the dynamic equations of a specific multibody system are usually possible depending on the choice of coordinates, p, and associated geometric constraints, $c(p)$. In the special case of non-redundant (minimal) coordinates the formulation is called a *state space form* rather than a descriptor form. No algebraic constraints are then needed, and DAE (5.21,5.22) reduces to ODE (1.13) with a positive definite mass matrix $M(p)$, cf. the robot example in chapter 1.

In order to simulate the motion of a multibody system, an initial value problem for the DAE must be solved numerically. Although invariants (5.23,5.24) are satisfied along the exact solution, numerical roundoff errors cause the velocity and position variables to drift off from the manifold. If the index 1 constraint is satisfied exactly, the velocity error grows linearly as a function of time, and the position error grows quadratically. This drift-off behavior can be prevented by using appropriate projections during numerical integration. A factorization of the KKT system (5.22) can be used in the integration as well as in the projection, which makes the projection technique very efficient. For more information see, e.g., [2, 22, 5].

Figure 5.46: An n-link chain



5.3.2 The kinematic chain

Consider a chain in $I\!R^3$ consisting of n links, see figure 5.46. Each link is modeled as a line segment of length l_j and mass m_j, with uniform mass distribution. We choose inertial Cartesian coordinates of the joints, x_j, and the vectors connecting the joints along the links, d_j, in order to describe the position p of the chain:

$$p = (p_1, \dots, p_m), \quad m = n + 1, \tag{5.25}$$

where

$$p_j = (x_j, d_j), \quad j = 1(1)n, \qquad p_m = x_m. \tag{5.26}$$

(The last joint has no link attached to it.) Here the holonomic constraints include length conditions

$$\|d_j\|^2 = l_j^2 \tag{5.27}$$

and connection conditions

$$x_{j+1} = x_j + d_j, \tag{5.28}$$

yielding

$$c(p) = \begin{pmatrix} c_1(p_1, x_2) \\ \vdots \\ c_n(p_n, x_m) \end{pmatrix}, \quad c_j(p_j, x_{j+1}) = \begin{pmatrix} \frac{1}{2}(\|d_j\|^2 - l_j^2) \\ x_j + d_j - x_{j+1} \end{pmatrix}. \tag{5.29}$$

These restrictions leave three degrees of freedom for the initial joint and two degrees of freedom for each link; the resulting state manifold is isomorphic to

$$\mathcal{M} = I\!R^3 \times \underbrace{S^2 \times \cdots \times S^2}_{n \text{ times}}. \tag{5.30}$$

Velocity and acceleration constraints are represented by

$$C(p) = \begin{pmatrix} C_1(p_1) \ P_1 & & \\ & \ddots \quad \ddots & \\ & & C_n(p_n) \ P_n \end{pmatrix}, \quad (C_j(p_j) \ P_j) = \begin{pmatrix} 0 & d_j^T & 0 & 0 \\ I & I & -I & 0 \end{pmatrix} \tag{5.31}$$

and

$$\gamma(\dot p) = \begin{pmatrix} \gamma_1(\dot p_1) \\ \vdots \\ \gamma_n(\dot p_n) \end{pmatrix}, \quad \gamma_j(\dot p_j) = \begin{pmatrix} -\|\dot d_j\|^2 \\ 0 \end{pmatrix}. \tag{5.32}$$

Due to the uniform mass distribution, the center of mass of each link coincides with ist geometric center, $x_j + d_j/2$, and the moment of inertia about the center of mass is $\frac{1}{3}m_j(l_j/2)^2$. Hence, the total kinetic energy is given by

$$\begin{aligned} T(\dot p) &= \sum_{j=1}^n \frac{1}{2}\left[m_j \left\|\dot x_j + \dot d_j/2\right\|^2 + \frac{1}{3}m_j(l_j/2)^2 \left\|\dot d_j/l_j\right\|^2\right] && (5.33) \\ &= \sum_{j=1}^n \frac{1}{2}m_j \begin{pmatrix} \dot x_j \\ \dot d_j \end{pmatrix}^T \begin{pmatrix} I & \frac{1}{2}I \\ \frac{1}{2}I & \frac{1}{3}I \end{pmatrix} \begin{pmatrix} \dot x_j \\ \dot d_j \end{pmatrix} = \frac{1}{2}\dot p^T M \dot p. && (5.34) \end{aligned}$$

From this it can easily be seen that the mass matrix is constant and block-diagonal,

$$M = \operatorname{diag}(M_1, \ldots, M_n, 0), \quad M_j = m_j \begin{pmatrix} I & \frac{1}{2}I \\ \frac{1}{2}I & \frac{1}{3}I \end{pmatrix}. \tag{5.35}$$

As a consequence, $\Gamma(p)$ vanishes everywhere for our choice of coordinates.

The potential energy due to gravity is given by

$$V(p) = \sum_{j=1}^n m_j g e_v^T (x_j + d_j/2), \tag{5.36}$$

yielding the force vector

$$f(p) = \begin{pmatrix} f_1(p) \\ \vdots \\ f_n(p) \\ 0 \end{pmatrix}, \quad f_j(p) = -m_j g \begin{pmatrix} e_v \\ \frac{1}{2}e_v \end{pmatrix}. \tag{5.37}$$

Here e_v is the unit vector along the vertical axis, $e_v^T(x_j + d_j/2)$ is the height of the center of mass of link j above zero level, and $g \approx 9.81\,\mathrm{m/s^2}$ is the gravity constant.

It may be noted that a similar model for a planar chain takes exactly the same form. In that case the state manifold is

$$\mathcal{M} = \mathbb{R}^2 \times \underbrace{S^1 \times \cdots \times S^1}_{n \text{ times}}. \tag{5.38}$$

The functions given above constitute the index 1 DAE for the kinematic chain,

$$\dot{p} \quad = \quad v, \tag{5.39}$$

$$\begin{pmatrix} M & C(p)^T \\ C(p) & \end{pmatrix} \begin{pmatrix} \dot{v} \\ -\lambda \end{pmatrix} \quad = \quad \begin{pmatrix} f(p) \\ \gamma(v) \end{pmatrix}, \tag{5.40}$$

with invariants

$$c(p) \quad = \quad 0, \tag{5.41}$$

$$C(p)v \quad = \quad 0. \tag{5.42}$$

Obviously the chain structure of our multibody system induces an m-stage block-sparse structure in the associated KKT system (5.40). Joint accelerations $\ddot{x}_j$ correspond to state variables Δx_j, and link accelerations $\ddot{d}_j$ play the role of control variables Δu_j. The block diagonal mass matrix corresponds to the Hessian, its blocks are

$$\begin{pmatrix} H_j & {}^tJ_j \\ J_j & K_j \end{pmatrix} = \begin{pmatrix} I & \frac{1}{2}I \\ \frac{1}{2}I & \frac{1}{3}I \end{pmatrix}, \quad H_m = 0. \tag{5.43}$$

Connection conditions play the role of continuity conditions, where the transition matrices are simply

$$(G_j \; E_j) = (I \;\; I). \tag{5.44}$$

Finally, the length conditions are similar to local control constraints, their constraint matrices being

$$D_j^u = d_j^T. \tag{5.45}$$

Additional constraints may be imposed to account for other geometric conditions. If, say, the initial joint is fixed at $x_1 = 0$, we introduce a local state constraint with matrix

$$F_1^x = I. \tag{5.46}$$

(Alternatively one can simply omit x_1 from p_1.) In a more general case only the two horizontal coordinates of some joint x_j might be fixed, modeling the suspension of that joint on a “rail”. If, finally, two joints j and k are connected to each other, a closed kinematic loop is formed. This is accounted for by global constraints with matrices

$$F_j = I, \quad F_k = -I. \tag{5.47}$$

Below we will give an example for a general chain topology including such constraints.

5.3.3 Examples

As example multibody systems we consider planar chains with linear topology (n-link pendulums), and chains swinging in three dimensions, with a more complicated topology involving closed loops. Run times for KKT system solution by MSKKT and for complete dynamic simulations will be summarized; a detailed presentation and comparison with other KKT solvers will be given in a forthcoming paper.

5.3.3.1 Planar chains with linear topology

The chains studied in this section have a linear topology as described in section 5.3.2. They are allowed to swing in a vertical plane under the influence of gravity, while the first joint is fixed at the origin of the coordinate system. This initial condition is explicitly formulated as explained above; no other forces or restrictions are modeled. Table 5.1 shows the CPU times needed for solution of the resulting KKT systems (in seconds).

Table 5.1: CPU times for planar chain KKT systems solution

Number of links	Factorization	Transformation and solution	Total time
1	< 0.01	< 0.01	< 0.01
5	< 0.01	< 0.01	< 0.01
10	< 0.01	< 0.01	≈ 0.01
30	0.01	0.01	0.02
100	0.04	0.03	0.07
300	0.12	0.09	0.21

The numbers of links considered are $n = 1, 5, 10, 30, 100, 300$. In each of these cases the chain has a total length of 1 m and a total mass of 1 kg, and each individual link has length $(1/n)$ m and mass $(1/n)$ kg.

Due to extremely small block sizes the overhead for index computations and BLAS calls is apparently high here, and the transformation and solution phases together take almost as much time as the factorization. Only for thirty and more links are these times greater than the clock resolution.

More interesting observations are made considering the simulation results. We use the special MBS integrator MBSABM from the software package MBSSIM [91], which treats the index 1 DAE (5.21,5.22) directly and applies a projection technique, see [23]. This integrator is based on Adams-Bashforth-Moulton formulae of variable step size and variable order; it allows large steps even for high accuracies. In our examples the integration accuracy was always set to $\texttt{TOL} = 10^{-6}$.

For the simulation we start the chain in a horizontally stretched-out position and let it drop (figure 5.47). The swinging motion is then simulated for a duration of one second. This amounts to about half a period, so that finally the chain is approximately stretched out on the opposite side of the suspension point. When the chain swings back it starts behaving like a whip. The links penetrate each other and the motion becomes highly chaotic, therefore the simulation is stopped before this phase begins.

Table 5.2 shows the simulation results. The data given are the total CPU time for each simulation run, the numbers of successful and unsuccessful integration steps, the total number of f-calls (evaluations of the DAE right hand side), and the time spent on each integration step.

Figure 5.47: Swinging motion of the 30-link chain, frozen every 0.05 seconds

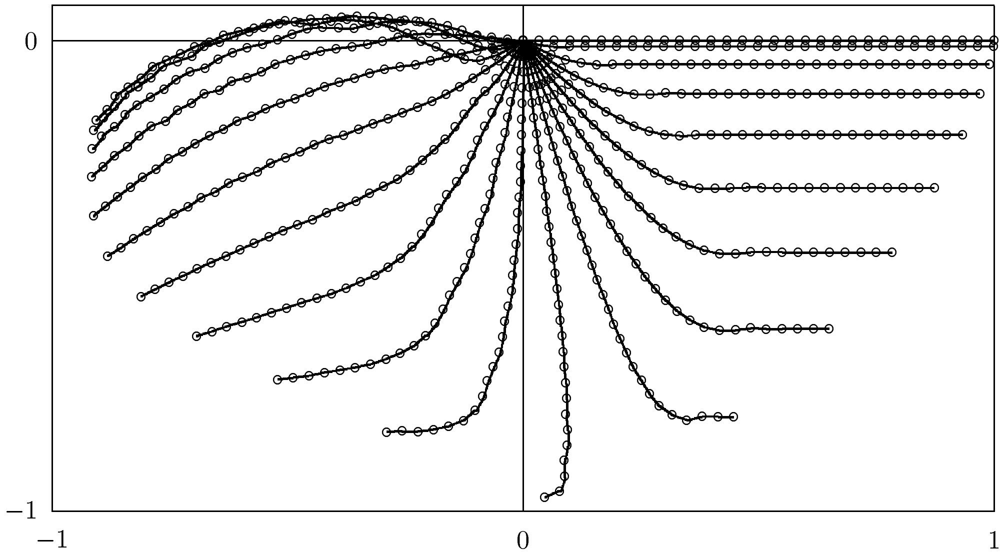


Table 5.2: Results of planar chain simulations

Number of links	CPU time (total)	Accepted steps	Rejected steps	f-calls	CPU time per step
1	0.2	59	0	126	0.003
5	3.0	365	0	738	0.008
10	10.6	639	1	1287	0.017
30	67.0	1494	1	3003	0.045
100	520.3	3504	0	7028	0.15
300	3652.0	8122	0	16264	0.45

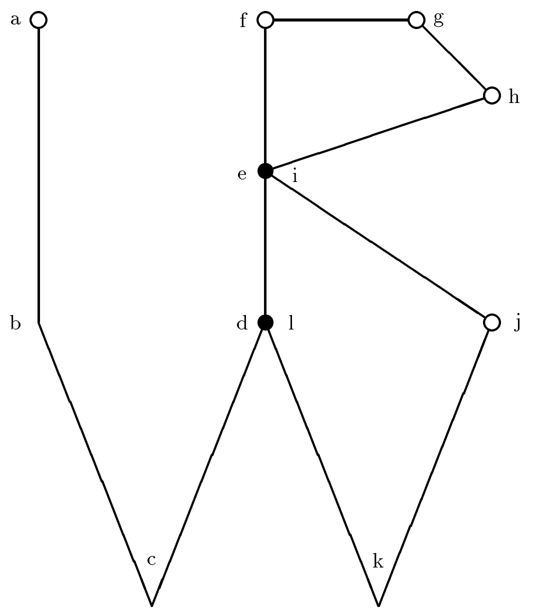
Each integration step requires only two f-calls, i.e., two evaluations of the KKT matrix and right hand side, plus two KKT system solutions. Therefore the integrator is very well-suited for problems with expensive right hand sides, as is the case here for large n.

As expected the time per step grows linearly with n. But although the order and step size selection scheme almost never rejects an integration step, the number of steps needed for a complete simulation also grows almost linearly with n. Therefore the total complexity is about $\mathcal{O}(n^2)$, as can be seen when the total CPU times are compared. This behavior occurs because the swinging motion excites waves in the chains, which are not damped due to the absence of friction. Since all the chains have equal length, their highest possible wave frequency grows linearly with n, and hence the integration becomes more expensive.

5.3.3.2 Chains with closed kinematic loops

Our final example, the IWR chain, has been invented for demonstration purposes. It is a folded chain modeling the three capital letters 'I', 'W', 'R' of our institution's name. (The small letters in figure 5.48 indicate the linear order of links in the chain.)

Figure 5.48: Schematic diagram of the IWR chain



Since 'W' hangs below 'I' and 'R', its right hand part forms a closed loop with the 'R'. A second closed loop is formed by the upper part of 'R'; the black circles in figure 5.48 mark the positions where parts of the chain are connected. Note that each line segment in the figure represents an unspecified number of links rather than a single link connecting two labeled positions. (Otherwise there would be very little freedom for swinging.)

The chain is suspended in five joints so that its geometric configuration does not change too much when it is in motion. The suspension points are the uppermost joint of 'I', and the two upper and two right corners of 'R', marked by the open circles in figure 5.48. However, these joints are not completely fixed in spatial positions but only in the two dimensions which determine the geometry of the letters. They may move in the direction perpendicular to the drawing plane as if sliding on horizontal rails. Therefore the whole configuration can swing back and forth in three dimensions.

As with the unfolded chain, configurations with different numbers of links have been constructed in such a way that all the links have equal mass and length, with a total mass of 1 kg and a total length of 16 m (roughly 2 m for 'I', 8 m for 'W', and 6 m for 'R'). In order to give enough degrees of freedom for a true 3-dimensional swinging motion, even the shortest labeled segment, (gh), must consist of at least two links. Under these conditions, the smallest reasonable number of links was found to be 42 (of course! [1]).

Other numbers considered are 1994 and certain multiples of 42. Table 5.3 gives the structural data for each of these cases, i.e., the joint index and planar coordinates of each suspension point, and the joint indices of the first and last joint of each loop.

Table 5.3: Structural data for the IWR chains

	$k \times 42$		1994	
Point (a)	1	$(-1.00, 2.00)$	1	$(-1.00, 2.00)$
Point (f)	$21k+1$	$(\ 0.00, 2.00)$	998	$(\ 0.00, 2.00)$
Point (g)	$23k+1$	$(\ 2/3, 2.00)$	1085	$(\ 2/3, 2.00)$
Point (h)	$25k+1$	$(\ 2.00, 1.35)$	1148	$(\ 2.00,\ 5/3)$
Point (j)	$32k+1$	$(\ 1.00, 0.00)$	1497	$(\ 1.00, 0.00)$
Loop (e,i)	$(18k+1, 28k+1)$		$(873, 1310)$	
Loop (d,l)	$(16k+1, 42k+1)$		$(749, 1995)$	

All the global and local constraints introduced by closed loops and suspension on rails are formulated as described in section 5.3.2. CPU times for the solution of the resulting KKT systems are given in table 5.4. Notice that both loops span only certain regions of the chain, which causes a step-pyramid structure in the matrix of global constraints. (The block associated with the smaller loop, spanning positions e–i, sits on top of the block associated with the bigger loop, spanning positions d–l.) Approximately 20 % of the CPU time could be saved if this substructure was exploited.

Table 5.4: CPU times for IWR chain KKT systems solution

Number of links	Factorization	Transformation and solution	Total time
42	0.02	0.02	0.04
84	0.05	0.03	0.08
126	0.08	0.05	0.13
168	0.10	0.07	0.17
210	0.13	0.09	0.22
420	0.25	0.18	0.43
1994	1.18	0.81	1.99

Simulation runs for the IWR chain were performed with the same MBS integrator and accuracy as in the case of planar chains. A swinging motion was started as follows. In its initial state the chain is completely stretched out, i.e., the links are in such positions that their projections onto the drawing plane form straight segments between each pair of labeled joints as in figure 5.48. For this purpose the right part of 'R' was pushed back on rails (g,h,j), position (f) was fixed, and the 'I' was pulled forward on rail (a), with joint (b) remaining exactly below it. Joints (c,d,e,i,k,l) were then suitably adjusted. From such a position the chain starts swinging back and forth as well as to the left and right in a complicated motion. However, it never behaves like a whip since the chain has no open ends.

The motion has been simulated over several periods of the slow back-and-forth mode in some cases, but for comparisons it suffices to use a shorter simulation time. Table 5.5 shows the results for a time of 0.1 seconds; the data given are the same as in the case of planar chains. (Notice that the 42-link chain needs about 420 integration steps and 42 seconds computing time.)

Table 5.5: Results of IWR chain simulations

Number of links	CPU time (total)	Accepted steps	Rejected steps	f-calls	CPU time per step
42	42.2	421	0	874	0.10
84	146.1	749	0	1536	0.20
126	326.0	1109	0	2256	0.29
168	567.5	1461	0	2960	0.39
210	849.7	1734	0	3512	0.49
420	2961.2	3141	0	6326	0.84
1994	65054.2	14193	0	28442	4.58

Appendix A

Generalized optimal control problems

In this appendix we discuss optimal control problems that are given in a more general form than [OCP]. All these generalized problems can be reduced to our standard form; the transformations will be given explicitly. Quantities appearing in the generalized problems will always be distinguished by a hat ($\hat{}$) if a corresponding transformed quantity appears in the standard form problem.

A.1 Parameters and non-autonomous problems

Many optimal control problems involve certain unknown parameters $p \in I\!R^{n_p}$ which have to be determined in addition to the optimal state and control functions x, u. Such parameters are simply treated as state variables with trivial differential equation, $\dot{p} = 0$. (Detailed formulae will be given below.) Parameters may be categorized in two groups. *System parameters*, such as the initial fuel load of an aircraft, may be chosen at the beginning of a specific process for an existing system. *Design parameters*, such as the length of a robot link, are chosen before constructing the system, and may not be altered afterwards.

In realistic applications the final time $\hat{t}_k$ is often unknown a priori; in many cases it is actually the objective of minimization. A typical example is the PTP maneuver of a robot manipulator; cf. section 1.1.2. Problems with free final time are transformed to the standard form via a time transformation $[0, \hat{t}_k] \ni \hat{t} \mapsto t \in [0, 1]$, and $\hat{t}_k$ becomes a system parameter.

If any of the problem functions is explicitly time-dependent, each of the time instances $\hat{t}_i$ may be free for optimization. A typical example is an interplanetary mission in space exploration. In order to minimize, say, the flight duration $\hat{t}_k - \hat{t}_1$ or the fuel consumption, one has to determine optimal choices of the initial time $\hat{t}_1$ and of rendezvous times $\hat{t}_i$ for swing-by maneuvers according to the constellation of the planetary system. Again, a time transformation is applied, and $\hat{t}_1, \ldots, \hat{t}_k$ are treated as system parameters.

All the transformations for the general case will now be given in detail. Consider a non-autonomous optimal control problem with system parameters and unknown initial, final and interior times $\hat{t}_i$,

$$\begin{aligned}
\hat{\phi}(\hat{x}(\hat{t}_k), \hat{t}_k, p) &= \min, && \text{(A.1)} \\
\hat{r}(\hat{x}(\hat{t}_1), \hat{t}_1, \ldots, \hat{x}(\hat{t}_k), \hat{t}_k, p) &= 0, && \text{(A.2)} \\
\dot{\hat{x}}(\hat{t}) - \hat{f}(\hat{x}(\hat{t}), \hat{u}(\hat{t}), \hat{t}, p) &= 0, && \text{(A.3)} \\
\hat{g}(\hat{x}(\hat{t}), \hat{u}(\hat{t}), \hat{t}, p) &\geq 0. && \text{(A.4)}
\end{aligned}$$

Here we make the same general assumptions as in chapter 1: $\hat{f}, \hat{g}, \hat{r}, \hat{\phi}$ are assumed to be C^2-functions, function spaces for controls and states are given by $\hat{u} \in L^\infty([\hat{t}_1, \hat{t}_k], I\!R^{\hat{n}_u})$ and $\hat{x} \in H^{1,\infty}([\hat{t}_1, \hat{t}_k], I\!R^{\hat{n}_x})$, and multipoint boundary conditions $\hat{r}$ are second order decoupled in the sense that

$$\frac{\partial^2 \hat{r}}{\partial y \partial z} = 0 \quad \forall y, z \in \left\{ \begin{pmatrix} \hat{x}(\hat{t}_i) \\ \hat{t}_i \end{pmatrix}, p \right\}. \qquad \text{(A.5)}$$

In addition we require that the *order* of the free time instances is known, $\hat{t}_1 < \cdots < \hat{t}_k$.

Lemma 2 *Problem (A.1–A.4) is equivalent to the standard optimal control problem (1.1–1.4) if the problem functions are transformed as follows.*

Choose any fixed time instances $0 = t_1 < \cdots < t_k = 1$ *and apply a piece-wise linear time transformation,*

$$\begin{array}{ccl}
[\hat{t}_i, \hat{t}_{i+1}] \ni & \hat{t} & \\
\downarrow & \downarrow & \\
[t_i, t_{i+1}] \ni & t & = t_i + \dfrac{t_{i+1} - t_i}{\hat{t}_{i+1} - \hat{t}_i}(\hat{t} - \hat{t}_i).
\end{array} \qquad \text{(A.6)}$$

Include the original time variable and all the parameters in the new state vector

$$x(t) = \begin{pmatrix} \hat{x}(\hat{t}) \\ \hat{t} \\ p \\ \hat{t}_1 \\ \vdots \\ \hat{t}_k \end{pmatrix}, \qquad n_x = \hat{n}_x + 1 + n_p + k, \qquad \text{(A.7)}$$

and define

$$u(t) = \hat{u}(\hat{t}). \qquad \text{(A.8)}$$

This yields the transformed cost function

$$\phi(x(1)) = \hat{\phi}(\hat{x}(\hat{t}_k), \hat{t}_k, p), \qquad \text{(A.9)}$$

boundary conditions

$$r(x(t_1), \ldots, x(t_k)) = \hat{r}(\hat{x}(\hat{t}_1), \hat{t}_1, \ldots, \hat{x}(\hat{t}_k), \hat{t}_k, p), \qquad \text{(A.10)}$$

and path constraints

$$g(x(t), u(t)) = \hat{g}(\hat{x}(\hat{t}), \hat{u}(\hat{t}), \hat{t}, p). \tag{A.11}$$

The differential equations must be augmented in order to accommodate the new state components. A piece-wise definition according to the time transformation yields

$$f(x(t), u(t)) = \begin{pmatrix} \hat{f}(\hat{x}(\hat{t}), \hat{u}(\hat{t}), \hat{t}, p) \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \cdot \frac{\hat{t}_{i+1} - \hat{t}_i}{t_{i+1} - t_i} \tag{A.12}$$

on subinterval (t_i, t_{i+1}).

Proof. The parameter p is regarded as component of $x(t)$ in f, g, or of $x(t_k) = x(1)$ in r, ϕ, and $\hat{t}, \hat{t}_i$ are regarded as components of $x(t), x(t_i)$, respectively. Obviously the transformed problem functions f, g, r, ϕ then have the correct (standard) form. Moreover, it is easily verified that they are all C^2-functions and that r is second order decoupled since this is true for $\hat{f}, \hat{g}, \hat{r}, \hat{\phi}$. Finally one has to check that (x, u) is a $(H^{1,\infty}, L^\infty)$-solution of [OCP] if and only if $(\hat{x}, \hat{u})$ is a $(H^{1,\infty}, L^\infty)$-solution of (A.1–A.4), which is also easily verified. ∎

Remarks. As indicated in the lemma, the transformation is already uniquely determined by the time transformation (A.6) and definitions (A.7,A.8) (which implicitly involve A.6). No freedom of choice remains for the definitions of f, g, r, ϕ, except that p may be regarded as component of *any* of the $x(t_i)$ in r.

The piece-wise linear time transformation has only been chosen for simplicity. Any piece-wise differentiable function $t = \varphi(\hat{t})$ satisfying $\varphi(\hat{t}_i) = t_i$ and $\varphi'(\hat{t}) > 0$ might be used instead.

Observe finally that the piece-wise definition of f actually implies that different differential equations apply on subintervals (t_i, t_{i+1}). This is a slight but reasonable extension of the situation in chapter 1; it does not cause any conceptual difficulties. Similarly, different path constraints may apply on each subinterval.

A.2 Lagrange and Chebyshev cost functions

The Mayer cost function (1.1) in [OCP], which depends only on the terminal state $x(1)$, is one of the most typical cost functions. Another type of similar importance is the *Lagrange* cost function

$$\int_0^1 L(x(t), u(t))\, dt = \min \tag{A.13}$$

which models a continuous dependence of the cost on the whole trajectory. Sometimes a combination of both appears in the literature, the *Bolza* cost function

$$\phi(x(1)) + \int_0^1 L(x(t), u(t))\, dt = \min\,. \tag{A.14}$$

Finally we consider the *Chebyshev* type cost function

$$\max_{t\in[0,1]} \gamma(x(t), u(t)) = \min \tag{A.15}$$

which involves the global maximum of a certain function taken over the whole trajectory.

In order to describe the transformation to standard form for each of these three types, we consider an optimal control problem with a combined cost function,

$$\begin{aligned} \alpha_1\hat{\phi}(\hat{x}(1)) + \alpha_2 \int_0^1 L(\hat{x}(t), u(t))\, dt + \alpha_3 \max_{t\in[0,1]} \gamma(\hat{x}(t), u(t)) &= \min, && \text{(A.16)}\\ \hat{r}(\hat{x}(t_1), \ldots, \hat{x}(t_k)) &= 0, && \text{(A.17)}\\ \dot{\hat{x}}(t) - \hat{f}(\hat{x}(t), u(t)) &= 0, && \text{(A.18)}\\ \hat{g}(\hat{x}(t), u(t)) &\geq 0, && \text{(A.19)} \end{aligned}$$

where $\alpha_i \geq 0$. The Lagrange integral will be regarded as a new state component

$$x_2(t) = \int_0^t L(\hat{x}(\tau), u(\tau))\, d\tau, \tag{A.20}$$

and the Chebyshev term will be incorporated as an additional system parameter which is to be an upper bound for γ,

$$x_3 \geq \gamma(\hat{x}(t), u(t)) \qquad \forall t \in [0, 1]. \tag{A.21}$$

We assume that both L and γ are C^2-functions.

Lemma 3 *Problem (A.16–A.19) is equivalent to the standard optimal control problem (1.1–1.4) if the problem functions are transformed as follows.*

Introduce the Lagrange and Chebyshev terms as an additional state component and parameter, respectively,

$$x(t) = \begin{pmatrix} \hat{x}(t) \\ x_2(t) \\ x_3 \end{pmatrix}, \qquad n_x = \hat{n}_x + 2, \tag{A.22}$$

and obtain the cost function in Mayer form

$$\phi(x(1)) = \alpha_1\hat{\phi}(\hat{x}(1)) + \alpha_2 x_2(1) + \alpha_3 x_3. \tag{A.23}$$

The Lagrange term requires an initial condition $x_2(0) = 0$, leading to boundary conditions

$$r(x(t_1), \ldots, x(t_k)) = \begin{pmatrix} \hat{r}(\hat{x}(t_1), \ldots, \hat{x}(t_k)) \\ x_2(t_1) \end{pmatrix}, \qquad n_r = \hat{n}_r + 1, \tag{A.24}$$

and both new terms appear in the transformed differential equation,

$$f(x(t), u(t)) = \begin{pmatrix} \hat{f}(\hat{x}(t), u(t)) \\ L(\hat{x}(t), u(t)) \\ 0 \end{pmatrix}. \tag{A.25}$$

Finally, the Chebyshev term yields a new path constraint,

$$g(x(t), u(t)) = \begin{pmatrix} \hat{g}(\hat{x}(t), u(t)) \\ x_3 - \gamma(\hat{x}(t), u(t)) \end{pmatrix}, \qquad n_g = \hat{n}_g + 1. \tag{A.26}$$

Proof. As in the previous lemma, the assertion is easily verified. ∎

Remarks. A natural extension of the Mayer cost function (1.4) in the multistage context of [OCP] is a multipoint version

$$\sum_{i=1}^{k} \phi_i(x(t_i)) = \min . \tag{A.27}$$

In the framework of our structured direct SQP approach (chapter 2) we can actually treat a multistage Bolza cost function

$$\sum_{i=1}^{k} \phi_i(x(t_i)) + \sum_{i=1}^{k-1} \int_{t_i}^{t_{i+1}} L_i(x(t), u(t))\, dt = \min . \tag{A.28}$$

As replacement of (2.25) this yields a more general discretized cost function

$$F_1(y) = \sum_{j=1}^{m} F_{1j}(y_j) \tag{A.29}$$

which is still decoupled with respect to the stages. Therefore the Hessian remains block-diagonal, cf. theorem 9, and the whole structured approach remains applicable.

A.3 Nonlinearly coupled constraints

In this section we treat optimal control problems with nonlinearly coupled boundary conditions,

$$\begin{aligned} \hat{\phi}(\hat{x}(1)) &= \min, && \text{(A.30)} \\ \hat{r}(\hat{x}(t_1), \ldots, \hat{x}(t_k)) &= 0, && \text{(A.31)} \\ \dot{\hat{x}}(t) - \hat{f}(\hat{x}(t), u(t)) &= 0, && \text{(A.32)} \\ \hat{g}(\hat{x}(t), u(t)) &\geq 0. && \text{(A.33)} \end{aligned}$$

This problem has precisely the same form as [OCP], but we admit the most general class of constraints $\hat{r}$, that is, any pair of variables $\hat{x}(t_i), \hat{x}(t_j)$ may be nonlinearly coupled. In other words, the contribution of $\hat{r}$ to the Hessian of the Lagrangian is potentially completely dense.

Lemma 4 *Problem (A.30–A.33) is equivalent to the standard optimal control problem (1.1–1.4) with second order decoupled constraints if the problem functions are transformed as follows.*

Introduce the values $\hat{x}(t_2), \ldots, \hat{x}(t_k)$ *as system parameters* $x_2, \ldots, x_k$*,*

$$x(t) = \begin{pmatrix} \hat{x}(t) \\ x_2 \\ \vdots \\ x_k \end{pmatrix}, \qquad n_x = k \cdot \hat{n}_x. \tag{A.34}$$

This yields the cost function

$$\phi(x(1)) = \hat{\phi}(\hat{x}(1)) \tag{A.35}$$

and augmented boundary conditions

$$r(x(t_1), \ldots, x(t_k)) = \begin{pmatrix} \hat{r}(\hat{x}(t_1), x_2, \ldots, x_k) \\ \hat{x}(t_2) - x_2 \\ \vdots \\ \hat{x}(t_k) - x_k \end{pmatrix}, \qquad n_r = \hat{n}_r + (k-1)\hat{n}_x. \tag{A.36}$$

The transformed differential equations and path constraints are given by

$$f(x(t), u(t)) = \begin{pmatrix} \hat{f}(\hat{x}(t), u(t)) \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{A.37}$$

and

$$g(x(t), u(t)) = \hat{g}(\hat{x}(t), u(t)). \tag{A.38}$$

Proof. The parameters x_i are regarded as components of $x(t_1)$ in row 1 of r (the original constraint), and as components of $x(t_i)$ in row i of r (the new constraints). Obviously the boundary conditions are then even *first* order decoupled; they take the form

$$r(x(t_1), \ldots, x(t_k)) = \begin{pmatrix} r_1^0(x(t_1)) \\ \vdots \\ r_k^0(x(t_k)) \end{pmatrix}. \tag{A.39}$$

The equivalence of problems (A.30–A.33) and [OCP] is easily verified. ∎

Remarks. In the lemma we evaluate the original constraints at node $t_1 = 0$. Any other node t_i could of course be chosen, and the remaining state values would be introduced as dummy variables $x_1, \ldots, x_{i-1}$ and $x_{i+1}, \ldots, x_k$.

In practice it is very unlikely that *all* the states are nonlinearly coupled across *all* the stages. Therefore one will usually need much less than $(k-1)\hat{n}_x$ dummy variables for decoupling. Their minimal number and the corresponding transformation depend on the sparsity patterns of the Hessians of components of $\hat{r}$; we will not give the technical details here.

Bibliography

[1] D. Adams. *The Hitchhiker's Guide to the Galaxy*, volume 1 of *The Increasingly Inaccurately Named Hitchhiker's Guide to The Galaxy Trilogy*. Pocket Books, New York, 1981.

[2] Taifun Alishenas. *Zur numerischen Behandlung, Stabilisierung durch Projektion und Modellierung mechanischer Systeme mit Nebenbedingungen und Invarianten*. Ph. D. dissertation, Royal Institute of Technology, Stockholm, Sweden, Mar. 1992.

[3] American Institute of Aeronautics and Astronautics. *Proceedings of the 1989 AIAA Guidance, Navigation and Control Conference*, Boston, MA, Aug. 1989.

[4] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide*. SIAM publications, Philadelphia, PA, 1992.

[5] T. Andrzejewski, H. G. Bock, E. Eich, and R. von Schwerin. Recent advances in the numerical integration of multibody systems. In Schiehlen [75], pages 127–151.

[6] V. I. Arnold. *Mathematical Methods of Classical Mechanics*. Springer Verlag, New York, 1978.

[7] U. M. Ascher, J. Christiansen, and R. D. Russell. COLSYS—a collocation code for boundary value problems. *Springer Lecture Notes in Computer Science*, 76:164–185, 1979.

[8] U. M. Ascher, R. M. M. Mattheij, and R. D. Russell. *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*. Prentice-Hall, London, 1988.

[9] U. M. Ascher, S. Pruess, and R. D. Russell. On spline basis selection for solving differential equations. *SIAM Journal on Numerical Analysis*, 20:121–142, 1983.

[10] U. M. Ascher and R. J. Spiteri. Collocation software for boundary value differential-algebraic equations. *SIAM Journal on Scientific Computing*, 15(4):938–952, July 1994.

[11] G. Bader and U. M. Ascher. A new basis implementation for a mixed order boundary value ODE solver. *SIAM Journal on Scientific and Statistical Computing*, 8(4):483–500, July 1987.

[12] Viola Bär. Ein Kollokationsverfahren zur numerischen Lösung allgemeiner Mehrpunktrandwertaufgaben mit Schalt- und Sprungbedingungen mit Anwendungen in der optimalen Steuerung und der Parameteridentifizierung. Diploma thesis, Universität Bonn, Germany, 1983.

[13] R. E. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[14] J. T. Betts and W. P. Huffman. Path constrained trajectory optimization using sparse sequential quadratic programming. *AIAA Journal of Guidance, Control, and Dynamics*, 16(1):59–68, 1993.

[15] H. G. Bock. Zur numerischen Behandlung zustandsbeschränkter Steuerungsprobleme mit Mehrzielmethode und Homotopieverfahren. *Zeitschrift für Angewandte Mathematik und Mechanik*, 57:T266–T268, 1977.

[16] H. G. Bock. Recent advances in parameter identification techniques for ODE. In Deuflhard and Hairer, editors, *Numerical Treatment of Inverse Problems, Progress in Scientific Computing 2.* Birkhäuser, Boston, 1983.

[17] Hans Georg Bock. *Randwertproblemmethoden zur Parameteridentifizierung in Systemen nichtlinearer Differentialgleichungen.* Ph. D. dissertation, Bonner Mathematische Schriften 183, Universität Bonn, Germany, 1987.

[18] H. G. Bock and R. W. Longman. Optimal control of velocity profiles for minimization of energy consumption in the New York subway system. In *Proceedings of the 2nd IFAC Workshop on Control Applications of Nonlinear Programming and Optimization*, Oberpfaffenhofen, Germany, Sept. 1980.

[19] H. G. Bock and R. W. Longman. Computation of optimal controls on disjoint control sets for minimum energy subway operation. In *Proceedings of the American Astronomical Society. Symposium on Engineering Science and Mechanics*, Taiwan, 1982.

[20] H. G. Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of constrained optimal control problems. In *Proceedings of the 9th IFAC World Congress*, pages 242–247, Budapest, Hungary, July 1984. Pergamon Press.

[21] H. G. Bock, J. P. Schlöder, and V. H. Schulz. Numerik großer Differentiell-Algebraischer Gleichungen. Simulation und Optimierung. In H. Schuler, editor, *Prozeßsimulation*, chapter 2, pages 35–80. VCH Verlagsgesellschaft, Weinheim, Germany, 1994.

[22] H. G. Bock and R. von Schwerin. An inverse dynamics ADAMS-method for constrained multibody systems. Preprint 93-27, IWR, Universität Heidelberg, Germany, June 1993.

[23] H. G. Bock and R. von Schwerin. An ADAMS method for complex systems of nonlinear differential-algebraic equations with state dependent events. Preprint of SFB 359, IWR, Universität Heidelberg, Germany, 1995. In preparation.

[24] J. V. Breakwell, J. L. Speyer, and A. E. Bryson. Optimization and control of nonlinear systems using the second variation. *SIAM Journal on Control, Series A*, 1(2), 1963.

[25] A. E. Bryson, Jr. and Y.-C. Ho. *Applied Optimal Control*. Ginn & Company, Waltham, MA, 1969.

[26] R. Bulirsch. Die Mehrzielmethode zur numerischen Lösung von nichtlinearen Randwertproblemen und Aufgaben der optimalen Steuerung. Technical report, Carl-Cranz-Gesellschaft, Oberpfaffenhofen, Germany, 1971.

[27] R. Bulirsch, A. Miele, J. Stoer, and K. H. Well, editors. *Optimal Control: Calculus of Variations, Optimal Control Theory and Numerical Methods*, volume 111 of *International Series of Numerical Mathematics*. Birkhäuser, Basel, 1993.

[28] R. Bulirsch and D. Kraft, editors. *Computational Optimal Control*, volume 115 of *International Series of Numerical Mathematics*. Birkhäuser, Basel, 1994.

[29] C. de Boor and B. Swartz. Collocation at Gaussian points. *SIAM Journal on Numerical Analysis*, 10(4):582–606, Sept. 1973.

[30] P. Deuflhard. A modified Newton method for the solution of ill-conditioned systems of nonlinear equations with application to multiple shooting. *Numerische Mathematik*, 22:289–315, 1974.

[31] J. J. Dongarra, J. du Croz, I. S. Duff, and S. Hammarling. A set of level 3 basic linear algebra subprograms. *ACM Transactions on Mathematical Software*, 16:1–17, 1990.

[32] J. J. Dongarra, J. du Croz, S. Hammarling, and R. J. Hanson. An extended set of Fortran basic linear algebra subprograms. *ACM Transactions on Mathematical Software*, 14:1–17, 1988.

[33] I. S. Duff. MA28—a set of Fortran subroutines for sparse unsymmetric linear equations. Report AERE R8730, HMSO, London, 1977.

[34] J. Farkas. Über die Theorie der einfachen Ungleichungen. *Journal für die Reine und Angewandte Mathematik*, 124:1–27, 1901.

[35] A. V. Fiacco and G. P. McCormick. *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*. John Wiley & Sons, New York, 1968. Reprint: Volume 4 of *SIAM Classics in Applied Mathematics*, SIAM Publications, Philadelphia, PA, 1990.

[36] A. F. Filippov. On certain questions in the theory of optimal control. *SIAM Journal on Control, Series A*, 1:76–84, 1962.

[37] J. Frehse. Existence of optimal controls I. *Methods of Operations Research*, 30:26–59, 1979.

[38] J. Frehse. Ein Existenzsatz für optimale Steuerungen. Preprint 199, SFB 72, Universität Bonn, Germany, 1979.

[39] K. R. Frisch. The logarithmic potential method for convex programming. Memorandum, Institute of Economics, University of Oslo, Norway, May 1955.

[40] J. V. Gallitzendörfer and H. G. Bock. Parallel algorithms for optimization boundary value problems in DAE. Preprint 94-73, IWR, Universität Heidelberg, Germany, Dec. 1994.

[41] P. E. Gill, W. Murray, D. B. Ponceleón, and M. A. Saunders. Primal-dual methods for linear programming. Technical Report SOL 91-3, Dept. of Operations Research, Stanford University, May 1991.

[42] P. E. Gill, W. Murray, M. A. Saunders, and M. H. Wright. A Schur-complement method for sparse quadratic programming. Technical Report SOL 87-12, Dept. of Operations Research, Stanford University, 1987.

[43] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, London, 1981.

[44] C. C. Gonzaga. An interior trust region method for linearly constrained optimization. *COAL Newsletter*, 19:55–66, August 1991.

[45] S. P. Han. Superlinearly convergent variable-metric algorithms for general nonlinear programming problems. *Mathematical Programming*, 11:263–282, 1976.

[46] S. P. Han. A globally convergent method for nonlinear programming. *Journal of Optimization Theory and Applications*, 22:297–310, 1977.

[47] C. R. Hargraves and S. W. Paris. Direct trajectory optimization using nonlinear programming and collocation. *AIAA Journal of Guidance, Control, and Dynamics*, 10(4):338–342, 1987.

[48] M. K. Horn. Solution of the optimal control problem using the software package STOMP. In *Proceedings of the 8th IFAC Workshop on Control Applications of Nonlinear Programming and Optimization*, Paris, June 1989.

[49] R. Isaacs. *Differential Games*. John Wiley & Sons, New York, 1965.

[50] C. Jänsch, K. Schnepper, and K. H. Well. Ascent and descent trajectory optimization of Ariane V / Hermes. Report 197, DFG-Schwerpunktprogramm „Anwendungsbezogene Optimierung und Steuerung“, 1990.

[51] N. K. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[52] William Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, University of Chicago, 1939.

[53] H. B. Keller. *Numerical Methods for Two-Point Boundary Value Problems*. Blaisdell, London, 1968.

[54] W. Klingenberg. *Riemannian Geometry*, volume 1 of *Studies in Mathematics*. De Gruyter, Berlin, 1982.

[55] Jutta Konzelmann. Numerische Berechnung zeitoptimaler Steuerungen von Industrierobotern mit direkt optimierenden RWP-Methoden. Diploma thesis, Universität Bonn, Germany, 1988.

[56] J. Konzelmann, H. G. Bock, and R. W. Longman. Time optimal trajectories of elbow robots by direct methods. In AIAA [3], pages 883–894.

[57] J. Konzelmann, H. G. Bock, and R. W. Longman. Time optimal trajectories of polar robot manipulators by direct methods. *Modeling and Simulation*, 20(5):1933–1939, 1989.

[58] Peter Krämer-Eis. *Ein Mehrzielverfahren zur numerischen Berechnung optimaler Feedback-Steuerungen bei beschränkten nichtlinearen Steuerungsproblemen*. Ph. D. dissertation, Bonner Mathematische Schriften 166, Universität Bonn, Germany, 1985.

[59] D. Kraft. *TOMP—FORTRAN modules for optimal control calculations*. Fortschrittsberichte VDI, Reihe 8, Nr. 254. VDI-Verlag, Düsseldorf, Germany, 1991.

[60] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–493, Berkeley, 1951. University of California Press.

[61] Frank-Stephan Kupfer. *Reduced Successive Quadratic Programming in Hilbert Space with Applications to Optimal Control*. Ph. D. dissertation, Universität Trier, Germany, 1992.

[62] C. Lawson, R. Hanson, D. Kincaid, and F. Krough. Basic linear algebra subprograms for Fortran usage. *ACM Transactions on Mathematical Software*, 5:308–325, 1979.

[63] E. B. Lee and L. Markus. *Foundations of Optimal Control Theory*. John Wiley & Sons, New York, 1967.

[64] S. R. McReynolds. The successive sweep method and dynamic programming. *Journal of Mathematical Analysis and Applications*, 19:565–598, 1967.

[65] A. Miele. Recent advances in gradient algorithms for optimal control problems. *Journal of Optimization Theory and Applications*, 17, 1975.

[66] R. D. C. Monteiro and I. Adler. Interior path following primal-dual algorithms: Part II: Convex quadratic programming. *Mathematical Programming*, 44:43–66, 1989.

[67] D. D. Morrison, J. D. Riley, and J. F. Zancanaro. Multiple shooting method for two-point boundary value problems. *Communications of ACM*, 5:613–614, 1962.

[68] M. R. Osborne. On shooting methods for boundary value problems. *Journal of Mathematical Analysis and Applications*, 27:417–433, 1969.

[69] Karl-Josef Plitt. Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen. Diploma thesis, Universität Bonn, Germany, June 1981.

[70] L. S. Pontryagin, V. G. Boltyansky, R. V. Gamkrelidze, and E. F. Mischenko. *Mathematical Theory of Optimal Processes*. Wiley-Interscience, New York, 1962.

[71] M. J. D. Powell. A new algorithm for unconstrained optimization. In J. B. Rosen, O. L. Mangasarian, and K. Ritter, editors, *Nonlinear Programming*. Academic Press, New York, 1970.

[72] M. J. D. Powell. The convergence of variable metric methods for nonlinearly constrained optimization calculations. In O. L. Mangasarian, R. R. Meyer, and S. M. Robinson, editors, *Nonlinear Programming 3*, pages 27–63. Academic Press, New York, 1978.

[73] R. D. Russell and L. F. Shampine. A collocation method for boundary value problems. *Numerische Mathematik*, 19:1–28, 1972.

[74] G. Sachs, K. Lesch, H. G. Bock, and M. C. Steinbach. Periodic optimal trajectories with singular control for aircraft with high aerodynamic efficiency. In Bulirsch et al. [27], pages 289–304.

[75] W. Schiehlen, editor. *Advanced Multibody System Dynamics*, volume 20 of *Solid Mechanics and Its Applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1993.

[76] Volker Schulz. Ein effizientes Kollokationsverfahren zur numerischen Behandlung von Mehrpunktrandwertaufgaben in der Parameteridentifizierung und Optimalen Steuerung. Diploma thesis, Universität Augsburg, Germany, Oct. 1990.

[77] Volker Schulz. *Reduced SQP Methods for Large-Scale Optimal Control Problems in DAE with Application to Path Planning Problems for Satellite Mounted Robots.* Ph. D. dissertation, Universität Heidelberg, Germany, 1995. In preparation.

[78] V. H. Schulz, H. G. Bock, and R. W. Longman. Optimal path planning for satellite mounted robot manipulators. Preprint 93-13, IWR, Universität Heidelberg, Germany, Mar. 1993.

[79] V. H. Schulz, H. G. Bock, and R. W. Longman. Shortest paths for satellite mounted robot manipulators. In Bulirsch and Kraft [28], pages 357–366.

[80] V. H. Schulz, H. G. Bock, and M. C. Steinbach. Exploiting invariants in the numerical solution of multipoint boundary value problems for DAE. Preprint 93-69, IWR, Universität Heidelberg, Germany, Dec. 1993. Submitted to SIAM Journal on Scientific Computing.

[81] Marc Steinbach. Numerische Berechnung optimaler Steuerungen für Industrieroboter. Diploma thesis, Universität Bonn, Germany, 1987.

[82] M. C. Steinbach. A structured interior point SQP method for nonlinear optimal control problems. In Bulirsch and Kraft [28], pages 213–222.

[83] M. C. Steinbach, H. G. Bock, and R. W. Longman. Time-optimal extension or retraction in polar coordinate robots: A numerical analysis of the switching structure. In AIAA [3], pages 895–910.

[84] M. C. Steinbach, H. G. Bock, and R. W. Longman. Time optimal control of SCARA robots. In *Proceedings of the 1990 AIAA Guidance, Navigation and Control Conference*, pages 707–716, Portland, OR, Aug. 1990. American Institute of Aeronautics and Astronautics.

[85] M. C. Steinbach, H. G. Bock, and R. W. Longman. Time-optimal extension and retraction of robots: Numerical analysis of the switching structure. *Journal of Optimization Theory and Applications*, 1995. In press.

[86] J. Stoer. *Einführung in die Numerische Mathematik I*, volume 105 of *Heidelberger Taschenbücher*. Springer Verlag, Berlin, 1972.

[87] R. J. Vanderbei. Interior-point methods: Algorithms and formulations. Technical report, Princeton University, Aug. 1993.

[88] R. J. Vanderbei. Interior point methods for linear and quadratic programming. Lecture held at IWR, University of Heidelberg, Germany, 4 May 1994.

[89] R. J. Vanderbei and T. J. Carpenter. Symmetric indefinite systems for interior point methods. *Mathematical Programming*, 58:1–32, 1993.

[90] R. J. Vanderbei, A. Duarte, and B. Yang. An algorithmic and numerical comparison of several interior-point methods. Technical Report SOR-94-05, Princeton University, Mar. 1993.

[91] R. von Schwerin and M. Winckler. MBSSIM—a guide to an integrator library for multibody systems simulation, version 1.00. Technical Report 94-75, IWR, Universität Heidelberg, Germany, Dec. 1994.

[92] O. von Stryk. Numerical solution of optimal control problems by direct collocation. In Bulirsch et al. [27], pages 129–143.

[93] R. Weiss. The application of implicit Runge-Kutta and collocation methods to boundary-value problems. *Mathematics of Computation*, 28(126):449–464, 1974.

[94] R. B. Wilson. *A Simplicial Algorithm for Concave Programming*. PhD thesis, Harvard University, 1963.

[95] Y. Ye and E. Tse. An extension of Karmarkar's projective algorithm for convex quadratic programming. *Mathematical Programming*, 44:157–179, 1989.