BLOCK STRUCTURED QUADRATIC PROGRAMMING FOR THE DIRECT MULTIPLE SHOOTING METHOD FOR OPTIMAL CONTROL

CHRISTIAN KIRCHES[†*], HANS GEORG BOCK[†], JOHANNES P. SCHLÖDER[†], SEBASTIAN SAGER[†]
[†]INTERDISCIPLINARY CENTER FOR SCIENTIFIC COMPUTING (IWR), RUPRECHT–KARLS–UNIVERSITY OF HEIDELBERG, IM NEUENHEIMER FELD 368, 69120 HEIDELBERG, GERMANY.

Abstract. In this contribution we address the efficient solution of optimal control problems of dynamic processes with many controls. Such problems arise, e.g., from the outer convexification of integer control decisions. We treat this optimal control problem class using the direct multiple shooting method to discretize the optimal control problem. The resulting nonlinear problems are solved using sequential quadratic programming methods. We review the classical condensing algorithm that preprocesses the large but sparse quadratic programs to obtain small but dense ones. We show that this approach leaves room for improvement when applied in conjunction with outer convexification. To this end, we present a new complementary condensing algorithm for quadratic programs with many controls. This algorithm is based on a hybrid null–space range–space approach to exploit the block sparse structure of the quadratic programs that is due to direct multiple shooting. An assessment of the theoretical run time complexity reveals significant advantages of the proposed algorithm. We give a detailed account on the required number of floating point operations, depending on the process dimensions. Finally we demonstrate the merit of the new complementary condensing approach by comparing the behavior of both methods for a vehicle control problem in which the integer gear decision is convexified.

1. Introduction

Mixed–integer optimal control problems (MIOCPs) in ordinary differential equations (ODEs) have a high potential for optimization. A typical example is the choice of gears in transport [14, 20, 23, 32, 36].

Direct methods, in particular *all–at–once approaches*, [5, 7, 26], have become the methods of choice for most practical optimal control problems. The drawback of direct methods with binary control functions obviously is that they lead to high–dimensional vectors of binary variables. Because of the exponentially growing complexity of the problem, techniques from mixed–integer nonlinear programming will work only for small instances [37]. In past contributions [23, 29, 30, 33] we proposed to use an *outer convexification* with respect to the binary controls, which has several main advantages over standard formulations or convexifications, cf. [29, 30]. A number of challenging mixed–integer optimal control problems has already been solved with this approach, cf. [23, 33].

* E-mail: christian.kirches@iwr.uni-heidelberg.de.

From the direct approach discretization of the MIOCP that is applied after outer convexification of the integer control, a highly structured Nonlinear Program (NLP) is obtained. For its solution, both Sequential Quadratic Programming (SQP) methods [5, 7, 26] and Interior Point (IP) methods [39] have become popular. In this contribution we consider SQP methods exclusively, which solve a sequence of Quadratic Programs (QPs) to obtain the NLP's solution and thus those of the discretized OCP. Here, the outer convexification approach results in QPs with *many control parameters*, one per possible discrete choice per discretization point in time.

Concerning the efficient solution of these QPs, active set methods are favored over interior point methods in an SQP context. This is due to the better performance of active set QP methods on a sequence of closely related QPs [4, 11]. Efficient exploitation of the problem structure found in the QP data is crucial for the efficiency of the QP solving procedure. Two possible approaches are thinkable here. First, in a preprocessing step the QP may be subjected to a reformulation that is tailored to the structures introduced by the discretization method. The classical *condensing* algorithm [26, 7] is reviewed here. It is shown to leave room for improvement if the QP has more control parameters than system states. Second, an active set QP code that directly exploits the block structure may be designed. This still is an active field of research, cf. [3, 21], where the difficulties lie with the efficient factorization of the QP's structured KKT system. We present a new approach at solving QPs with block structure due to *direct multiple shooting*, named *complementary condensing*, based on work by [35]. This approach was first introduced in [22] and provides a factorization of the QP's KKT system tailored to the direct multiple shooting block structure. In this contribution, an evaluation of an ANSI C implementation of this approach and a comparison to classical condensing are presented for the first time. A detailed analysis of the required number of floating point operations is made, depending on the process dimensions.

We finally apply the new complementary condensing approach for the first time to a vehicle control problem due to [13, 14] in which the integer gear decision is convexified as first proposed for this problem in [23]. The obtained run times are compared to a general–purpose sparse symmetric indefinite factorization of the QP's KKT system using the HSL code MA57, as well as to the performance of the dense active set QP code QPOPT solving the condensed QPs obtained from the classical condensing algorithm.

1.1. **Structure of the Paper.** In section 1 we describe an exemplary vehicle control problem with gear shift. The integer gear choice is treated by outer convexification, which is briefly mentioned. Section 2 describes *direct multiple shooting* as our method of choice for discretizing the OCP and presents the structure of the block–sparse NLP. We briefly mention SQP methods and motivate the source of the QPs with block sparse structure and many control parameters, to be dealt with in the two following sections. Section 3 reviews the classical condensing algorithm that reduces the large and block–sparse QP to a small but dense one. This condensed QP may be solved with any available QP code. This first approach at QP solving is applied to the example problem, and the resulting run times are discussed. Section 4 presents a new alternative approach at solving the QPs that exploits the block sparse structure inside the QP solver. Application of this new method to the example problem yields improved run times, as will be discussed. Section 5 concludes

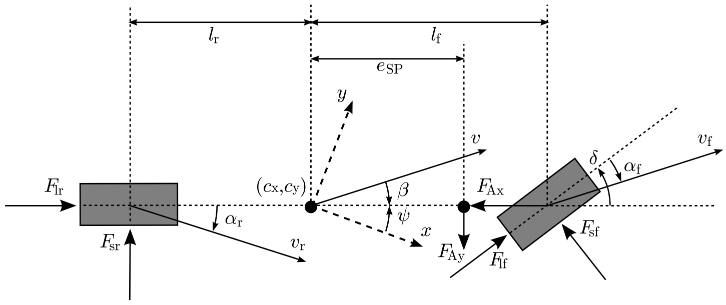


FIGURE 1. Coordinates, angles, and forces in the single–track car model used as a test bed.

State	Unit	Description
c_x	m	Horizontal position of the car
c_y	m	Vertical position of the car
v	$\frac{m}{s}$	Magnitude of directional velocity of the car
δ	rad	Steering wheel angle
β	rad	Side slip angle
ψ	rad	Yaw angle
w_z	$\frac{rad}{s}$	Yaw angle velocity

TABLE 1. Coordinates and states used in the single–track car model.

this contribution and provides an outlook on further developments in block sparse active set QP solving.

1.2. **Example: A Vehicle Control Problem with Gear Shift.** In this section we review a vehicle control problem that is due to [13, 14] as a test bed for both presented approaches to solving the block sparse quadratic problems within a direct multiple shooting method for optimal control.

1.2.1. *Vehicle Model.* We consider a single–track model of a vehicle as depicted in figure 1 whose dynamics are modeled by a system of ordinary differential equations (ODEs) with 7 states as briefly listed in table 1. As this is a time optimal problem, an additional differential state representing the current time t is introduced for the transformation from a fixed time horizon $\tau \in [0, 1]$ to the one of variable length $t \in [0, t_f]$. The length t_f of the time horizon is a global model parameter subject to optimization, which for simplicity of the implementation is introduced as a constant differential state as well. Finally, together with the objective function of Lagrangian type, the problem has a total of $n^x = 10$ differential states.

The driver, in our case the optimal control problem solver, exercises control over the steering wheel, the pedal, the brakes, and the choice of the gear, as listed in table 2. A more extensive description of this optimal control problem, its differential equations, model parameters, objective function, and constraints can be found in [23] together with optimal solutions and computation times for a test driving scenario.

1.2.2. *Outer Convexification of the Integer Control.* We treat the integer gear choice $\mu(t) \in \{1, \dots, n^\mu\}$, wherein n^μ denotes the number of available gears, by outer

Control	Unit	Description
w_δ	$\frac{\text{rad}}{\text{s}}$	Steering wheel angular velocity
F_{B}	N	Total braking force
ϕ	–	Accelerator pedal position
μ	–	Selected gear

TABLE 2. Controls used in the single–track car model.

convexification as detailed in [32, 23]. Reasonable choices for n^μ range from 4 up to 24 in heavy–duty trucks, cf. [20, 36]. Outer convexification basically amounts to replacing the right hand side $f(\cdot)$ of the car model's ODE system

$$\dot{x}(t) = f(t, x(t), u(t), \mu(t)) \tag{1}$$

wherein $x = (c_{\text{x}}, c_{\text{y}}, v, \delta, \beta, \psi, w_{\text{z}})$ and $u = (w_\delta, F_{\text{B}}, \phi)$, by its outer convexified reformulation

$$\dot{x}(t) = \sum_{i=1}^{n^\mu} w_i(t) \cdot f(t, x(t), u(t), \mu_i), \quad \sum_{i=1}^{n^\mu} w_i(t) = 1, \qquad \forall\ t \in \mathcal{T}. \tag{2}$$

For each element $\mu_i \in \{1, \ldots, n^\mu\}$ a separate binary control $w_i(\cdot) \in \{0, 1\}$ is introduced, subject to the Special Ordered Set 1 (SOS1) constraint ensuring that for all $t \in [t_0, t_{\text{f}}]$ exactly one of the choices is attained. The same is done for every constraint function that involves $\mu(\cdot)$. The total number of control parameters for this car model then is $n^{\text{q}} = 3 + n^\mu$. Note that this formulation is still equivalent to the original one. The optimal control problem is solved with relaxed controls $w_i(t) \in [0, 1] \subset \mathbb{R}$, making the $w_i(t)$ convex multipliers. We refer to [23, 29, 32] for a discussion of the favorable properties of the obtained relaxed solution as well as a detailed presentation of possibilities to reconstruct an integer solution from the relaxed one.

2. Direct Multiple Shooting for Optimal Control

In this section we describe the direct multiple shooting method due to [26, 7] as an efficient tool for the discretization and parameterization of a general class of infinite dimensional optimal control problems (OCP). Using this method, we obtain from the OCP a highly structured NLP which we solve with an SQP method. Condensing, a preprocessing step to exploit the block structure of the discretized problem, is presented in section 3, and our proposition of a new complementary condensing approach is found in section 4.

2.1. **Optimal Control Problem Formulation.** We consider the following general class (3) of optimal control problems

$$\min_{x(\cdot), u(\cdot)} \quad l(x(\cdot), u(\cdot)) \tag{3a}$$

$$\text{s.t.} \quad \dot{x}(t) = f(t, x(t), u(t)) \qquad \forall t \in \mathcal{T} \tag{3b}$$

$$0 \le c(t, x(t), u(t)) \qquad \forall t \in \mathcal{T} \tag{3c}$$

$$0 = r_i^{\text{eq}}(t_i, x(t_i)) \qquad 0 \le i \le m \tag{3d}$$

$$0 \le r_i^{\text{in}}(t_i, x(t_i)) \qquad 0 \le i \le m \tag{3e}$$

in which we strive to minimize objective function $l(\cdot)$ depending on the trajectory $x(\cdot)$ of a dynamic process described in terms of a system $f : \mathcal{T} \times \mathbb{R}^{n^\mathrm{x}} \times \mathbb{R}^{n^\mathrm{u}} \to \mathbb{R}^{n^\mathrm{x}}$ of ordinary differential equations (ODE), running on a time horizon $\mathcal{T} := [t_0, t_\mathrm{f}] \subset \mathbb{R}$, and governed by a control trajectory $u(\cdot)$ subject to optimization. The process trajectory $x(\cdot)$ and the control trajectory $u(\cdot)$ shall satisfy certain inequality path constraints $c : \mathcal{T} \times \mathbb{R}^{n^\mathrm{x}} \times \mathbb{R}^{n^\mathrm{u}} \to \mathbb{R}^{n^{\mathrm{c,in}}}$ on the time horizon $\mathcal{T}$, as well as equality and inequality point constraints $r_i^{\mathrm{eq}} : \mathcal{T} \times \mathbb{R}^{n^\mathrm{x}} \to \mathbb{R}^{n_i^{\mathrm{r,eq}}}$ and $r_i^{\mathrm{in}} : \mathcal{T} \times \mathbb{R}^{n^\mathrm{x}} \to \mathbb{R}^{n_i^{\mathrm{r,in}}}$ on a prescribed grid on $\mathcal{T}$ consisting of $m+1$ grid points

$$t_0 < t_1 < \ldots < t_{m-1} < t_m := t_\mathrm{f}, \qquad m \in \mathbb{N},\ m \geq 1. \tag{4}$$

In order to make this infinite dimensional optimal control problem computationally accessible, the direct multiple shooting method is applied to discretize the control trajectory $u(\cdot)$ subject to optimization.

2.2. **Control Discretization.** We introduce a discretization of the control trajectory $u(\cdot)$ by defining a *shooting grid*

$$t_0 < t_1 < \ldots < t_{m-1} < t_m := t_\mathrm{f}, \qquad m \in \mathbb{N},\ m \geq 1. \tag{5}$$

that shall be a superset of the constraint grid used in (3). For clarity, we assume in the following that the two grids coincide, though this is not a theoretical or algorithmic requirement. On each interval of the *shooting grid* (5) we introduce a vector $q_i \in \mathbb{R}^{n_i^\mathrm{q}}$ of *control parameters* together with an associated *control base function* $b_i : \mathcal{T} \times \mathbb{R}^{n_i^\mathrm{q}} \to \mathbb{R}^{n^\mathrm{u}}$,

$$u(t) := \sum_{j=1}^{n_i^\mathrm{q}} b_{ij}(t, q_{ij}), \qquad t \in [t_i, t_{i+1}] \subseteq \mathcal{T},\ 0 \leq i \leq m-1. \tag{6}$$

Popular examples of control discretizations are piecewise constant, piecewise linear, or piecewise cubic functions. The number and location of the shooting grid points obviously affects the quality of the optimal solution of the discretized problem approximating the solution of (3).

2.3. **State Parameterization.** In addition to the control parameter vectors, we introduce state vectors $s_i \in \mathbb{R}^{n^\mathrm{x}}$ in all shooting nodes serving as initial values for m IVPs

$$\dot{x}_i(t) = f(t, x_i(t), q_i), \qquad t \in [t_i, t_{i+1}] \subseteq \mathcal{T},\ 0 \leq i \leq m-1, \tag{7a}$$

$$x_i(t_i) = s_i, \qquad 0 \leq i \leq m-1. \tag{7b}$$

This parameterization of the process trajectory $x(\cdot)$ will in general be discontinuous on $\mathcal{T}$. Continuity of the solution is ensured by introduction of additional *matching conditions*

$$x_i(t_{i+1};\ t_i, s_i, q_i) - s_{i+1} = 0, \qquad 0 \leq i \leq m-1, \tag{8}$$

where $x_i(t_{i+1};\ t_i, s_i, q_i)$ denotes the evaluation of the state trajectory $x_i(\cdot)$ at the final time t_{i+1} of shooting interval i, and depending on the start time t_i, initial value s_i, and control parameters q_i on that interval.

2.4. **Constraint Discretization.** The path constraints of problem (3) are enforced on the nodes of the shooting grid (5) only. While in general it can be observed that this formulation already leads to a solution that satisfies the path constrains on the whole of $\mathcal{T}$, methods from semi–infinite programming exist [27] to ensure this in a rigorous fashion. For clarity we define the combined equality–inequality constraint functions $r_i : \mathcal{T} \times \mathbb{R}^{n^\mathrm{x}} \times \mathbb{R}^{n^\mathrm{u}} \to \mathbb{R}^{n_i^\mathrm{r}}$ where $n_i^\mathrm{r} := n^\mathrm{c} + n_i^{\mathrm{r,eq}} + n_i^{\mathrm{r,in}}$ that comprise all discretized inequality path constraints as well as equality and inequality point constraints,

$$0 \leqq r_i(t_i, s_i, q_i), \qquad 0 \leq i \leq m-1, \tag{9a}$$

$$0 \leqq r_m(t_m, s_m). \tag{9b}$$

2.5. **Separable Objective.** The objective function $l(x(\cdot), u(\cdot))$ shall be separable with respect to the shooting grid structure, i.e.

$$l(x(\cdot), u(\cdot)) = \sum_{i=0}^{m} l_i(x_i(\cdot), q_i). \tag{10}$$

In general, $l(\cdot)$ will be a Mayer type function evaluated at the end of the horizon $\mathcal{T}$, or Lagrange type integral objective evaluated on the whole of $\mathcal{T}$. For both types, a formulation that is separable with respect to the shooting grid structure is easily found,

$$l(x(\cdot), u(\cdot)) = M(s_m), \tag{11}$$

$$l(x(\cdot), u(\cdot)) = \sum_{i=0}^{m-1} \int_{t_i}^{t_{i+1}} L_i(x_i(t), q_i) \, \mathrm{d}t, \tag{12}$$

where in (12) the integral actually is a function of (s_i, q_i). We may thus assume the objective function to be of the general structure

$$l(x(\cdot), u(\cdot)) = \sum_{i=0}^{m} l_i(s_i, q_i). \tag{13}$$

2.6. **Multiple Shooting Discretized Optimal Control Problem.** Summarizing, the discretized multiple shooting optimal control problem can be cast as a nonlinear problem

$$\min_{w} \quad \sum_{i=0}^{m} l_i(w_i) \tag{14a}$$

$$\text{s.t.} \quad 0 = x_i(t_{i+1};\ t_i, w_i) - s_{i+1}, \qquad 0 \leq i \leq m-1 \tag{14b}$$

$$0 \leqq r_i(w_i), \qquad 0 \leq i \leq m \tag{14c}$$

with the vector of unknowns w being

(15a)

$$w := (s_1, q_1, \dots, s_{m-1}, q_{m-1}, s_m), \qquad w_i := (s_i, q_i), \ 0 \leq i \leq m-1, \qquad w_m := s_m,$$

where the evaluation of the matching condition constraint (14b) requires the solution of an initial value problem (7).

2.7. **Sparse Quadratic Subproblem.** Sequential Quadratic Programming (SQP) methods are a long–standing and highly effective method for the solution of NLPs that also allow for much flexibility in exploiting the problem's special structure. First introduced by [19, 28], SQP methods iteratively progress towards a KKT point of the NLP by solving a linearly constrained local quadratic model of the NLP's Lagrangian [25].For the NLP (14) arising from direct multiple shooting this local quadratic model reads

$$\min_{\delta w} \quad \tfrac{1}{2}\sum_{i=0}^{m} \delta w_i' B_i \delta w_i + g_i' \delta w_i \tag{16a}$$

$$\text{s.t.} \quad 0 = X_i(w_i)\delta w_i - \delta s_{i+1} - h_i(w_i), \qquad 0 \le i \le m-1, \tag{16b}$$

$$0 \leqq R_i(w_i)\delta w_i - r_i(w_i), \qquad 0 \le i \le m, \tag{16c}$$

with the following notations for vector of unknowns δw and its components

$$\delta w := (\delta s_1, \delta q_1, \ldots, \delta s_{m-1}, \delta q_{m-1}, \delta s_m)\,, \tag{17a}$$

$$\delta w_i := (\delta s_i, \delta q_i)\,,\ 0 \le i \le m-1, \qquad \delta w_m := \delta s_m, \tag{17b}$$

similar to the notation used in (15a), and with vectors h_i denoting the matching conditions residuals,

$$h_i(w_i) := x_i(t_{i+1};\ t_i, w_i) - s_{i+1}. \tag{18}$$

The matrices B_i denote the node Hessians (or a suitable approximations, cf. [7]) of the NLP's Lagrangian, while the vectors g_i denotes the node gradients of the NLP's objective function. Matrices X_i, R_i^{eq}, and R_i^{in} denote linearizations of the constraint functions obtained in w_i,

$$B_i \approx \frac{\mathrm{d}^2 l_i(w_i)}{\mathrm{d}w_i^2}, \qquad g_i := \frac{\mathrm{d}l_i(w_i)}{\mathrm{d}w_i}, \tag{19a}$$

$$R_i := \frac{\mathrm{d}r_i(w_i)}{\mathrm{d}w_i}, \qquad X_i := \frac{\partial x_i(t_{i+1};\ t_i, w_i)}{\partial w_i}. \tag{19b}$$

In particular, the computation of the *sensitivity matrices* X_i requires the computation of derivatives of the solution of IVP (7) with respect to the initial values w_i. To ensure consistency of the derivatives, this should be done according to the principle of *internal numerical differentiation* (IND) [1, 6], i.e. by computing nominal solution and its derivatives using the same discretization scheme.

3. Condensing to obtain a Dense Quadratic Problem

In order to solve the QP (16) efficiently, one has to take advantage of its block structure that is due to multiple shooting. In view of the widespread availability and reliable performance of active–set QP codes, an obvious choice is to employ one of these solvers for that purpose. System (16) does not suit the majority of codes, though. They either do not exploit sparsity in the QP data, i.e. they are dense solvers [17, 34], or do exploit sparsity at a general–purpose level by employing linear algebra working on specially shaped dense data [3, 17], where the shape assumptions are not fulfilled by QP (16). Generic sparse data in triplets or column–compressed format is commonly accepted by interior–point QP solvers only, cf. [38, 15], which are not ideally suited for employment inside an SQP method. Only recently, some progress towards a general–purpose sparse active set solver has been made as presented in [21].

The block sparse structure in QP (16) therefore is exploited in a preprocessing or *condensing* step that transforms the QP into a related, considerably smaller, and densely populated one. In this section we briefly review this condensing algorithm due to [26, 7] and presented to great detail in [24], and give an account of the runtime complexity of the various steps.

3.1. **Reordering the Sparse Quadratic Problem.** We start by reordering the constraint matrix of QP (16) to separate the additionally introduced node values $\delta v = (\delta s_1, \ldots, \delta s_m)$ from the single shooting values $\delta u = (\delta s_0, \delta q_0, \ldots, \delta q_{m-1})$ as shown below,

$$\left(\begin{array}{cccc|cccc} X_0^{\mathrm{s}} & X_0^{\mathrm{q}} & & & -I & & & \\ & & X_1^{\mathrm{q}} & & X_1^{\mathrm{s}} & -I & & \\ & & & \ddots & & \ddots & \ddots & \\ & & & X_{m-1}^{\mathrm{q}} & & & X_{m-1}^{\mathrm{s}} & -I \\ \hline R_0^{\mathrm{s}} & R_0^{\mathrm{q}} & & & & & & \\ & & R_1^{\mathrm{q}} & & R_1^{\mathrm{s}} & & & \\ & & & \ddots & & \ddots & & \\ & & & R_{m-1}^{\mathrm{q}} & & & R_{m-1}^{\mathrm{s}} & \\ & & & & & & & R_m^{\mathrm{s}} \end{array}\right). \tag{20}$$

3.2. **Elimination Using the Matching Conditions.** We may now use the negative identity matrix blocks of the equality matching conditions as pivots to formally eliminate the additionally introduced multiple shooting state values $(\delta s_1, \ldots, \delta s_m)$ from system (20), analogous to the usual Gaussian elimination method for triangular matrices. This elimination procedure was introduced in [7] and a detailed presentation can be found in [24]. From this elimination procedure the dense constraint matrix

$$\begin{pmatrix} \overline{X} & -I \\ \overline{R} & 0 \end{pmatrix} = \left(\begin{array}{ccccc|cccc} X_0^{\mathrm{s}} & X_0^{\mathrm{q}} & & & & -I & & & \\ X_1^{\mathrm{s}} X_0^{\mathrm{s}} & X_1^{\mathrm{s}} X_0^{\mathrm{q}} & X_1^{\mathrm{q}} & & & & -I & & \\ \vdots & \vdots & \vdots & \ddots & & & & \ddots & \\ \Pi_0^{m-1} & \Pi_1^{m-1} X_0^{\mathrm{q}} & \Pi_2^{m-1} X_1^{\mathrm{q}} & \cdots & X_{m-1}^{\mathrm{q}} & & & & -I \\ \hline R_0^{\mathrm{s}} & R_0^{\mathrm{q}} & & & & & & & \\ R_1^{\mathrm{s}} X_0^{\mathrm{s}} & R_1^{\mathrm{s}} X_0^{\mathrm{q}} & R_1^{\mathrm{q}} & & & & & & \\ \vdots & \vdots & \vdots & \ddots & & & & & \\ R_m^{\mathrm{s}} \Pi_0^{m-1} & R_m^{\mathrm{s}} \Pi_1^{m-1} X_0^{\mathrm{q}} & R_m^{\mathrm{s}} \Pi_2^{m-1} X_1^{\mathrm{q}} & \cdots & R_m^{\mathrm{s}} X_{m-1}^{\mathrm{q}} & & & & \end{array}\right). \tag{21}$$

is obtained, with sensitivity matrix products Π_j^k defined to be

$$\Pi_j^k := \prod_{l=j}^{k} X_l^{\mathrm{s}},\ 0 \le j \le k \le m-1, \quad \Pi_j^k := I,\ j > k. \tag{22}$$

From (21) we deduce that, after this elimination step, the transformed QP in terms of the two unknowns δu and δv reads

$$\min_{\delta u, \delta v} \quad \frac{1}{2}\begin{pmatrix} \delta u \\ \delta v \end{pmatrix}' \overbrace{\begin{pmatrix} \overline{B}_{11} & \overline{B}_{12} \\ \overline{B}_{12}' & \overline{B}_{22} \end{pmatrix}}^{=B} \begin{pmatrix} \delta u \\ \delta v \end{pmatrix} + \overbrace{\begin{pmatrix} \overline{g}_1 \\ \overline{g}_2 \end{pmatrix}'}^{=g'} \begin{pmatrix} \delta u \\ \delta v \end{pmatrix} \tag{23a}$$

$$\text{s.t.} \quad 0 = \overline{X}\delta u - I\delta v - \overline{h} \tag{23b}$$

$$0 \leqq \overline{R}\delta u - \overline{r} \tag{23c}$$

with appropriate right hand side vectors $\overline{h}$ and $\overline{r}$ obtained by applying the Gaussian elimination steps to h and r.

3.3. **Reduction to a Single Shooting Sized System.** System (23) easily lends itself to the elimination of the unknown δv. By this step we arrive at the final *condensed QP*

$$\min_{\delta u} \quad \tfrac{1}{2}\delta u' \overline{\overline{B}} \delta u + \overline{\overline{g}}' \delta u \tag{24a}$$

$$\text{s.t.} \quad 0 \leq \overline{R}\delta u - \overline{r} \tag{24b}$$

with the following dense Hessian matrix and gradient obtained from substitution of δv in the objective (23a)

$$\overline{\overline{B}} = \overline{B}_{11} + \overline{B}_{12}\overline{X} + \overline{X}'\overline{B}_{12}' + \overline{X}'\overline{B}_{22}\overline{X}, \tag{25a}$$

$$\overline{\overline{g}} = \overline{g}_1 + \overline{X}'\overline{g}_2 - \overline{B}_{12}'\overline{h} - \overline{X}'\overline{B}_{22}\overline{h} \tag{25b}$$

The matrix multiplications required for the computation of these values are easily laid out to exploit the block sparse structure of $\overline{X}$ and B. In addition, from the elimination steps of sections 3.2 and 3.3 one obtains relations that allow to recover $\delta v = (\delta s_1, \dots, \delta s_m)$ from the solution $\delta u = (\delta s_0, \delta q_0, \dots, \delta q_{m-1})$ of the condensed QP (24).

3.4. **Solving the Condensed Quadratic Problem.** As the resulting condensed QP (24) no longer has a multiple shooting specific structure, it may be solved using any standard dense method for quadratic programming, which is what condensing aims for. Popular codes are the null space method QPSOL, available as subroutine E04NAF in the NAG library, and its successor QPOPT [17], available as subroutine E04NFF. An efficient code for parametric quadratic programming is qpOASES [11]. Further active set codes such as the Schur complement code QPSchur [3] and the QPKWIK code [34] exist. The primal–dual null–space solver BQPD [12] is also able to exploit sparsity remaining in the condensed QP to a certain extent. An extensive bibliography of existing QP methods and codes can be found in [18].

3.5. **Condensing and Dense QP Solving for the Example Problem.** We briefly introduced condensing as the established way of solving the sparse QP (16). In this section we examine the computational complexity of the condensing algorithm, as well as that of a dense null–space active set method working on the condensed QP (24). The application of this classical algorithm to the exemplary vehicle control problem presented in the introductory section reveals some shortcomings of the condensing algorithm for OCPs with many controls, e.g. due to outer convexification of integer controls.

Action	Run time complexity
Computing the Hessian $\overline{\overline{B}}$	$O(m^2 n^{\mathrm{x}3}) + O(m^2 n^{\mathrm{x}2} n^{\mathrm{q}})$
Computing the Constraints $\overline{X}$, $\overline{R}$	$O(m^2 n^{\mathrm{x}3}) + O(m^2 n^{\mathrm{x}2} n^{\mathrm{q}})$
Dense QP solver on (24), startup	$O((mn^{\mathrm{q}} + n^{\mathrm{x}})^3)$
Dense QP solver on (24), per iteration	$O((mn^{\mathrm{q}} + n^{\mathrm{x}})^2)$
Recovering δv	$O(mn^{\mathrm{x}2})$

TABLE 3. Run time complexity of the condensing algorithm and a dense active–set null–space QP solver, given in terms of the optimal control problem dimensions. The symbol m denotes the shooting grid length, while $n = n^{\mathrm{x}} + n^{\mathrm{q}}$ is the total number of unknowns per shooting node.

			Block sparse		Condensed		Dense QP solver
m	n^μ	Matrix	Size	nnz	Size	nnz	nnz seen
20	4	Hess.	330×330	5,136	130×130	16,900	16,900 (3.3×)
		Constr.	264×330	2,005	64×130	3,116	8,320 (4.1×)
50	4	Hess.	810×810	12,816	310×310	96,100	96,100 (7.5×)
		Constr.	654×810	4,756	154×310	16,767	47,740 (10.0×)
20	16	Hess.	570×570	15,624	370×370	136,900	136,900 (8.8×)
		Constr.	264×570	3,585	64×370	8,876	23,680 (6.6×)
50	16	Hess.	1410×1410	39,144	910×910	828,100	828,100 (21.2×)
		Constr.	654×1410	8,956	154×910	49,167	140,140 (15.6×)

TABLE 4. Comparison of dimensions and number of nonzero elements (nnz) of the Hessian and constraints matrix of QPs (16) and (24) for the exemplary vehicle control problem. All numbers for $n^{\mathrm{x}} = 10$, $n^{\mathrm{q}} = 3 + n^\mu$, m and n^μ varied. The last column gives the number of nonzero elements seen by the dense QP solver. In parentheses the increase compared to the number of nonzero elements in the block sparse QP is given.

Clearly from table 3 it can already be deduced that the classical condensing algorithm will be suitable for problems with limited grid lengths m and with considerably more states than controls, i.e. $n^{\mathrm{q}} \ll n^{\mathrm{x}}$, which is exactly contrary to the situation encountered for MIOCPs. Nonetheless, using this approach we could solve several challenging mixed–integer optimal control problems to optimality with little computational effort, as reported in [23, 29, 31].

In table 4 the dimensions and amount of sparsity present in the Hessian and constraints matrices are given for the exemplary vehicle control problem for $n^\mu = 4$ and $n^\mu = 16$ available gears. Here, the shooting grid lengths of $m = 20$ and $m = 50$ intervals were examined. As can be seen in the left part of the table, the block structured QP is only sparsely populated with the number of nonzero matrix entries never exceeding 3 percent. After the condensing step, the sparsity of both Hessian and constraints has been lost almost completely, as expected. This would be of no concern if the overall dimension of the QP had reduced considerably, as is the case for optimal control problems with $n^{\mathrm{x}} \gg n^{\mathrm{q}}$. For the case of an outer

	$m = 20$	$m = 30$	$m = 40$	$m = 50$
$n^\mu = 4$	4	12	25	45
$n^\mu = 8$	7	20	44	81
$n^\mu = 12$	11	31	68	126
$n^\mu = 16$	15	43	97	183

Table 5. Run times in milliseconds of the classical condensing algorithm of section 3 for the presented vehicle control problem with increasing number of shooting nodes m and number of gears n^μ.

	$m = 20$	$m = 30$	$m = 40$	$m = 50$
$n^\mu = 4$	0.3	0.9	2.0	3.7
$n^\mu = 8$	0.6	1.6	4.7	8.7
$n^\mu = 12$	1.2	2.9	7.6	11.1
$n^\mu = 16$	2.2	3.9	13.2	20.7

Table 6. Average run times in milliseconds per iteration of the dense null–space active–set QP solver QPOPT running on the condensed QPs for the presented vehicle control problem with increasing number of shooting nodes m and number of gears n^μ.

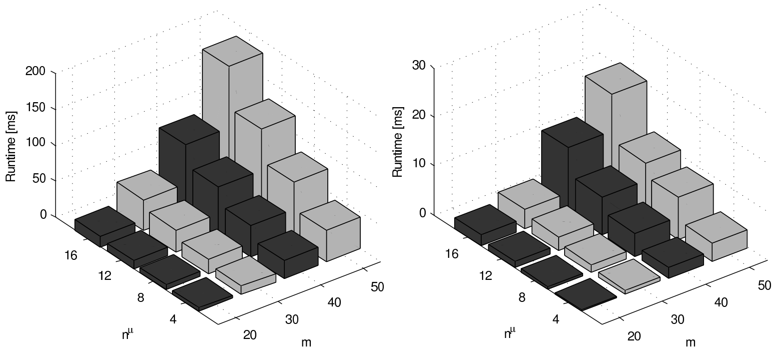
convexified MIOCP, however, this is not achieved. Worse yet, the dense active set method is unable to exploit what sparsity remains in the condensed constraints matrix, impairing the QP solver's performance further.

The results shown in table 4 indicate that for larger values of m or n^μ, a considerable increase of the run time is to be expected. The matrices' size has been reduced only marginally, while the number of matrix entries treated by the dense QP solver has, for the largest instance examined, risen by a more than a factor of 15 when compared to the block sparse QP.

This concern is supported by the results shown in tables 5 and 6. Here we list the run times in milliseconds of the classical condensing algorithm and of a single iteration of the dense null–space active–set QP solver QPOPT [17]. Averages have been taken over the all SQP iterations required to solve the optimal control problem to a precision of 10^{-6}. All run times have been obtained for an ANSI C99 (direct multiple shooting, condensing) and Fortran 77 (QPOPT) implementation running under Ubuntu Linux 9.04 on *a single core* of an Intel Core i7 920 machine at 2.67 GHz. BLAS linear algebra operations were done by ATLAS [40] in all parts of the implementation.

While the condensing algorithm's quadratic run time growth with the number m of multiple shooting nodes is acceptable for small systems, it becomes very noticeable for a larger number of integer decisions n^μ. The cubic complexity of the dense QP solver's initial setup with respect to m is clearly visible. The run time per iteration grows quadratically with both dimensions. When many active set iterations are required to find the QP's solution, this quickly becomes the bottleneck of the entire optimal control problem solution process as m or n^μ grow.

Summarizing the results presented in this section, we have seen that for optimal control problems with larger dimension n^q of the control parameters vector,



(a) Classical condensing runtime

(b) Average run time per iteration of the dense active–set QP solver.

Figure 2. Average run times in milliseconds for the presented vehicle control problem when solved using the dense null–space active–set QP solver QPOPT running on the condensed QPs obtained from the classical condensing algorithm of section 3.

the condensing algorithm is unable to significantly reduce the QPs dimension. After condensing, the QP's Hessian and constraints matrix nonetheless are densely populated. It therefore appears likely that the dense QP solver's performance on the unnecessarily large QP is worse than what could be achieved by a suitable exploitation of the sparse block structure for the case $n^{\mathrm{q}} \geq n^{\mathrm{x}}$.

4. Block Sparse Quadratic Programming: "Complementary Condensing"

In this section we present a new approach of solving the KKT system of a QP with block sparse structure due to multiple shooting that is suited for embedding in a standard active–set loop. This approach is based on related work by [35] and was first presented in [22]. It does not work as a preprocessing step but directly exploits the block sparse structure inside the solver. We derive in detail the necessary elimination steps that will ultimately retain the duals of the matching condition equalities only. In classical condensing, these were used for elimination, which gives rise to the name *complementary condensing* for our new method. An analysis of the run time complexity as well as a detailed account on the number of floating point operations spent in the various parts of the algorithm is presented.

4.1. **The Sparse KKT System Structure.** For a given active set, the KKT system of the QP (16) to be solved for the primal step δw_i and the dual step $(\delta\lambda, \delta\mu)$ reads for $0 \leq i \leq m$

$$P_i' \delta\lambda_{i-1} + B_i(-\delta w_i) + R_i' \delta\mu_i + X_i' \delta\lambda_i = B_i w_i + g_i \quad =: \overline{g}_i, \tag{26a}$$

$$R_i(-\delta w_i) = R_i w_i - r_i \quad =: \overline{r}_i, \tag{26b}$$

$$X_i(-\delta w_i) + P_{i+1}(-\delta w_{i+1}) = X_i w_i + P_{i+1} s_{i+1} - h_i =: \overline{h}_i. \tag{26c}$$

with Lagrange multipliers $\delta\lambda \in \mathbb{R}^{n^{\mathrm{x}}}$ for the matching conditions (16b) and $\delta\mu \in \mathbb{R}^{n^{\mathrm{r}}_i}$ for the equality point constraints and the active subset of the discretized inequality path and point constraints (16c). The projection matrices P_i are defined as

$$P_i := \begin{pmatrix} -I & 0 \end{pmatrix} \in \mathbb{R}^{n^x \times (n^x + n^q)},\ 1 \le i \le m, \tag{27}$$

and for the first and last shooting nodes as

$$P_0 := 0 \in \mathbb{R}^{n^x \times (n^x + n^q)}, \qquad P_{m+1} := 0 \in \mathbb{R}^{n^x \times n^x}. \tag{28}$$

In the following, all matrices and vectors are assumed to comprise the components of the active set only. To avoid the need for repeated special treatment of the first and last shooting node throughout this paper, we introduce the following conventions that make equation (26) hold also for the border cases $i = 0$ and $i = m$:

(29a)

$$\delta\lambda_{-1} := 0 \in \mathbb{R}^{n^x}, \qquad \lambda_{-1} := 0 \in \mathbb{R}^{n_x}, \qquad \delta\lambda_m := 0 \in \mathbb{R}^{n^x}, \qquad \lambda_m := 0 \in \mathbb{R}^{n^x},$$

(29b)

$$\delta w_{m+1} := 0 \in \mathbb{R}^{n^x}, \quad w_{m+1} := 0 \in \mathbb{R}^{n^x}, \qquad h_m := 0 \in \mathbb{R}^{n^x}, \qquad X_m := 0 \in \mathbb{R}^{n^x \times n^x}.$$

The system (26) can also be put in matrix form as

(30)

$$\begin{pmatrix} B_0 & R_0' & X_0' & & & & & & \\ R_0 & & & & & & & & \\ X_0 & & & P_1 & & & & & \\ & & P_1' & B_1 & R_1' & X_1' & & & \\ & & & R_1 & & & & & \\ & & & X_1 & & & & & \\ & & & & & & \ddots & & \\ & & & & & & & B_m & R_m' \\ & & & & & & & R_m & \end{pmatrix} \begin{pmatrix} -\delta w_0 \\ \delta\mu_0 \\ \delta\lambda_0 \\ -\delta w_1 \\ \delta\mu_1 \\ \delta\lambda_1 \\ \vdots \\ -\delta w_m \\ \delta\mu_m \end{pmatrix} = \begin{pmatrix} \overline{g}_0 \\ \overline{r}_0 \\ \overline{h}_0 \\ \overline{g}_1 \\ \overline{r}_1 \\ \overline{h}_1 \\ \vdots \\ \overline{g}_m \\ \overline{r}_m \end{pmatrix}$$

4.2. **Hessian Projection Step.** Under the assumption that the number of active point constraints does not exceed the number of unknowns (i.e. the active set is not degenerate), we can perform QR decompositions of the linearized point constraints matrices R_i,

$$R_i \underbrace{\begin{pmatrix} Y_i & Z_i \end{pmatrix}}_{=:Q_i} = \begin{pmatrix} R_i^{\mathrm{R}\prime} & 0 \end{pmatrix}. \tag{31}$$

Here Q_i are a unitary matrices and R_i^{R} is upper triangular. We partition δw_i into its range space part δw_i^Y and its null space part δw_i^Z, where the identity $\delta w_i = Y_i \delta w_i^Y + Z_i \delta w_i^Z$ holds. We find δw_i^Y from the range space projection of (26b)

$$R_i(-\delta w_i) = -R_i^{\mathrm{R}} \delta w_i^Y = \overline{r}_i. \tag{32}$$

We transform the remainder of the KKT system onto the null space of R_i by substituting $Y_i \delta w_i^Y + Z_i \delta w_i^Z$ for δw_i and solving for δw_i^Z. First, we find for the matching conditions (26c)

$$-X_i Z_i \delta w_i^Z - P_{i+1} Z_i \delta w_{i+1}^Z = \overline{h}_i + X_i Y_i \delta w_i^Y + P_{i+1} Y_i \delta w_{i+1}^Y \tag{33}$$

which can be solved for δw_i^Z once δw_{i+1}^Z is known. Second, from the stationarity conditions (26a) we find

$$Z_i' P_i{}' \delta\lambda_{i-1} - Z_i' B_i Z_i \delta w_i^Z + \underbrace{Z_i' R_i' \mu_i}_{=0} + Z_i' X_i' \delta\lambda_i = Z_i' \overline{g}_i + Z_i' B_i Y_i \delta w_i^Y \tag{34}$$

(35) and $$\underbrace{Y_i' R_i'}_{R_i^{\mathrm{R}}} \delta\mu_i = -Y_i'(B_i \delta w_i + P_i' \delta\lambda_{i-1} - X_i' \delta\lambda_i + \overline{g}_i).$$

Here, (34) can to be solved for $\delta\lambda_i$ once δw_i and $\delta\lambda_{i-1}$ are known, while (35) can be used to determine the point constraints multipliers $\delta\mu_i$. Let thus null space projections be defined as follows:

(36a) $\tilde{B}_i := Z_i' B_i Z_i, \quad \tilde{g}_i := Z_i' \overline{g}_i + Z_i' B_i Y_i \delta w_i^Y, \quad 0 \le i \le m,$

(36b) $\tilde{X}_i := X_i Z_i, \quad \tilde{h}_i := \overline{h}_i + X_i Y_i \delta w_i^Y + P_{i+1} Y_i \delta w_{i+1}^Y, \quad 0 \le i \le m-1,$

(36c) $\tilde{P}_i := P_i Z_i, \quad 0 \le i \le m-1.$

With this notation the projection of the KKT system on the null space of the point constraints can be read from equations (33) and (34) for $0 \le i \le m-1$ as

(37a) $$\tilde{P}_i' \delta\lambda_{i-1} + \tilde{B}_i(-\delta w_i^Z) + \tilde{X}_i' \delta\lambda_i = \tilde{g}_i,$$

(37b) $$\tilde{X}_i(-\delta w_i^Z) + \tilde{P}_{i+1}(-\delta w_{i+1}^Z) = \tilde{h}_i.$$

This again can be put in matrix form as

(38) $$\begin{pmatrix} \tilde{B}_0 & \tilde{X}_0' & & & & & \\ \tilde{X}_0 & & \tilde{P}_1 & & & & \\ & \tilde{P}_1' & \tilde{B}_1 & \tilde{X}_1' & & & \\ & & \tilde{X}_1 & & & & \\ & & & & \ddots & \tilde{P}_m \\ & & & & \tilde{P}_m' & \tilde{B}_m \end{pmatrix} \begin{pmatrix} -\delta w_0^Z \\ \delta\lambda_0 \\ -\delta w_1^Z \\ \delta\lambda_1 \\ \vdots \\ -\delta w_m^Z \end{pmatrix} = \begin{pmatrix} \tilde{g}_0 \\ \tilde{h}_0 \\ \tilde{g}_1 \\ \tilde{h}_1 \\ \vdots \\ \tilde{g}_m \end{pmatrix}.$$

4.3. **Schur Complement Step.** In (38) the elimination of δw^Z is possible using a Schur complement step, provided that the reduced Hessians $\tilde{B}_i$ are positive definite. We find from (37a)

(39) $$(-\delta w_i^Z) = \tilde{B}_i^{-1}(\tilde{g}_i - \tilde{P}_i' \delta\lambda_{i-1} - \tilde{X}_i' \delta\lambda_i)$$

depending on the knowledge of $\delta\lambda_i$. Inserting into (37b) and collecting for $\delta\lambda_i$ yields

(40) $$\tilde{X}_i \tilde{B}_i^{-1} \tilde{P}_i' \delta\lambda_{i-1} + (\tilde{X}_i \tilde{B}_i^{-1} \tilde{X}_i' + \tilde{P}_{i+1} \tilde{B}_{i+1}^{-1} \tilde{P}_{i+1}') \delta\lambda_i + \tilde{P}_{i+1} \tilde{B}_{i+1}^{-1} \tilde{X}_{i+1}' \delta\lambda_{i+1}$$
$$= -\tilde{h}_i + \tilde{X}_i \tilde{B}_i^{-1} \tilde{g}_i + \tilde{P}_{i+1} \tilde{B}_{i+1}^{-1} \tilde{g}_{i+1}$$

With Cholesky decompositions

(41) $$\tilde{B}_i = R_i^{\mathrm{B}\prime} R_i^{\mathrm{B}}$$

we define the following symbols

(42a)
$$\hat{X}_i := \tilde{X}_i R_i^{\mathrm{B}^{-1}}, \quad A_i := \tilde{X}_i \tilde{B}_i^{-1} \tilde{X}_i' + \tilde{P}_{i+1} \tilde{B}_{i+1}^{-1} \tilde{P}_{i+1}' = \hat{X}_i \hat{X}_i' + \hat{P}_{i+1} \hat{P}_{i+1}',$$

(42b)
$$\hat{P}_i := \tilde{P}_i R_i^{\mathrm{B}^{-1}}, \quad B_i := \tilde{X}_i \tilde{B}_i^{-1} \tilde{P}_i' = \hat{X}_i \hat{P}_i',$$

(42c)
$$\hat{g} := R_i^{\mathrm{B}^{-T}} \tilde{g}_i, \quad a_i := -\tilde{h}_i + \tilde{X}_i \tilde{B}_i^{-1} \tilde{g}_i + \tilde{P}_{i+1} \tilde{B}_{i+1}^{-1} \tilde{g}_{i+1} = -\tilde{h}_i + \hat{X}_i \hat{g}_i + \hat{P}_{i+1} \hat{g}_{i+1}.$$

Equation (40) can be written in terms of these values for $0 \le i \le m-1$ as

(43) $$B_i \delta\lambda_{i-1} + A_i \delta\lambda_i + B_{i+1}' \delta\lambda_{i+1} = a_i.$$

In matrix form, the remaining symmetric positive definite system reads

$$\begin{pmatrix} A_0 & B_1' & & \\ B_1 & A_1 & \ddots & \\ & \ddots & \ddots & B_{m-1}' \\ & & B_{m-1} & A_{m-1} \end{pmatrix} \begin{pmatrix} \delta\lambda_0 \\ \delta\lambda_1 \\ \vdots \\ \delta\lambda_{m-1} \end{pmatrix} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{m-1} \end{pmatrix}. \tag{44}$$

4.4. **Solving the Banded System.** In the symmetric positive definite banded system (44), only the matching condition duals $\delta\lambda_i \in \mathbb{R}^{n^{\mathrm{x}}}$ remain as unknowns. Since in classical condensing, exactly these matching conditions were used for elimination of a part of the primal unknowns, this new method is in a sense *complementary* to the classical condensing method. For optimal control problems with dimensions $n^{\mathrm{q}} \geq n^{\mathrm{x}}$, the presented approach obviously is computationally more favorable than retaining unknowns of dimension n^{q}. System (44) can be solved for $\delta\lambda$ by means of a tridiagonal block Cholesky decomposition [2] and two backsolves with the block Cholesky factors.

4.5. **Recovering the Sparse QP's Solution.** Once $\delta\lambda$ is known, the primal null space step δw^Z can be recovered using equation (39). The full primal step δw is then obtained from $\delta w = Y\delta w^Y + Z\delta w^Z$. Finally, the decoupled point constraint multipliers step $\delta\mu$ can be recovered by insertion into (35).

4.6. **Computational Complexity.** In table 7 a detailed list of the linear algebra operations required to carry out the individual steps of the complementary condensing method can be found. A clear distinction between matrix operations of quadratic or cubic runtime complexity on the one hand, and vector operations of linear or quadratic runtime complexity on the other hand has been made. The number of floating point operations required for the linear algebra operations, depending on the system dimensions $n = n^{\mathrm{x}} + n^{\mathrm{q}}$ and n_i^{r}, is given in table 8. The numbers n^{y} and n^{z} with $n^{\mathrm{y}} + n^{\mathrm{z}} = n_i^{\mathrm{r}}$ denote the range–space and null-space dimension resulting from the QR decomposition (31), respectively. All FLOP counts are given on a per shooting node basis. It's easy to see that the method's runtime complexity is $O(m)$, in sharp contrast to the classical condensing method, as the shooting grid length m does not appear as a dependency in table 8. In addition, the run time of a significant part of the complementary condensing, the decomposition of the banded system (44), even is *independent* of the number n^{q} of discrete choices.

4.7. **Complementary Condensing for the Example Problem.** In this final section concerning the newly developed complementary condensing approach, we apply our technique to the introductory vehicle control example. Table 9 lists the run times obtained for an ANSI C99 implementation of a primal active set QP code using the presented complementary condensing technique for the factorization of the KKT system. We compare the run times obtained for the exemplary vehicle control problem for different values of the shooting grid length m and the number n^{μ} of available gears.

The claimed run time complexity of $O(m)$ is easily seen in figure 3(a), while the $O({n^{\mathrm{q}}}^3)$ complexity is not noticeable for the examined instances, as the computationally demanding parts of the complementary condensing approach are independent of the number n^{q} of discrete choices. The growth of the run time of a single QP

Action	Matrix dec	Matrix bs	Matrix mul	Matrix add	Vector bs	Vector mul	Vector add
Decompose R_i	1	–	–	–			
Solve for δw^Y, $Y\delta w^Y$					1	1	–
Build $\tilde{B}_i$	–	–	2	–			
Build $\tilde{X}_i$, $\tilde{P}_i$	–	–	2	–			
Build $\tilde{g}_i$, $\tilde{h}_i$					–	4	3
Decompose $\tilde{B}_i$	1	–	–	–			
Build $\hat{X}_i$, $\hat{P}_i$	–	2	–	–			
Build A_i, B_i	–	–	3	1			
Build $\hat{g}_i$, a_i					1	2	2
Decompose (44)	1	1	1	–			
Solve for $\delta\lambda_i$					2	2	2
Solve for δw_i^Z, $Z\delta w_i^Z$					1	3	2
Solve for $\delta\mu_i$					1	4	3

TABLE 7. Number of matrix and vector operations per node required for the individual parts of the proposed block sparse solver, separated into decompositions (dec), backsolves (bs), multiplications (mul), and additions (add).

Action	Floating point operations
Decompose R_i	${n_i^{\mathrm{r}}}^2 n$
Solve for δw^Y, $Y\delta w^Y$	$n_i^{\mathrm{r}} n^{\mathrm{y}} + n^{\mathrm{y}} n$
Build $\tilde{B}_i$	${n^{\mathrm{z}}}^2 n + n^{\mathrm{z}} n^2$
Build $\tilde{X}_i$, $\tilde{P}_i$	$2n^{\mathrm{x}} n^{\mathrm{z}} n$
Build $\tilde{g}_i$, $\tilde{h}_i$	$2n^{\mathrm{x}} n + n^{\mathrm{z}} n + n^2 + 2n^{\mathrm{x}} + n$
Decompose $\tilde{B}_i$	$\frac{1}{3} {n^{\mathrm{z}}}^3$
Build $\hat{X}_i$, $\hat{P}_i$	$2n^{\mathrm{x}} {n^{\mathrm{z}}}^2$
Build A_i, B_i	$3{n^{\mathrm{x}}}^2 n^{\mathrm{z}} + {n^{\mathrm{x}}}^2$
Build $\hat{g}_i$, a_i	${n^{\mathrm{z}}}^2 + 2n^{\mathrm{x}} n^{\mathrm{z}} + 2n^{\mathrm{x}}$
Decompose (44)	$\frac{7}{3} {n^{\mathrm{x}}}^3$
Solve for $\delta\lambda_i$	$4{n^{\mathrm{x}}}^2 + 2n^{\mathrm{x}}$
Solve for δw_i^Z, $Z\delta w_i^Z$	${n^{\mathrm{z}}}^2 + 2n^{\mathrm{x}} n^{\mathrm{z}} + n^{\mathrm{z}} n + 2n^{\mathrm{z}}$
Solve for $\delta\mu_i$	$n_i^{\mathrm{r}} n^{\mathrm{y}} + n^{\mathrm{y}} n + 2n^{\mathrm{x}} n + n^2 + 3n$

TABLE 8. Number of floating point operations (FLOPs) per shooting node required for the individual parts of the proposed block sparse solver. One FLOP comprises one scalar floating point multiplication and addition. The numbers n^{y} and n^{z} with $n^{\mathrm{y}} + n^{\mathrm{z}} = n_i^{\mathrm{r}}$ denote the range–space and null-space dimension resulting from the QR decomposition (31), respectively. Further, we use $n := n^{\mathrm{x}} + n^{\mathrm{q}}$ to denote the system's dimension.

	$m = 20$	$m = 30$	$m = 40$	$m = 50$
$n^\mu = 4$	0.3	0.3	0.6	0.7
$n^\mu = 8$	0.4	0.6	0.7	0.9
$n^\mu = 12$	0.4	0.6	0.8	0.9
$n^\mu = 16$	0.4	0.6	0.9	1.0

Table 9. Average computation times in milliseconds per iteration of the proposed structure exploiting block sparse QP solver for the presented vehicle control problem with increasing horizon length and number of gears.

	$m = 20$	$m = 30$	$m = 40$	$m = 50$
$n^\mu = 4$	0.9	1.5	2.1	2.5
$n^\mu = 8$	1.5	2.2	3.0	3.8
$n^\mu = 12$	2.3	3.1	4.7	5.0
$n^\mu = 16$	2.6	4.3	6.0	8.1

Table 10. Average computation time in milliseconds per iteration of a symmetric indefinite factorization of the QP's KKT system using the highly efficient HSL subroutine MA57.

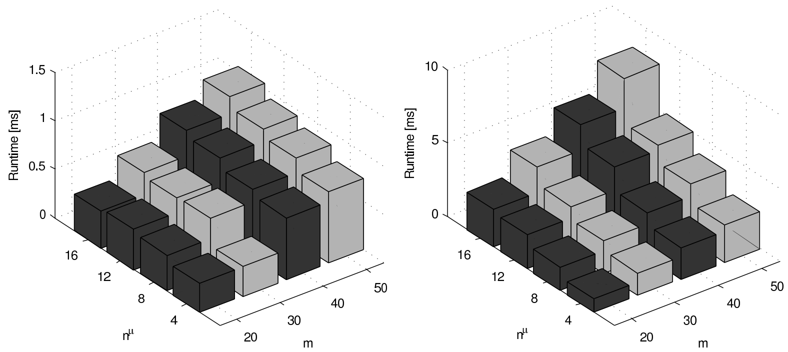
iteration with growing problem dimensions is very small. The total number of required QP iterations will still grow, though. For the largest instance examined, we find an average per iteration speedup of more than a factor of 20 when comparing the proposed block sparse active set solver to the dense active–set solver QPOPT running on the condensed QP. In addition, the run time required for condensing, up to 200 milliseconds per SQP iteration, is saved entirely.

In a model–predictive control setup, giving fast feedback close to the controlled process' reference trajectory will enable an active–set QP solver to complete in very few iterations, often requiring only one iteration, cf. [10, 11]. In this case, the proposed algorithm substitutes one block sparse QP iteration for condensing plus one dense QP iteration, and the achievable speedup will be as high as a factor of $(200 \text{ ms} + 20.7 \text{ ms})/1.0 \text{ ms} > 200$.

Table 10 compares the performance of the complementary condensing approach, which effectively proposes a factorization of the KKT system with special block structure, to the highly efficient multifrontal symmetric indefinite factorization subroutine MA57 [8, 9] available from the Harwell Subroutine Library (HSL). The computation times listed include the necessary assembly of the KKT matrix in triplets storage format. Our proposed method has a performance advantage of up to a factor of 8 for the largest examined instance. Since our method does not make use of pivoting strategies, MA57 is very likely to be numerically more stable. In addition, our method does not currently exploit model–inherent sparsity, i.e. structures of the KKT matrix induced by the model rather than by the multiple shooting discretization.

5. Conclusions and Outlook

5.1. **Conclusions.** In this contribution we have considered the solution of mixed–integer optimal control problems in ordinary differential equations. We treated the



(a) Average run time per iteration of the proposed block sparse solver.

(b) Average run time per iteration when solving the KKT system using HSL MA57.

FIGURE 3. Average run times in milliseconds per iteration of the proposed block sparse active set QP solver. The KKT system was solved with the proposed factorization (a) and with the highly efficient sparse symmetric indefinite factorization code HSL MA57 (b).

integer control by outer convexification [29] and reviewed the direct multiple shooting method [26, 7, 24] to obtain a discretized optimal control problem. Sequential quadratic programming methods have been our motivation to investigate the solution of the highly structured quadratic subproblems. We reviewed the classical condensing algorithm [26, 7, 24] that works as a preprocessing step for the quadratic subproblems, and enables the efficient usage of a wealth of available dense quadratic programming codes. Application of this approach to an exemplary vehicle control problem with gear shift revealed that for longer horizons or larger numbers of choices for the integer control, the classical condensing algorithm leaves room for improvement. To address this issue, we presented a new approach at solving the highly structured quadratic program by devising a new factorization of the QP's KKT matrix that respects the block structure introduced by direct multiple shooting. We employed this new method to solve the exemplary vehicle control problem and compared it a) to the classical condensing approach, and b) to the highly efficient sparse symmetric indefinite factorization code MA57 which was used as an alternative means to obtain a factorization of the highly structured QP's KKT matrix. The presented computational results indicate that the proposed method is able to deliver promising run times for all examined instances of the vehicle control problem. We derived an $O(mn^3)$ runtime complexity for our method, in contrast to $O(m^2n^3)$ for the classical condensing. A speedup of a factor of 20 was obtained for the largest instance of the example problem examined, and we proposed a speedup of a factor of 200 for a special model predictive control scenario.

5.2. **Future Work.** Future work on the complementary condensing algorithm presented in section 4 includes the following topics.
Exploiting simple bounds. A first improvement to the presented approach inside an active–set loop would be the exploitation of so–called *simple bounds* $\underline{w}_i \leq \delta w_i \leq \overline{w}_i$ on the unknowns by introducing the notion of *free* and *fixed unknowns*. This

effectively reduces the size of the matrices B_i, R_i, and X_i to be decomposed and multiplied during solution of the KKT system.

Devising a chain of decomposition updates. Within the active–set loop of the QP solver, one QR decomposition of R_i, one $R'R$ decomposition of the null–space Hessians $\tilde{B}_i$, and the banded Cholesky decomposition of the entire system (44) have to be recomputed whenever a point constraint enters or leaves the active set. When exploiting simple bounds, the same holds true whenever an unknown hits or leaves one of its bounds. From dense null–space and range–space methods it is common knowledge that certain decompositions can be updated during an active set change in $O(n^2)$ time [16], relieving the algorithm from the burden of having to recompute the entire decomposition in $O(n^3)$ time. Such update techniques would essentially remove all matrix decompositions and matrix–matrix operations listed in table 7 from the active–set loop. This would yield an $O(mn^2)$ block structured active set method with only an initial factorization in $O(mn^3)$ time. Investigation into updates for the presented factorization shall be the topic of a forthcoming publication.

References

[1] J. Albersmeyer and H. G. Bock. Sensitivity Generation in an Adaptive BDF-Method. In H. G. Bock, E. Kostina, X. Phu, and R. Rannacher, editors, *Modeling, Simulation and Optimization of Complex Processes: Proceedings of the International Conference on High Performance Scientific Computing, March 6-10, 2006, Hanoi, Vietnam*, pages 15–24. Springer, 2008.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 3rd edition, 1999. ISBN 0-89871-447-8 (paperback).

[3] R. Bartlett and L. Biegler. QPSchur: A dual, active set, schur complement method for large-scale and structured convex quadratic programming algorithm. *Optimization and Engineering*, 7:5–32, 2006.

[4] R. Bartlett, A. Wächter, and L. Biegler. Active set vs. interior point strategies for model predicitve control. In *Proceedings of the American Control Conference*, pages 4229–4233, Chicago, IL, 2000.

[5] L. Biegler. Solution of dynamic optimization problems by successive quadratic programming and orthogonal collocation. *Computers and Chemical Engineering*, 8:243–248, 1984.

[6] H. G. Bock. Numerical treatment of inverse problems in chemical reaction kinetics. In K. Ebert, P. Deuflhard, and W. Jäger, editors, *Modelling of Chemical Reaction Systems*, volume 18 of *Springer Series in Chemical Physics*, pages 102–125. Springer, Heidelberg, 1981.

[7] H. G. Bock and K. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings 9th IFAC World Congress Budapest*, pages 243–247. Pergamon Press, 1984. Available at http://www.iwr.uni-heidelberg.de/groups/agbock/FILES/Bock1984.pdf.

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

[9] I. Duff and J. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Transactions on Mathematical Software*, 9(3):302–325, 1983.

[10] H. Ferreau. An online active set strategy for fast solution of parametric quadratic programs with applications to predictive engine control. Master's thesis, University of Heidelberg, 2006.

[11] H. Ferreau, H. G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.

[12] R. Fletcher. Resolving degeneracy in quadratic programming. Numerical Analysis Report NA/135, University of Dundee, Dundee, Scotland, 1991.

[13] M. Gerdts. Solving mixed-integer optimal control problems by Branch&Bound: A case study from automobile test-driving with gear shift. *Optimal Control Applications and Methods*, 26:1–18, 2005.

[14] M. Gerdts. A variable time transformation method for mixed-integer optimal control problems. *Optimal Control Applications and Methods*, 27(3):169–182, 2006.

[15] E. Gertz and S. Wright. Object-oriented software for quadratic programming. *ACM Transactions on Mathematical Software*, 29:58–81, 2003.

[16] P. Gill, G. Golub, W. Murray, and M. A. Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, 28(126):505–535, 1974.

[17] P. Gill, W. Murray, and M. Saunders. *User's Guide For QPOPT 1.0: A Fortran Package For Quadratic Programming*, 1995. Available at http://www.sbsi-sol-optimize.com/manuals/QPOPT%20Manual.pdf.

[18] N. Gould and P. Toint. A quadratic programming bibliography. Technical Report 01/02, Rutherford Appleton Laboratory, Computational Science and Engineering Department, June 2003.

[19] S. Han. Superlinearly convergent variable-metric algorithms for general nonlinear programming problems. *Mathematical Programming*, 11:263–282, 1976.

[20] E. Hellström, M. Ivarsson, J. Åslund, and L. Nielsen. Look-ahead control for heavy trucks to minimize trip time and fuel consumption. *Control Engineering Practice*, 17:245–254, 2009.

[21] H. Huynh. *A Large-Scale Quadratic Programming Solver Based On Block-LU Updates of the KKT System.* PhD thesis, Stanford University, 2008.

[22] C. Kirches, H. G. Bock, J. Schlöder, and S. Sager. Complementary condensing for the direct multiple shooting method. In H. Bock, E. Kostina, H. Phu, and R. Rannacher, editors, *Proceedings of the Fourth International Conference on High Performance Scientific Computing: Modeling, Simulation, and Optimization of Complex Processes, Hanoi, Vietnam, March 2–6, 2009*, Springer Verlag Berlin Heidelberg New York, 2009. submitted.

[23] C. Kirches, S. Sager, H. G. Bock, and J. P. Schlöder. Time-optimal control of automobile test drives with gear shifts. *Optimal Control Applications and Methods*, 30(5), September/October 2009. DOI 10.1002/oca.892.

[24] D. Leineweber, I. Bauer, H. G. Bock, and J. P. Schlöder. An efficient multiple shooting based reduced SQP strategy for large-scale dynamic process optimization. Part I: Theoretical aspects. *Computers and Chemical Engineering*, 27:157–166, 2003.

[25] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Verlag, Berlin Heidelberg New York, 2nd edition, 2006. ISBN 0-387-30303-0.

[26] K. Plitt. Ein superlinear konvergentes Mehrzielverfahren zur direkten Berechnung beschränkter optimaler Steuerungen. Master's thesis, Universität Bonn, 1981.

[27] A. Potschka, H. G. Bock, and J. P. Schlöder. A minima tracking variant of semi-infinite programming for the treatment of path constraints within direct solution of optimal control problems. *Optimization Methods and Software*, 24(2):237–252, 2009.

[28] M. Powell. Algorithms for nonlinear constraints that use lagrangian functions. *Mathematical Programming*, 14(3):224–248, 1978.

[29] S. Sager. *Numerical methods for mixed–integer optimal control problems*. Der andere Verlag, Tönning, Lübeck, Marburg, 2005. ISBN 3-89959-416-9. Available at http://sager1.de/sebastian/downloads/Sager2005.pdf.

[30] S. Sager. Reformulations and algorithms for the optimization of switching decisions in nonlinear optimal control. *Journal of Process Control*, 2009. DOI 10.1016/j.jprocont.2009.03.008.

[31] S. Sager, H. G. Bock, and M. Diehl. The integer approximation error in mixed-integer optimal control. *Optimization Online*, 2:1–16, 2009. Submitted to Mathematical Programming A.

[32] S. Sager, C. Kirches, and H. G. Bock. Fast solution of periodic optimal control problems in automobile test-driving with gear shifts. In *Proceedings of the 47th IEEE Conference on Decision and Control (CDC 2008), Cancun, Mexico*, pages 1563–1568, 2008. ISBN: 978-1-4244-3124-3.

[33] S. Sager, G. Reinelt, and H. G. Bock. Direct methods with maximal lower bound for mixed-integer optimal control problems. *Mathematical Programming*, 118(1):109–149, 2009.

[34] C. Schmid and L. Biegler. Quadratic programming methods for tailored reduced Hessian SQP. *Computers & Chemical Engineering*, 18(9):817–832, September 1994.

[35] M. Steinbach. *Fast recursive SQP methods for large-scale optimal control problems*. PhD thesis, Universität Heidelberg, 1995.

[36] S. Terwen, M. Back, and V. Krebs. Predictive powertrain control for heavy duty trucks. In *Proceedings of IFAC Symposium in Advances in Automotive Control*, pages 451–457, Salerno, Italy, 2004.

[37] J. Till, S. Engell, S. Panek, and O. Stursberg. Applied hybrid system optimization: An empirical investigation of complexity. *Control Engineering Practice*, 12:1291–1303, 2004.

[38] R. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Methods and Software*, 11(1–4):451–484, 1999.

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

[40] R. C. Whaley and A. Petitet. Minimizing development and maintenance costs in supporting persistently optimized BLAS. *Software: Practice and Experience*, 35(2):101–121, February 2005. Available at http://www.cs.utsa.edu/~whaley/papers/spercw04.ps.