Manuscript submitted to
AIMS' Journals
Volume **X**, Number **0X**, XX **200X**

doi:10.3934/xx.xx.xx.xx

pp. **X–XX**

A STUDY OF STRUCTURE-EXPLOITING SQP ALGORITHMS FOR AN OPTIMAL CONTROL PROBLEM WITH COUPLED HYPERBOLIC AND ORDINARY DIFFERENTIAL EQUATION CONSTRAINTS

Jan-Hendrik Webert

Institut für Mathematik und Rechneranwendung (LRT-1), Universität der Bundeswehr
Werner-Heisenberg-Weg 39, 85577 Neubiberg/München, Germany

Philip E. Gill

Center for Computational Mathematics
University of California, San Diego
9500 Gilman Drive, La Jolla, CA 92093-0112, USA

Sven-Joachim Kimmerle*

Institut für Mathematik und Bauinformatik (BAU-1), Universität der Bundeswehr
Werner-Heisenberg-Weg 39, 85577 Neubiberg/München, Germany

Matthias Gerdts

Institut für Mathematik und Rechneranwendung (LRT-1), Universität der Bundeswehr
Werner-Heisenberg-Weg 39, 85577 Neubiberg/München, Germany

Abstract. In this article, structure-exploiting optimisation algorithms of the sequential quadratic programming (SQP) type are considered for optimal control problems with control and state constraints. Our approach is demonstrated for a 1D mathematical model of a vehicle transporting a fluid container. The model involves a fully coupled system of ordinary differential equations (ODE) and nonlinear hyperbolic first-order partial differential equations (PDE), although the ideas for exploiting the particular structure may be applied to more general optimal control problems as well. The time-optimal control problem is solved numerically by a full discretisation approach. The corresponding nonlinear optimisation problem is solved by an SQP method that uses exact first and second derivative information. The quadratic subproblems are solved using an active-set strategy. In addition, two approaches are examined that exploit the specific structure of the problem: (A) a direct method for the KKT system, and (B) an iterative method based on combining the limited-memory BFGS method with the preconditioned conjugate gradient method. Method (A) is faster for our model problem, but can be limited by the problem size. Method (B) opens the door for a potential extension of the truck-container model to three space dimensions.

2010 *Mathematics Subject Classification.* Primary: 49J15, 49J20, 90C53, 90C55; Secondary: 35Q35, 35L04, 49N90.

Key words and phrases. ODE-PDE constrained optimisation, SQP method, Limited-Memory-BFGS method, Saint-Venant equations, shallow water equations, fully coupled ODE-PDE system, Lax-Friedrichs scheme, hyperbolic conservation law.

* Corresponding author: Sven-Joachim Kimmerle.

1. **Introduction.** Optimal control problems arise in various contexts, such as economics, biology or engineering. The dynamic behaviour of these systems is often subject to constraints and can be influenced by control variables. In this article, a mechanical system of a truck with a container as a load is considered. By means of optimal control, the control of the mechanical system for the performance of certain predefined manoeuvres is to be determined. The truck-container system is modeled mathematically as a set of fully coupled ODEs and PDEs. The dynamics of the truck are described by the Lagrangian equations, whereas the motion of the fluid is characterised by the shallow water equations.

The mathematical model has been developed in [7]. Herein, the optimal control problems were solved numerically by an adjoint-based direct shooting method. In [10] the same problem is revisited using a first-optimise-then-discretise approach. In this study we follow a first-discretise-then-optimise approach in which we exploit the specific structure of the fully discretised optimisation problem. This article is based on work completed as part of Jan-Hendrik Webert's M.Sc. thesis [15]. Direct discretisation of the optimal control problem yields a nonlinear finite-dimensional optimisation problem which is large-scale and exhibits a sparse structure. Although software (commercial and non-commercial) for the solution of optimisation problems that exhibit this structure, such as `SNOPT`, `IPOPT` or `WORHP`, is available, this article focuses on the implementation and evaluation of alternative algorithms. In a first step, the structure of the discretised optimal control problem of the truck-container system is analysed. Exact first and second derivatives, such as the Jacobian of the constraints and the Hessian of the Lagrangian are derived analytically and exploited in the solution process. In a second step, two solution approaches are implemented, adjusted to the specific characteristics of the problem and tested in numerical experiments. Both approaches employ the sequential quadratic programming (SQP) method, where the resulting quadratic programs (QP) are solved with an active-set strategy. The main effort in the solution process involves the solution of large and sparse systems of linear equations, the so-called KKT systems. The first method relies on the direct solution of the KKT systems with exact second derivatives and `MA57`, a well-known solver for large and sparse systems. For the second method, a block-elimination approach, the so-called range-space method, is chosen. Exact second derivatives are replaced by a Limited-Memory-BFGS approximation (cf. [11]) and the resulting linear systems are solved with a preconditioned conjugate gradient method. We emphasise that our structure exploitation methods may apply well to other optimal control problems with control and state constraints, though we present our approach in this study on the basis of the aforementioned truck-container problem.

This study is structured as follows: after a brief introduction to the mathematical model of the truck-container system, the nonlinear optimisation problem to be solved is formulated. Furthermore, the exact first and second derivatives associated with the optimisation problem are given in this section. Section 3 describes the model verification process with the optimisation software package `SNOPT` and the analysis of certain influential model parameters. In Section 4, both solution techniques mentioned above are outlined in detail, including implementational details. Computational results are presented in Section 5, before we close with a discussion.

2. **Mathematical model with coupled hyperbolic and ordinary differential equations.** The technical application that is subject to optimal control in this

article is a truck-container system, performing certain driving manoeuvres. Being part of a technical system in motion, the fluid in the container develops its own dynamics, which in turn has an impact on the whole system. In combination with spring-damper elements, unwanted and potentially harmful oscillations can occur. By means of optimal control, the manoeuvering performance of the truck-container system can be improved, for example by reducing braking times or preventing fluid overflow, whilst maintaining the safe operability of the technical system.

In the following two subsections we briefly review the optimal control problem for the truck-container system described in [7].

2.1. **Mathematical model.** We consider a truck with a container containing a fluid, which is linked to the truck by a spring-damper connection, as depicted in Figure 1. The truck, modeled as a point mass of mass m_T, can move in the horizontal direction d and is controlled by a time-dependent force $u(t)$, representing acceleration or deceleration. The distances travelled by the truck and the fluid container in the (X, Z) reference system are denoted by d_T and d_W and their velocities by $\dot{d}_T$ and $\dot{d}_W$, respectively. The function $b(x)$ characterises the bottom surface

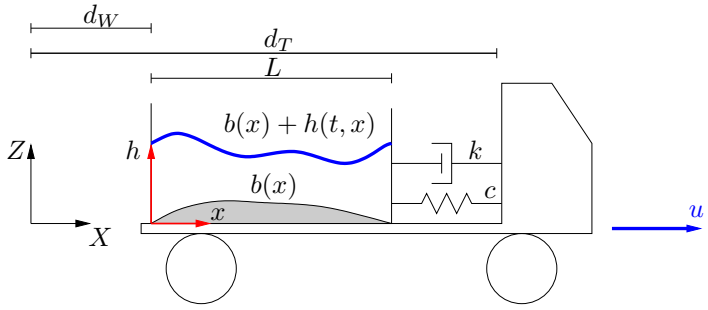

FIGURE 1. Model of the truck and fluid tank, cf. [7].

of the fluid container for $0 \leq x \leq L$, L being the length of the container. Both the height of the fluid column $h(t,x)$ and the horizontal fluid velocity $v(t,x)$ in the moving coordinate system (x,h) obey the Saint-Venant-equations (shallow water equations). The overall mass of the container is given by

$$m_W = \hat{\rho}_W \int_0^L (h_0(x) + b(x))\, dx,$$

where $h_0(x)$ is the initial height of the fluid column and $\hat{\rho}_W \left[kg/m^2\right]$ is the mass of the fluid per area. The grey area in Fig. 1 below $b(x)$ is assumed to have the same density as the fluid. For simplicity, the container itself is assumed to have no mass.

Truck dynamics. The motion of the truck is characterised by a set of ordinary differential equations. Let T denote the final time. Applying the horizontal equilibrium of forces to the masses m_T and m_W and neglecting any friction terms yields the equations

$$m_T \ddot{d}_T = u - F, \qquad t \in (0,T), \tag{2.1a}$$

$$m_W \ddot{d}_W = F + m_W a_W, \qquad t \in (0,T), \tag{2.1b}$$

where F is the spring damper force and a_W the average acceleration of the fluid in the container. The spring-damper force, which obeys the linear spring and damper force laws, is given by

$$F(d_T, d_W, \dot{d}_T, \dot{d}_W) = c(d_T - d_W + \overline{d}) + k(\dot{d}_T - \dot{d}_W), \tag{2.2}$$

with the constant spring stiffness $c\big[N/m\big]$ and the damper force constant $k\big[Ns/m\big]$. Furthermore, the mean acceleration of the fluid, a_W, is given by

$$a_W = \frac{1}{L}\int_0^L v_t(t,x)\,dx, \tag{2.3}$$

where $v_t(t,x)$ denotes the horizontal acceleration at the point (t,x).

Shallow water equations. Fluid motion is generally described by the Navier-Stokes equations. Assuming that horizontal scales are much larger than vertical scales and neglecting 3D effects in the fluid motion, these equations can be simplified significantly, thereby reducing complexity and computational effort, whilst still conveying the desired information. The fluid is supposed to be incompressible. In this article the fluid is assumed to be water. For $h > 0$ the resulting shallow water equations (SWE) may be written as

$$h_t + (hv)_x = 0, \qquad (t,x) \in \Omega, \tag{2.4a}$$

$$v_t + \left(\frac{1}{2}v^2 + gh\right)_x = -gb_x - \frac{F}{m_W}, \qquad (t,x) \in \Omega, \tag{2.4b}$$

where $\Omega := (0,T) \times (0,L)$ and g is the gravity acceleration. The right-hand side in (2.4b) models a non-straight bottom geometry $b(x)$ [14, pp. 3-4] and the acceleration of the fluid container due to the coupling force $F(t)$. It must be noted that the fluid-level in the coordinate system (x,h) is $b(x)+h(t,x)$. The initial and boundary conditions for the SWE (2.4) are given by

$$\begin{aligned} h(0,x) &= h_0(x), \qquad & v(0,x) &= v_0(x), \qquad & x &\in [0,L], \\ v(t,0) &= 0, & v(t,L) &= 0, & t &\in [0,T], \end{aligned}$$

and it is assumed that

$$\underline{h} \le h(t,x) \le \overline{h} \quad \text{with} \quad 0 < \underline{h} < \overline{h} < H(x), \tag{2.5}$$

where $\underline{h}$ and $\overline{h}$ are given numbers and $H(x)$ is the wall height of the tank. For compatibility, $h_0(x)$ must also satisfy (2.5). As the fluid cannot penetrate the tank wall, we have

$$v(t,0) = v(t,L) = 0, \qquad t \in [0,T], \tag{2.6}$$

and it follows that $v_t \equiv 0$ at the left and right boundary of the tank. Exploiting this information in Equation (2.4b), boundary conditions for h can be obtained at the boundary points $x_e = 0$ and $x_e = L$, respectively:

$$h_x(t,x_e) = -b_x(x_e) - \frac{1}{gm_W}F\Big|_t, \qquad t \in [0,T].$$

Solving (2.4b) for v_t and inserting the resulting expression into the integral in (2.3), the mean fluid acceleration simplifies to

$$a_W(t) = -\frac{g}{L}\left(b(L) - b(0)\right) - \frac{F}{m_W} - \frac{g}{L}\left(h(t,L) - h(t,0)\right),$$

where (2.6) was exploited. The expression for a_W can be inserted in (2.1b) to yield

$$m_W \ddot{d}_W = -m_W \frac{g}{L}[b(x) + h(t,x)]_{x=0}^{x=L}, \qquad t \in (0,T), \tag{2.7}$$

where the spring damper force F cancels out. In summary, the technical system considered in this study is characterised by a set of fully coupled ODEs and PDEs. In the following, the ODEs are transformed to a first-order system by introducing the variables $\eta_T := \dot{d}_T$ and $\eta_W := \dot{d}_W$.

2.2. **The optimal control problem.** With the set of differential equations describing the truck and fluid motion available, we can pose an optimal control problem for the truck motion from a given initial state to a terminal state. The initial state is characterised by the initial position and velocity of the truck $d_T(0) = 0$ and $\dot{d}_T(0) = \tilde{\eta}_T^0$ and of the fluid container $d_W(0) = \tilde{d}_W^0$ and $\dot{d}_W(0) = \tilde{\eta}_W^0$. For the terminal state, the truck and the container position are given by $d_T(T) = \tilde{d}_T^T$ and $d_W(T) = \tilde{d}_W^T$ with the free final time T and the given scalars $\tilde{\eta}_T^0$, $\tilde{d}_W^0$, $\tilde{\eta}_W^0$, $\tilde{d}_T^T$, and $\tilde{d}_W^T$.

The objective function J to be minimised is composed of a sum of several functions, where each summand is weighted with a non-negative constant α_i:

$$\begin{aligned} J =& \alpha_0 T + \alpha_1 \int_0^T \int_0^L \left(h(t,x) - h_d(x)\right)^2 \, dx \, dt + \alpha_2 \int_0^T \frac{u(t)^2}{m_T^2} dt \\ &+ \alpha_3 \left(|\eta_T(T) - \tilde{\eta}_T^T|^2 + |\eta_W(T) - \tilde{\eta}_W^T|^2 \right). \end{aligned}$$

Here the summands to be minimised are: the final time T, the deviation from a given (constant) fluid-level h_d, the control effort, and the deviation of the terminal truck and container velocities from given terminal velocities $\tilde{\eta}_T^T$ and $\tilde{\eta}_W^T$. If $\tilde{\eta}_T^T = \tilde{\eta}_W^T = 0$, a braking manoeuvre is performed, where truck and container have come to a halt after the time T and given distance. To ensure that the acceleration and deceleration of the truck is limited, we impose bounds on the control

$$u_L \leq u(t) \leq u_R \quad \text{with} \quad u_L < u_R.$$

Furthermore, to prevent fluid overflow the fluid-level must not reach the upper rim of the tank. Fluid overflow would imply a reduction in mass, which is not included in the mathematical model. In addition, the shallow water equation model requires the bottom of the tank to be covered everywhere with fluid. These restrictions are enforced by the constraint (2.5) bounding the fluid column height.

The domain Ω contains the free end time T, which is not known a priori. Furthermore, T appears in the bounds of the objective function integrals. For the numerical solution of the problem, it is convenient to consider a domain with a fixed time horizon $\tilde{\Omega} := (0,1) \times (0,L)$. This can be achieved by means of the time transformation $t(\tau) = \tau T$ with $\tau \in [0,1]$, where T is considered as an additional optimisation variable. All time dependent expressions, including ODEs, PDEs, and the objective function, must to be transformed. For example, in the transformed equations there is an additional multiplicative factor $1/T$ associated with each of the time derivatives, and the integrations for J are performed from 0 to 1, while dt is scaled by the multiplicative factor T. For improved readability, the normalised time τ is denoted again by t in the following.

2.3. **Discretisation.** The domain $\tilde{\Omega}$ is discretised by an equidistant grid in space and time

$$x_i = i\Delta x, \quad i = 0, \dots, M, \qquad t_j = j\Delta t, \quad j = 0, \dots, N,$$

with $\Delta x = L/M$ and $\Delta t = 1/N$, respectively. The continuous state and control variables are approximated by discrete functions defined on the grid, e.g., $u^j \approx u(t_j)$ for all $j = 0, \dots, N-1$. Similarly, we introduce d_T^j, d_W^j, η_T^j, η_W^j, h_i^j, and v_i^j for $i = 0, \dots, M$ and $j = 0, \dots, N$. We define the vector of discretised state variables

$$q = \left[q^0, q^1, \dots, q^N\right]^\top \quad \in \mathbb{R}^{(N+1)(6+2M)}, \tag{2.8}$$

where $q^j = [d_T^j, d_W^j, \eta_T^j, \eta_W^j, h_0^j, \dots, h_M^j, v_0^j, \dots, v_M^j] \in \mathbb{R}^{6+2M}$, and the vector of the discretised control and final time

$$w = \left[u^0, \dots, u^{N-1}, T\right]^\top \quad \in \mathbb{R}^{N+1}. \tag{2.9}$$

Finally, the vector z is defined by $z = [q, w]^\top$.

In order to rewrite the optimal control problem (OCP) as a nonlinear program (NLP), the dynamic constraints corresponding to the ODE and PDE must be discretised. Various discretisation methods can be applied. For the discretisation of the ODE, the explicit Euler method was chosen. The shallow water equations can be discretised, for example, with the Lax-Friedrichs (LF) method or the Godunov scheme, both discretisation schemes being suitable for the approximation of hyperbolic conservation laws, such as the shallow water equations. For the discretisation of the PDE in this study, the LF method was chosen. The LF scheme exhibits some desirable properties, as it is consistent, conservative, and monotone [2]. Approximating the original conservation law with the LF method is equivalent to solving the original equation with an artificial diffusion term. This property is referred to as numerical dissipation or numerical viscosity, which reduces undesired oscillations of the numerical solution. The artificial diffusion term guarantees a unique solution, whereas hyperbolic conservation laws have, in general, no unique solution without imposing further so-called entropy conditions that follow from physical principles. Furthermore, the solution obtained in the limit of vanishing numerical viscosity also satisfies the entropy condition.

For the LF scheme it is necessary and sufficient for convergence that the Courant-Friedrichs-Lewy (CFL) condition is satisfied. Informally speaking, the space and time grids must be harmonised so that the duration between two neighbouring time gridpoints is less than the time needed by the wave to travel from one time gridpoint to an other. For our problem, it was found experimentally in [7, Ch. 3] that the CFL condition implies $30\Delta t \approx \Delta x/L$ and $N \approx 30M$ for the time and space grids.

The discretised differential equations may be written in the form

$$q^{j+1} = q^j + \Delta t\, \Phi^j(q^j, u^j, T),$$

where we introduce the increment function Φ^j by

$$\Phi^j_{d_T} = T\eta^j_T, \qquad \Phi^j_{d_W} = T\eta^j_W, \qquad \Phi^j_{\eta_T} = \frac{T}{m_T}\left(u^j - F^j\right),$$
$$\Phi^j_{\eta_W} = \frac{-gT}{L}\left([b(x)]^L_{x=0} - \Delta x(b_x(L) + b_x(0)) + h^j_{M-1} - h^j_1 - \frac{2\Delta x}{gm_W}F^j\right),$$
$$\Phi^{i,j}_h = \frac{-T}{2\Delta x}\left(h^j_{i+1}v^j_{i+1} - h^j_{i-1}v^j_{i-1}\right),$$
$$\Phi^{i,j}_v = \frac{-T}{2\Delta x}\left(\frac{1}{2}(v^j_{i+1})^2 + gh^j_{i+1} - \frac{1}{2}(v^j_{i-1})^2 - gh^j_{i-1}\right) + T\left(-gb_x(x_i) - \frac{1}{m_W}F^j\right).$$

Here we have abbreviated $F^j = c(d^j_T - d^j_W + \overline{d}) + k(\eta^j_T - \eta^j_W)$. For the Neumann boundary conditions in h we use the notation

$$\Psi^j_e(d^j_T, d^j_W, \eta^j_T, \eta^j_W) = \Delta x\left(b_x(x_e) + \frac{1}{g\,m_W}F^j\right),$$

where the index $e = l$ stands for the left boundary at $x_e = 0$ and $e = r$ for $x_e = L$, respectively. Approximating the time and space derivatives with the LF-method yields the discretised differential equations

$$C(z) = 0 \in \mathbb{R}^{N(4+2M)+2} \tag{2.10}$$

with

$$C = \begin{bmatrix} C^0 \\ C^1 \\ \vdots \\ C^{N-1} \\ H^N_r \end{bmatrix}, \text{ where } C^j = \begin{bmatrix} d^j_T + \Delta t\Phi^j_{d_T} - d^{j+1}_T \\ d^j_W + \Delta t\Phi^j_{d_W} - d^{j+1}_W \\ \eta^j_T + \Delta t\Phi^j_{\eta_T} - \eta^{j+1}_T \\ \eta^j_W + \Delta t\Phi^j_{\eta_W} - \eta^{j+1}_W \\ \frac{1}{2}(h^j_2 + h^j_0) + \Delta t\Phi^{1,j}_h - h^{j+1}_1 \\ \vdots \\ \frac{1}{2}(h^j_M + h^j_{M-2}) + \Delta t\Phi^{M-1,j}_h - h^{j+1}_{M-1} \\ \frac{1}{2}(v^j_2 + v^j_0) + \Delta t\Phi^{1,j}_v - v^{j+1}_1 \\ \vdots \\ \frac{1}{2}(v^j_M + v^j_{M-2}) + \Delta t\Phi^{M-1,j}_v - v^{j+1}_{M-1} \\ h^j_0 - \Delta x\Psi^j_l - h^j_1 \\ h^j_{M-1} - \Delta x\Psi^j_r - h^j_M \end{bmatrix} \in \mathbb{R}^{4+2M},$$

i.e., a subset of $4+2M$ constraints with time index $j = 0, \ldots, N-1$. The term H^N_r equals the last two constraints in C^j but with the time index N. As it was done for the vector of the nonlinear equality constraints, the vector of the linear boundary conditions

$$B(z) = 0 \in \mathbb{R}^{2(M+N)+8} \tag{2.11}$$

is sorted by the time index of the variables. We obtain $B = \left[B^0, B^1, \ldots, B^N\right]^\top$, where

$$B^0 = \begin{bmatrix} d_T^0 \\ d_W^0 - \tilde{d}_W^0 \\ \eta_T^0 - \tilde{\eta}_T^0 \\ \eta_W^0 - \tilde{\eta}_W^0 \\ h_1^0 - h_0(x_1) \\ \vdots \\ h_{M-1}^0 - h_0(x_{M-1}) \\ v_0^0 \\ v_1^0 - v_0(x_1) \\ \vdots \\ v_{M-1}^0 - v_0(x_{M-1}) \\ v_M^0 \end{bmatrix}, \quad B^j = \begin{bmatrix} v_0^j \\ v_M^j \end{bmatrix}, j = 1, \ldots, N-1, \quad B^N = \begin{bmatrix} v_0^N \\ v_M^N \\ d_T^N - \tilde{d}_T^T \\ d_W^N - \tilde{d}_W^T \end{bmatrix}.$$

The box constraints for control u and the state h, in ascending time order, are

$$G(z) \leq 0 \in \mathbb{R}^{2N(M+2)} \tag{2.12}$$

with $G = \left[G^0, \ldots, G^N\right]^\top$, where

$$G^0 = \begin{bmatrix} u^0 - u_R \\ -u^0 + u_L \end{bmatrix}, \quad G^j = \begin{bmatrix} u^j - u_R \\ -u^j + u_L \\ h_0^j - \overline{h} \\ \vdots \\ h_M^j - \overline{h} \\ -h_0^j + \underline{h} \\ \vdots \\ -h_M^j + \underline{h} \end{bmatrix}, j = 1, \ldots, N-1, \quad G^N = \begin{bmatrix} h_0^N - \overline{h} \\ \vdots \\ h_M^N - \overline{h} \\ -h_0^N + \underline{h} \\ \vdots \\ -h_M^N + \underline{h} \end{bmatrix}.$$

Furthermore, the tracking-type term integral is approximated by the trapezoidal rule. Thus, in the following, we abbreviate $\omega_0 = \omega_M = 1/2$ and $\omega_i = 1$, $i = 1, \ldots, M$. In addition, we add a regularization term for the variation of the control with a weight factor $\alpha_4 \geq 0$. The reasons for this modification will be explained in Section 3.

Finally, for further details we refer to [15]. Our fully discretised optimal control problem reads:
Minimise

$$\begin{aligned} \tilde{J} =& \alpha_0 T + \frac{\alpha_1 T \Delta t \Delta x}{2} \sum_{j=0}^{N-1} \sum_{i=0}^{M} \omega_i \left(h_i^j - h_d(x_i)\right)^2 + \frac{\alpha_2 T \Delta t}{2 m_T^2} \sum_{j=0}^{N-1} \left(u^j\right)^2 \\ &+ \frac{\alpha_3}{2} \left(|\eta_T^N - \tilde{\eta}_T^T|^2 + |\eta_W^N - \tilde{\eta}_W^T|^2\right) + \frac{\alpha_4 T \Delta t}{2 m_T^2} \sum_{j=0}^{N-2} \left(\frac{u^{j+1} - u^j}{\Delta t}\right)^2 \end{aligned}$$

with respect to $z \in \mathbb{R}^{(N+1)(7+2M)}$, subject to the general constraints (2.10), the initial and boundary conditions (2.11), and the box constraints (2.12).

2.4. **Derivation of the Hessian and the Jacobian.** The constraints in the NLP given above can be divided into three classes: The vector $C(z)$ contains the mostly nonlinear equality constraints, whereas the linear boundary conditions and linear box constraints are found in the vectors $B(z)$ and $G(z)$, respectively. We calculate the dimension of the NLP for given discretisations fulfilling the CFL condition. For $M = 20$, $N = 600$, we find that the number of optimisation variables is $n_z = 28247$ and the number of equality constraints is $n_{eq} = 27650$. For $M = 50$, $N = 1500$, we have $n_z = 160607$ and $n_{eq} = 159110$. Thus n_z is greater than n_{eq}, which is important for the solvability of the formulated NLP, as the number of degrees of freedom is greater than zero. The algorithms used for nonlinear optimisation problems require first and second derivatives, i.e., the Jacobian of the constraints and the Hessian of the Lagrangian.

Jacobians $C'(z)$ and $B'(z)$. The derivation of the Jacobian of the nonlinear constraints, $C'(z)$, requires a structural approach. We note that the constraints in the vector $C(z)$ are sorted by time index j. The main part of the Jacobian is made up of sparse block matrices on its diagonal. Using the notation for $C(z)$ introduced above, the symbolic Jacobian

$$C'(z) = \left[\begin{array}{cccccc|cc} \frac{\partial C^0}{\partial q^0} & \frac{\partial C^0}{\partial q^1} & 0 & \dots & \dots & 0 & \frac{\partial C^0}{\partial u} & \frac{\partial C^0}{\partial T} \\ 0 & \frac{\partial C^1}{\partial q^1} & \frac{\partial C^1}{\partial q^2} & 0 & \dots & 0 & \frac{\partial C^1}{\partial u} & \frac{\partial C^1}{\partial T} \\ \vdots & 0 & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 & \frac{\partial C^{N-1}}{\partial q^{N-1}} & \frac{\partial C^{N-1}}{\partial q^N} & \frac{\partial C^{N-1}}{\partial u} & \frac{\partial C^{N-1}}{\partial T} \\ 0 & \dots & \dots & \dots & 0 & \frac{\partial H_T^N}{\partial q^N} & 0 & 0 \end{array}\right] \tag{2.13}$$

is obtained. The vertical line indicates the formal distinction between the state variables $(d_T, d_W, \dot{d}_T, \dot{d}_W, h, v)$ and the control variables (u, T) of the original optimal control problem. However, in the solution process of the NLP there is no difference between state and control variables, since all optimisation variables are contained in the vector z of optimisation variables. The quotient

$$\eta_J = \frac{\text{number of nonzero entries}}{\text{number of entries}} = \frac{N(19+16M)+12}{(N+1)(7+2M)(N(4+2M)+2)}$$

is a measure of the sparsity of the Jacobian. For $N = 600$, $M = 20$ we obtain $\eta_J \approx 0.000273 = 0.0273\%$.

Differentiating the sorted vector $B(z)$ with respect to z yields the symbolic Jacobian

$$B'(z) = \left[\begin{array}{ccccc|cc} \frac{\partial B^0}{\partial q^0} & 0 & \dots & \dots & 0 & 0 & 0 \\ 0 & \frac{\partial B^1}{\partial q^1} & 0 & \dots & 0 & 0 & 0 \\ \vdots & 0 & \ddots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 & \vdots & \vdots \\ 0 & 0 & \dots & 0 & \frac{\partial B^N}{\partial q^N} & 0 & 0 \end{array}\right],$$

where each matrix $\partial B^i / \partial q^i$, $i = 0, \dots, N$, has unit or zero vectors as columns.

Hessian $\nabla_{zz}L$. We assemble the Lagrange function

$$L(z,\lambda,\nu,\mu) = \tilde{J} + \sum_{k=1}^{N(4+2M)+2} \lambda_k C(z)_k + \sum_{k=1}^{2(M+N)+8} \nu_k B(z)_k + \sum_{k=1}^{2N(M+2)} \mu_k G(z)_k \tag{2.14}$$

of the NLP, where the constraints are coupled to the objective function by unknown numbers λ_k, ν_k, and μ_k, the so-called Lagrange multipliers. Differentiating L twice with respect to z yields non-zeros for the objective function and nonlinear constraints only. Thus the linear boundary conditions and box constraints $B(z) = 0$ and $G(z) \leq 0$ will not appear in the Hessian. The Hessian of the Lagrangian with respect to z

$$\nabla_{zz}L(z,\lambda,\mu) = \left[\begin{array}{cccc|cc} \nabla_{q^0,q^0}L & 0 & \dots & 0 & 0 & \nabla_{q^0,T}L \\ 0 & \nabla_{q^1,q^1}L & 0 & \vdots & \vdots & \nabla_{q^1,T}L \\ \vdots & 0 & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \nabla_{q^N,q^N}L & 0 & \nabla_{q^N,T}L \\ \hline 0 & 0 & \dots & 0 & \nabla_{u,u}L & \nabla_{u,T}L \\ \nabla_{T,q^0}L & \nabla_{T,q^1}L & \dots & \nabla_{T,q^N}L & \nabla_{T,u}L & 0 \end{array}\right], \tag{2.15}$$

referred to as H in what follows, is symmetric and exhibits a sparse structure with sparse blocks on its main diagonal and a dense last row and column vector. Similar to η_J we consider the quotient

$$\eta_H = \frac{\text{number of nonzero entries}}{\text{number of entries}} = \frac{N(8M+21)}{(N+1)^2(7+2M)^2} \tag{2.16}$$

as a measure of the sparsity of the Hessian. For $N = 600$, $M = 20$ we obtain $\eta_H \approx 0.000136 = 0.0136\%$.

The detailed sparsity structure and the values of C', B', and H are provided in [15, Appendix C]. We note that we work with a coordinate storage format for sparse matrices.

3. Model verification.

In order to verify the fully discretised model with the exact Jacobian, the formulation has been implemented in `SNOPT`, a general-purpose software package for large-scale nonlinear programming. It employs an SQP method, solving the QP subproblems with a reduced-Hessian active-set method. Thus, the exact Hessian does not have to be provided by the user. For further documentation, see [9].

Various numerical experiments on the NLP posed in Section 2.3 have been conducted with `SNOPT` to examine the sensitivity towards certain model parameters and the necessity of a regularisation strategy. Furthermore, an efficient method for the initialisation of calculations based on information from previous solutions is outlined below. A suitable initialisation has the potential to reduce computing times significantly.

3.1. Regularisation strategies.

When only the free end-time T is minimised, i.e., the objective function is $J = \alpha_0 T$, the computed control tends to be non-smooth and a so-called "bang-bang solution" is obtained. For the truck-container system performing a braking manoeuvre in minimum time, it is intuitive that the truck accelerates as long as possible and then brakes with the maximum possible force until it comes to a halt. The optimal solution resembles this intuitive behaviour,

yet the optimisation algorithm struggles to find an optimal control without frequent jumps, unless a regularisation term is added to the objective function. Two different regularisation approaches have been implemented and tested.

Let $\alpha_0 > 0$. Solving the discretised OCP for the modified objective function

$$\tilde{J} = \alpha_0 T + \frac{\alpha_2 T \Delta t}{2m_T^2} \sum_{j=0}^{N-1} (u^j)^2 + \frac{\alpha_4 T \Delta t}{2m_T^2} \sum_{j=0}^{N-2} \left(\frac{u^{j+1} - u^j}{\Delta t} \right)^2, \tag{3.1}$$

regularises the solution because the control effort is minimised in addition to the final time. When considering only $\alpha_2 \neq 0$, while $\alpha_4 = 0$, as proposed by Betts [1, Ch. 5.5], the regularisation strategy fails for poor initial guesses of T. It turns out that an alternative regularisation strategy that punishes jumps and oscillations in the control, i.e. with $\alpha_4 \neq 0$, is more robust. As we assume that $u \in L^\infty$, the derivative u' does not exist in general, thus additional assumptions concerning the control have to be made. Considering the control u as a state variable subject to the ODE-constraint $u' = \xi$, where $\xi \in L^\infty$ is the new control, yields that $u \in W^{1,\infty}$. Physically, the change in the control variable from u to ξ means controlling the jerk instead of the acceleration. It can be observed that this regularisation strategy yields more stable results, but there is still a slight dependency on the initial guess for T, though the oscillations in u are reduced. While neither of the proposed approaches is completely satisfactory, mixing the two strategies increases stability. If the parameters α_2 and α_4 are chosen in a way that the regularisation term is two or three orders of magnitude smaller than the actual objective term in the objective function, the influence of the smoothing terms on the optimal solution is negligible.

3.2. **Initialisation with the collocation method.** In order to obtain solutions with increased accuracy, the discretisation of the PDE and ODE must be refined. Both the number of variables and the number of nonzero entries in the Jacobian is $\mathcal{O}(NM)$. To reduce computing times, the collocation method can be employed to initialise a problem with a refined discretisation by interpolating a solution obtained on a coarse grid. Given a discrete solution on a coarse grid, the solution of the ODE and PDE is approximated by interpolating polynomials. Evaluating the resulting spline at the gridpoints of a refined grid yields a discrete starting vector for the refined problem. In general, the initialisation vector does not satisfy the constraints due to the interpolation error. Nevertheless, the optimisation algorithm needs fewer steps to find a feasible point, as the initial point is already close to feasibility. The solution for $N = 300$ has been utilised to initialise the problems with $N = 450$ and $N = 900$. We observe that a refined discretisation leads to more accurate results, which is indicated by decreasing values for T with increasing N. Computing times were reduced significantly by employing the collocation method. It was observed that only about a third of the computing time is required to find the solution when the problem is initialised by collocation.

4. **Implementation with `sqpfiltertoolbox`.** The software `sqpfiltertoolbox` is written in Fortran 90 and implements different sequential quadratic and sequential linear programming methods for general nonlinear optimisation problems. It was originally developed by M. Gerdts for small- to medium-scale problems with dense Jacobian and Hessian matrices [6]. Although there are more sophisticated, yet more complicated algorithms for large scale and sparse optimisation problems available (see e.g. [1, Ch. 2.3]), for the example presented in this paper, the SQP active-set method is employed, as it is implemented in the `sqpfiltertoolbox`. Nevertheless,

some adjustments to the algorithm must be made to exploit the sparsity and to tackle the challenges arising from the large scale of the problem. At the centre of interest is the solution of the KKT system in each QP iteration step. This section focuses on the comparison of different methods for the solution of the KKT systems and the adjustments that have to be made to ensure global convergence of the SQP method.

4.1. **Sequential quadratic programming.** For convenience, we briefly review the SQP method. To adjust to the common notation in nonlinear programming, in this subsection the vector of optimisation variables is referred to as $x \in \mathbb{R}^n$ and all equality and inequality constraints are comprised in the vector $c : \mathbb{R}^n \to \mathbb{R}^m$. The objective function is denoted by $f : \mathbb{R}^n \to \mathbb{R}$. It is assumed that the functions f and c are at least twice continuously differentiable with Lipschitz continuous second derivatives. The general formulation of a nonlinear program is now

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimise}} \quad & f(x) && \text{(4.1a)}\\ \text{subject to} \quad & c_i(x) = 0, \quad i \in \mathcal{R} && \text{(4.1b)}\\ & c_i(x) \le 0, \quad i \in \mathcal{S}, && \text{(4.1c)} \end{aligned}$$

where the index set $\mathcal{R}$ contains the indices of the equality constraints, while the indices of the inequality constraints are included in the set $\mathcal{S}$. At a given point x, the active set $\mathcal{A}(x)$ is the union of the set of equality constraints $\mathcal{R}$ with the indices of the active inequality constraints

$$\mathcal{A}(x) = \mathcal{R} \cup \{i \in \mathcal{S} \,|\, c_i(x) = 0\}. \tag{4.2}$$

The Lagrangian of the NLP (4.1) is

$$L(x, \lambda) = f(x) + \lambda^\top c(x),$$

where $\lambda \in \mathbb{R}^m$ comprises here all Lagrange multipliers. For the first-order optimality conditions to hold, a constraint qualification must be satisfied. For further considerations, the linear independence constraint qualification was assumed [13, Def. 12.1]. In this context, the Hessian $H(x, \lambda) = \nabla^2_{xx} L(x, \lambda)$ is well-defined.

The SQP method solves a sequence of quadratic models for the objective with linearised constraints

$$\begin{aligned} \underset{d_k\in\mathbb{R}^n}{\text{minimise}} \quad & \frac{1}{2} d_k^\top H_k d_k + \nabla f_k^\top d_k && \text{(4.3a)}\\ \text{subject to} \quad & \nabla c_i(x_k)^\top d_k + c_i(x_k) = 0, \quad i \in \mathcal{R} && \text{(4.3b)}\\ & \nabla c_i(x_k)^\top d_k + c_i(x_k) \le 0, \quad i \in \mathcal{S} && \text{(4.3c)} \end{aligned}$$

that approximate the NLP locally at (x_k, λ_k), where $H_k = H(x_k, \lambda_k)$ and $\nabla f_k = \nabla f(x_k)$. The QPs are solved by iterative methods, such as the active-set method. The SQP algorithm starts from an initial point (x_0, λ_0) and iteratively calculates KKT points (d_k, λ_{k+1}) for (4.3) updating $x_{k+1} = x_k + d_k$, until (x_k, λ_k) is a KKT point of the original NLP.

As the complementarity condition must be satisfied at an optimal solution (x^*, λ^*), the optimal Lagrange multipliers λ_i^* associated with the inactive constraints $c_i(x), i \notin \mathcal{A}(x^*)$ are zero. Given a solution x^* of (4.1) and assuming that (i) the Jacobian of the active constraints A^* at x^* has full rank; (ii) H^* is positive definite on the nullspace of A^*; and (iii) strict complementarity holds, there is a local solution of the QP (4.3) whose active set $\mathcal{A}_k$ is the same as the active set $\mathcal{A}(x^*)$ of the NLP

(4.1) at x^*, if (x_k, λ_k) is sufficiently close to (x^*, λ^*) [13, p. 533]. Thus, locally the QP can be treated as an equality constrained optimisation problem once the active set is known. Given this fact, the local SQP method is equivalent to Newton's method applied to the first-order optimality conditions of the NLP and quadratic convergence of the SQP method towards the solution can be achieved.

4.2. **Direct solution with exact second derivatives.** In Section 2.4, the exact first and second derivatives of the truck NLP are provided. Let $\mathcal{A}_k$ denote the active set at the current iteration step k. The Jacobian of G associated with the active set is referred to as $G'_{\mathcal{A}_k}$. Having all derivatives available, the KKT system can be assembled as follows

$$\begin{bmatrix} H & B'^\top & G'^\top_{\mathcal{A}_k} & C'^\top \\ B' & 0 & 0 & 0 \\ G'_{\mathcal{A}_k} & 0 & 0 & 0 \\ C' & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda_B \\ \lambda_G \\ \lambda_C \end{bmatrix} = \begin{bmatrix} -\nabla f \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{4.4}$$

where $H \in \mathbb{R}^{n\times n}$ is the exact Hessian and $\nabla f \in \mathbb{R}^n$ denotes the gradient of the Lagrangian of the QP. To improve readability, the index k, indicating the evaluation of functions and derivatives at the point (x_k, λ_k), has been omitted.

As we do not have to distinguish between the different Jacobians B', $G'_{\mathcal{A}_k}$ and C' for the further examination of the KKT system, we introduce the notation

$$\begin{bmatrix} H & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f \\ 0 \end{bmatrix} \tag{4.5}$$

and refer to $A \in \mathbb{R}^{m\times n}$ as the Jacobian of the active constraints. The solution of the system (4.5) poses two major challenges. Firstly, the KKT matrix is large and sparse, which calls for a sophisticated and efficient solver for linear systems of equations (LES), exploiting the sparsity. Secondly, the solvability of the system must be guaranteed, i.e., certain criteria must be met, such that the KKT system has a unique solution.

For the first challenge, open-source software for the solution of large, sparse and symmetric systems of equations is available. The code `MA57` [4] uses the so-called multifrontal method, which is based on a sparse variant of Gaussian elimination. Solving the specific KKT system presented above with `MA57`, revealed two aspects that have a major influence on efficiency and stability of the algorithm:

(i) The control variables u_i are of the order of magnitude 10^4. As sums of u_i^2 appear in the objective function of the NLP (2.3), the parameters α_2 and α_4 have to be chosen very small ($\sim 10^{-7}$) to balance the control terms against the other terms in the objective function. Terms of the form α_2/m_T^2 and α_4/m_T^2 appear in the Hessian, causing the respective Hessian values to be very small ($\sim 10^{-13}$) and sometimes even close to machine precision. The issues with round-off errors are overcome by a suitable scaling, e.g., by working with the scaled acceleration $\hat{u} = u/m_T$.

(ii) As argued in [4, Ch. 6.3] the pivoting parameter `threshold` has a decisive effect on the performance of `MA57`. Smaller values for `threshold` give a pivot based on sparsity considerations and reduce the required computing time. However, this can lead to an unstable factorisation. To resolve the trade-off between computing time and stability, a suitable value is `threshold` $= 10^{-2}$.

The KKT system (4.5) is not always solvable, but a necessary condition for the KKT matrix to be non-singular is that A has full row rank and the reduced

Hessian $Z^\top HZ$ is positive definite, where Z denotes the matrix whose columns are a basis for the null space of A. Numerical tests showed that neither of the two requirements are met in general, leading to a breakdown of the QP active-set method. In the following, two modifications of the KKT matrix are proposed that ensure the solvability of the KKT system.

Dual regularisation. In order to determine the row rank of A, being the Jacobian of the active constraints, a singular value decomposition can be performed. Given that A has more columns than rows it has full row rank, if the number of singular values $\sigma_i \neq 0$ is equal to the number of rows. An examination of the singular values reveals that A has full row rank at the solution z^*. If A has more rows than columns, which can happen, for example, for a bad initial guess with too many active box constraints, full row rank of the Jacobian cannot be achieved and the KKT matrix is singular.

By means of dual regularisation, a non-singular KKT matrix can be obtained even if A is rank deficient. The KKT system is altered to

$$\begin{bmatrix} H & A^\top \\ A & -\mu I \end{bmatrix} \begin{bmatrix} d \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f \\ 0 \end{bmatrix}, \tag{4.6}$$

where $\mu > 0$ is a parameter and I the identity. Introducing $-\mu I$ on the (2,2)-block of the KKT matrix corresponds to solving a QP problem with shifted constraints. As the KKT system is perturbed this way, the obtained solution is only an approximate solution of the original problem. However, if μ is sufficiently small, the deviation of the approximate solution is negligible.

For the KKT system of the specific problem under consideration, the invasive influence of the proposed technique could be reduced by introducing dual regularisation on a sub-block only. It has been found that it is sufficient to modify the KKT system as

$$\begin{bmatrix} H & B'^\top & G'^\top_{\mathcal{A}_k} & C'^\top \\ B' & 0 & 0 & 0 \\ G'_{\mathcal{A}_k} & 0 & 0 & 0 \\ C' & 0 & 0 & -\mu I \end{bmatrix} \begin{bmatrix} d \\ \lambda_B \\ \lambda_G \\ \lambda_C \end{bmatrix} = \begin{bmatrix} -\nabla f \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{4.7}$$

with regularisation in the (4,4)-block only. Numerical tests indicate that choosing $\mu = 10^{-12}$ yields a stable configuration. Even when in the neighbourhood of the solution, where regularisation would not necessarily be needed for the KKT matrix to be non-singular, regularisation is still useful, as it speeds up the solution process with `MA57`.

Primal regularisation. When using exact second derivatives instead of a BFGS approximation, neither the Hessian H nor the reduced Hessian $Z^\top HZ$ can be guaranteed to be positive definite. By means of primal regularisation, positive definiteness of the reduced Hessian can be achieved. In the following, a method for primal regularisation is outlined. We refer to the inertia of the symmetric matrix K as the triplet giving the numbers of positive, negative and zero eigenvalues of K: $\mathrm{In}(K) = (n_+, n_-, n_0)$. By repeatedly applying Sylvester's law of inertia, we may show that if the reduced Hessian $Z^\top HZ$ is positive definite and A has full row rank m, then $\mathrm{In}(K) = (n, m, 0)$. In case of dual regularisation the KKT matrix K changes to the form in (4.6) with a nonzero (2,2)-block μI, for which a similar result is obtained, see [5, Ch. 4, Proposition 2]. If $\mathrm{In}(K) = (n, m, 0)$ the local model

of the optimisation problem is convex and the KKT system (4.6) has a unique solution. However, if $\mathrm{In}(K) \neq (n, m, 0)$, the local model of the optimisation problem is non-convex and the QP algorithm will fail. In this case, the local model must be convexified, which is referred to as inertia control in the following.

In order to determine whether or not inertia control must be performed, the inertia of the KKT system must be computed. Fortunately the desired information is a byproduct of the solution process with `MA57`. The KKT matrix is factorised by $PKP^\top = LDL^\top$ with the block diagonal matrix D. The inertia of D can easily be determined. As a permutation with the matrix P does not change the eigenvalues of K and L is non-singular, Sylvester's law of inertia can be exploited, leading to the result that $\mathrm{In}(K) = \mathrm{In}(D)$. Thus, the inertia of the KKT matrix is available in each QP iteration step and it can be checked whether K has the required inertia.

In case the inertia of the KKT matrix is not equal to $(n, m, 0)$ in a QP iteration step, inertia control is invoked. For inertia control the Levenberg method [1, Ch. 2.5] with a modification to fit our specific application is used. Whenever inertia control is invoked, the current QP is discarded and a new QP with the shifted Hessian is set up and solved. The new search direction d_k that solves the modified QP is significantly biased toward a gradient direction and convergence is degraded [1, Ch. 2.5].

Numerical tests on the specific truck NLP showed that the inertia of the KKT matrix has the desired properties in the neighbourhood of the solution, which implies that the exact Hessian is accepted in the QP. Inertia control with the shifted Hessian is invoked far from the solution only and local quadratic convergence to the solution is still achieved.

4.3. **Solution with the range-space method.** The direct solution with exact second derivatives described above turned out to be reliable and efficient. Nevertheless, there are some difficulties that motivate a different approach to the solution of the KKT system. On the one hand, methods that solve the LES directly, such as the method involving `MA57` presented above, might fail for very large problems because of a lack of available storage. On the other hand, deriving the exact Hessian analytically for more complicated problems, e.g., for a 3D PDE, can be very time-consuming and prone to errors. The method presented in the following circumvents these disadvantages by approximating the Hessian with a Limited-Memory-BFGS (LM-BFGS) matrix and solving the resulting KKT system with the iterative conjugate gradient method, which allows for the solution of larger systems.

In [13, Ch. 16] the range-space method for the solution of the KKT system

$$\begin{bmatrix} W & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} d \\ \lambda \end{bmatrix} = \begin{bmatrix} -\nabla f \\ 0 \end{bmatrix}, \tag{4.8}$$

which is based on block-elimination, is described. Note that the exact Hessian H was replaced by an approximate Hessian W. By solving the first equation for

$$d = -W^{-1}(\nabla f + A^\top \lambda) \tag{4.9}$$

and substituting in the second equation, we obtain a system of equations for the multipliers

$$\tilde{B}\lambda = \tilde{b} \tag{4.10}$$

with the right-hand side $\tilde{b} = -AW^{-1}\nabla f$. Here the matrix $\tilde{B} := AW^{-1}A^\top$ is symmetric and positive definite if W is symmetric and positive definite, which is the case for an LM-BFGS approximation of the Hessian W. Thus (4.10) can be solved iteratively with the conjugate gradient (CG) method. Once λ is obtained,

the search direction d can be computed by inserting λ in equation (4.9). Before the LM-BFGS and CG methods are outlined, it must be mentioned that the range-space method as presented above is well-defined for non-singular KKT systems only. If the number of constraints is larger than the number of variables, the Jacobian A is rank deficient and the KKT system is not solvable, leading to a breakdown of the proposed method. As mentioned in the previous subsection, non-singularity can be achieved by means of dual regularisation. Nevertheless, performing block-elimination with a regularised KKT matrix leads to an ill-conditioned system of linear equations. As this implies numerical difficulties, dual regularisation should not be the method of choice.

4.3.1. *The Limited-Memory-BFGS method.* The exact Hessian can be approximated by a quasi-Newton matrix, which is referred to as a BFGS matrix. By the way these matrices are constructed, they exhibit a dense structure even though the exact Hessian might be sparse. In [3], a limited memory quasi-Newton method is described that uses a less memory-intensive representation of the BFGS matrix. To avoid any confusion concerning nomenclature, it must be stated that the term LM-BFGS method refers to an entire algorithm for the solution of large-scale unconstrained optimisation problems. As the optimisation problem examined in this article involves nonlinear constraints, the LM-BFGS method can not be applied as a whole. Still, the representation of the quasi-Newton approximation for the Hessian, which is a crucial component in the LM-BFGS method, can be integrated in the range-space method presented above. In particular, matrix-vector multiplications of the form Wp and $W^{-1}p$, where W denotes the LM-BFGS Hessian and p is an arbitrary vector, can be performed efficiently.

The main difference between the BFGS and the LM-BFGS matrix representation is the storage of the Hessian W_k in the k-th SQP iteration. Whereas a dense $n \times n$ matrix is stored in the BFGS method, only a set of the l most recent update vector pairs $\{s_i, y_i\}, i = k-l, \ldots, k-1$ is stored, which defines the Hessian implicitly. The update vector pairs are constructed as $s_i := x_{i+1} - x_i$ and $y_i = g_{i+1} - g_i$, where $g_i = \nabla f(x_i)$, in line with the standard notation. After a new iterate has been computed, the new update vector pair is either added to the set if $k \leq l$, or the oldest update vector pair is replaced by the newest if $k > l$. Thus, in the first l steps, the LM-BFGS method resembles the classical BFGS method. For practical purposes it is often sufficient to choose $l \in [3, 7]$, which implies that only $(2l+1)n$ instead of n^2 elements must be stored to reconstruct the approximate Hessian.

Auxiliary matrices must be defined and stored for the representation of the quasi-Newton matrix and its inverse. The storage and computational effort required to obtain these matrices is negligible, especially because they have to be computed once for each SQP-iteration only. If the initial matrix is of the form $W_0 = \sigma_k I$, the approximate Hessian is represented by

$$
\begin{aligned}
W_k &= \sigma_k I - \begin{bmatrix} \sigma_k S_k & Y_k \end{bmatrix} \begin{bmatrix} \sigma_k S_k^\top S_k & L_k \\ L_k^\top & -D_k \end{bmatrix}^{-1} \begin{bmatrix} \sigma_k S_k^\top \\ Y_k^\top \end{bmatrix}, \text{with} \\
S_k &= [s_{k-l}, \ldots, s_{k-1}] \quad \in \mathbb{R}^{n \times l}, \\
Y_k &= [y_{k-l}, \ldots, y_{k-1}] \quad \in \mathbb{R}^{n \times l}, \\
(L_k)_{i,j} &= \begin{cases} s_{k-l-1+i}^\top y_{k-l-1+j}\,; & \text{if } i > j, \\ 0 \qquad\qquad\quad\;\; ; & \text{otherwise,} \end{cases} \qquad i, j = 1, \ldots m, \\
D_k &= \operatorname{diag}[s_{k-l}^\top y_{k-l}, \ldots, s_{k-1}^\top y_{k-1}] \quad \in \mathbb{R}^{l \times l},
\end{aligned}
$$

where σ_k is a positive scalar that can either be fixed or updated in the course of the iterations based on curvature information. The choice of σ_k will be addressed in the subsection on restarting the BFGS method. In a similar manner, the inverse of the LM-BFGS matrix is given by

$$W_k^{-1} = \gamma_k I - \begin{bmatrix} S_k & \gamma_k Y_k \end{bmatrix} \begin{bmatrix} R_k^{-\top}(D_k + \gamma_k Y_k^\top Y_k) R_k^{-1} & -R_k^{-\top} \\ -R_k^{-1} & 0 \end{bmatrix} \begin{bmatrix} S_k^\top \\ \gamma_k Y_k^\top \end{bmatrix},$$

$$\text{with } (R_k)_{i,j} = \begin{cases} s_{k-l-1+i}^\top y_{k-l-1+j} \,; & \text{if } i \le j, \\ 0 \,; & \text{otherwise,} \end{cases} \qquad i,j = 1,\dots,m,$$

and with S_k, Y_k and D_k as defined above and the scalar γ_k. Although there are efficient algorithms for matrix multiplications involving the LM-BFGS Hessian and its inverse, only matrix-vector multiplications are required in the context of the range-space method and the iterative solution of the resulting LES. For an efficient algorithm for the computation of the product $W_k p$ see [3, Ch. 3.2].

According to (4.12) the computation of the product $W_k^{-1} p$ involves multiplications with the inverse of the matrix R_k. Since l is small and R_k is triangular, the inverse R_k^{-1} can be provided with little computational effort. Furthermore, the inverse needs only be computed once per SQP-iteration. In total, $(4l+1)n + (5/2)l^2$ multiplications are required to obtain the matrix-vector product.

In the context of the range-space method, a linear system of equations of the form (4.10) must be solved. The matrix $\tilde{B}$ is never computed and stored explicitly. The matrix A is stored in the coordinate storage format, whereas the matrix W^{-1} is represented as an LM-BFGS matrix. Hence, direct methods, which would require the explicit representation of the matrix $\tilde{B}$, are not suitable.

The main effort of a single CG iteration lies in the computation of the matrix vector product $\tilde{B}p_k$. In the context of the range-space method this product has the form $AW^{-1}A^\top p_k$. As mentioned above, the matrix $\tilde{B}$ is not available in its explicit form. Thus, the product $\tilde{B}p_k$ involves three consecutive matrix-vector products. For a detailed convergence analysis of the CG method and the essential preconditioning, see [13, pp. 112-119].

Although preconditioning can speed up convergence significantly, the overall computing time is not necessarily reduced, as computational effort is required to obtain a preconditioning matrix and to perform arithmetic operations with its inverse. Therefore, the choice of a suitable preconditioning matrix is problem dependent. In the context of the range-space method with a quasi-Newton approximation of the Hessian, the coefficient matrix $\tilde{B}$ is not explicitly known. In this context, the SSOR and the Cholesky preconditioner are not suitable. Thus, the Jacobi preconditioner has been chosen for further considerations. Detailed information on the efficient computation of the preconditioner is given in the following subsection.

4.3.2. *Implementational details.* By slightly modifying the presented algorithms above, i.e., the range-space method with an LM-BFGS approximation solved by the CG method, and taking advantage of the characteristic structure of the QP subproblem, the computational effort can be reduced substantially.

Preconditioning the CG method: For computational reasons, the diagonal of the coefficient matrix $\tilde{B} = AW^{-1}A^\top$ is the preconditioner of choice for further considerations. Nevertheless, the overall computing time increases, if the preconditioner is recalculated in each QP iteration. However, by exploiting the structure of the

QP iterations, the diagonal preconditioning matrix $\tilde{M}$ can be obtained efficiently, reducing the overall computing time. In the first QP iteration, $\tilde{M}$ must be computed entirely, but can be updated in the subsequent iterations with very little computational effort. The i-th diagonal element of the preconditioning matrix $\tilde{m}_{ii}$ is obtained by pre- and post-multiplying the coefficient matrix with the i-th unit vector e_i

$$\tilde{m}_{ii} = e_i^\top A_j W^{-1} A_j^\top e_i,$$

which involves three matrix-vector products. The index j denotes the QP iteration. Depending on the result of the j-th QP iteration, three cases must be distinguished. (i) If the computed stepsize is accepted, the Jacobian of the active constraints does not change in the $(j+1)$-th iteration, which implies that $A_{j+1} = A_j$. Therefore the preconditioner can be reused. (ii) In case the constraint with index i is deactivated, A_{j+1} is obtained by deleting the i-th row in A_j. Instead of recomputing the preconditioner, it can be updated without any computational effort by deleting the i-th row. (iii) This possible case involves the activation of a constraint with index i, which implies that A_{j+1} results from A_j by adding a row at the i-th position. The preconditioner $\tilde{M}$ is updated by adding a row and column at the respective position, with the diagonal element $\tilde{m}_{ii} = e_i^\top A_{j+1} W^{-1} A_{j+1}^\top e_i$.

If these update formulae are considered, the main effort of computing the preconditioner lies in the first QP iteration, where the preconditioner must be determined from scratch. The effort required to update the preconditioner is negligible. The numerical results show that preconditioning with updates instead of recomputation reduces the number of required CG iterations by about 15%.

Initialising the CG method: If no information on the solution of the LES is available, the CG method is initialised with $\lambda_0 = 0$. However, after a deactivation step in the j-th QP iteration, the initial point can be chosen so that only very few CG iterations are necessary to achieve convergence. As the QP stepsize is zero in a deactivation step, the right-hand side of the KKT system (4.8) does not change in the following iteration. If the i-th constraint was deactivated, the Jacobian of the active constraints is updated by deleting the i-th row. Reconsidering the LES that is to be solved by the CG method

$$AW^{-1}A^\top \lambda = -AW^{-1}\nabla f, \tag{4.13}$$

the following observation can be made: as the Jacobian of the active constraints is sparse and exhibits a block structure, the deletion of a row has little impact on the structure of the system above. The solution of the system changes slightly, as only a few multipliers in λ are impacted by the change. If we initialise $\lambda_0 = \lambda \setminus \{\lambda_i\}$ in the $(j+1)$-th QP step where λ are the multipliers of the j-th QP step in which the constraint with index i was deactivated, the convergence rate of the CG method can be increased significantly. In several calculations it has been observed that only one CG iteration is needed after a deactivation step. Although this result cannot be generalised for other problems, in the context of the NLP considered in this study, which has linear box constraints as inequality constraints only, the careful choice of an initial point λ_0 speeds up the CG method. The numerical results reveal that the average number of CG iterations needed for each QP drops by about 15% when the initial point is chosen as described above. Nevertheless, numerical difficulties were encountered more frequently for initial guesses far from the solution and the overall algorithm turned out to be less robust. This behaviour might be unexpected at first, as a different initial point in the CG method should not change the solution.

Bearing in mind the fact that the CG method is an iterative method that terminates when the residual is smaller than a certain tolerance `CGTOL`, it is not surprising that different initial guesses lead to slightly different solutions in the iteration process. Even if the residual has machine precision, it cannot be concluded that the solution has the same precision. A slightly different solution in each QP might lead to activation or deactivation of different constraints and thus impact the overall SQP procedure. Therefore, the initialisation with the multipliers of the previous iteration may not have a positive effect. All calculations with the range-space method in the following subsection on numerical results were performed with an uninitialised CG method, i.e., $\lambda_0 = 0$. Stable results were obtained with $\texttt{CGTOL} = 10^{-14}$, which was chosen as the standard value for all calculations performed with the range-space method.

Tuning the LM-BFGS method: There are several adjustments to the LM-BFGS method to speed up global convergence of the SQP algorithm. Three modifications in particular are addressed in the following.

(i) When the initial guess is far away from the solution, it is often useful to perform a certain number of steepest-descent steps. In this case the quasi-Newton Hessian is replaced by the identity matrix and arithmetic operations involving the Hessian and its inverse require less computational effort.

(ii) As outlined in Subsection 4.3, a certain number m of vector pairs is stored to reconstruct the LM-BFGS Hessian approximation. If the number of SQP-iterations exceeds m, the newest vector pair replaces the oldest one. Although "old" curvature information is discarded this way, the convergence rate can be improved by deleting all update vectors and restarting the LM-BFGS method with a scaled identity matrix. In [12, Ch. 10] Luenberger and Ye propose to restart after a fixed amount of iteration steps. However, it is also possible to introduce criteria that trigger a restart. For example, the reduction in the constraint violation or in the gradient of the Lagrangian can be measured and compared to the previous iterations. If either of the two criteria

$$\max(0, |c(x_k)|) \leq \alpha \min_{j=1,\dots,r} (\max(0, |c(x_{k-j})|)),$$
$$||\nabla_x L(x_k, \lambda_k)|| \leq \alpha \min_{j=1,\dots,r} ||\nabla_x L(x_{k-j}, \lambda_{k-j})||$$

with a variable integer r and $0 < \alpha < 1$ (e.g. $\alpha = 0.99$), is violated after the k-th iteration, the LM-BFGS method is restarted. Numerical experiments indicate that choosing r as the number of iterations since the last restart yields satisfactory results. On average, the restart is triggered after 3-4 iteration steps.

It turns out that performing steepest-descent steps before starting the LM-BFGS Hessian approximation often has a detrimental effect, as computing times increase in most cases. However, restarting the LM-BFGS method has a beneficial impact. Both a restart after a fixed amount of iteration steps and the automated restart reduce computing times significantly, where the automatic restart is superior to the fixed restart [15, Table B.7].

(iii) As proposed in [12, Ch. 10] and [11, Ch. 4], the performance of the LM-BFGS method can be improved substantially by means of self-scaling. After a restart in SQP-iteration k, the quasi-Newton matrix is initialised with a scaled identity $H_0 = \sigma_k I$ with

$$\sigma_k = \frac{y_k^\top s_k}{s_k^\top s_k},$$

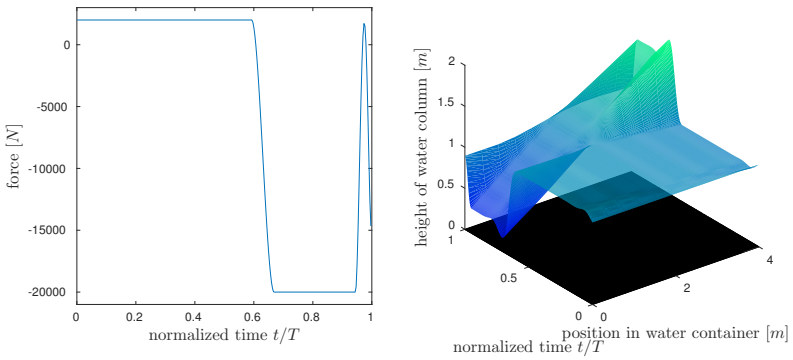

FIGURE 2. Optimal solution: control force $u(t)$ vs. time (left) and fluid column height $h(t,x)+b(x)$ (right) for $N = 300$, $M = 10$ (final time minimised, $T = 10.8s$) and a straight fluid tank bottom.

where $[s_k, y_k]$ is the latest update vector pair [3, p. 142]. Luenberger, Ye and Liu propose scaling the inverse Hessian $H_0^{-1} = \gamma_k I$ with

$$\gamma_k = \frac{y_k^\top s_k}{y_k^\top y_k}.$$

Since $\gamma_k \neq 1/\sigma_k$, the two methods are not equivalent. Both scaling variants have been implemented and tested, where self-scaling with σ_k turned out to be more efficient. In numerical experiments, a decisive beneficial impact of self-scaling was revealed, which has the potential of reducing computing times by about two thirds.

5. **Numerical results and evaluation.** All numerical experiments were performed on a machine with 2 Hexa core Intel Xeon CPU X5680s (-HT-MCP-SMP-) with 3.33 GHz each.

In Figures 2 and 3 results for different manoeuvres of the truck-container system are shown. In the first case, the truck performs a braking manoeuvre over a distance of 100m, where only the final time is minimised (Problem (1)), whereas in the second case, the fluid-level deviation is also minimised (Problem (2)). Figure 4 shows the results for Problem (1) with a non-straight fluid tank bottom. In all three cases, the control looks sufficiently smooth, with the undesired oscillations being damped by the smoothing terms. Before discussing the performance of the two different solution techniques examined in this paper, we observe that the solution process for Problem (1) takes significantly more time than for Problem (2), see [15, Tables B.2, B.3, B.7 – B.9]. Comparing the controls in Figures 2 and 3 for the different problems, reveals that most of the control variables in (1) are active, i.e., at their lower or upper bounds, whereas none of the control variables are active in Problem (2). It can be concluded that a significant share of computational effort is connected to the determination of the active set.

As the numerical results in [15, Appendix B] show, the two different solution strategies examined in this study perform differently. The direct Method (A) involving the solution of the KKT systems with `MA57` is clearly superior to the range-space Method (B) with the quasi-Newton approximation of the Hessian and the iterative solution of the linear systems of equations. Not only does Method (A) require only a fraction of the computing time of Method (B), but also higher solution

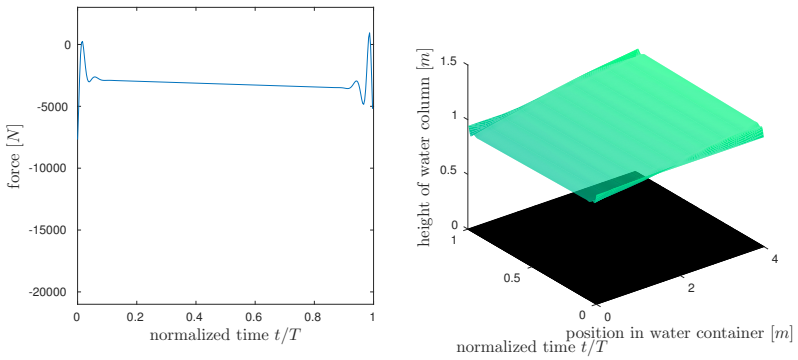

FIGURE 3. Optimal solution: control force $u(t)$ (left) and fluid column height $h(t,x)+b(x)$ (right) for $N = 300$, $M = 10$ (final time and fluid-level deviation minimised, $T = 19.1s$) and a straight fluid tank bottom.

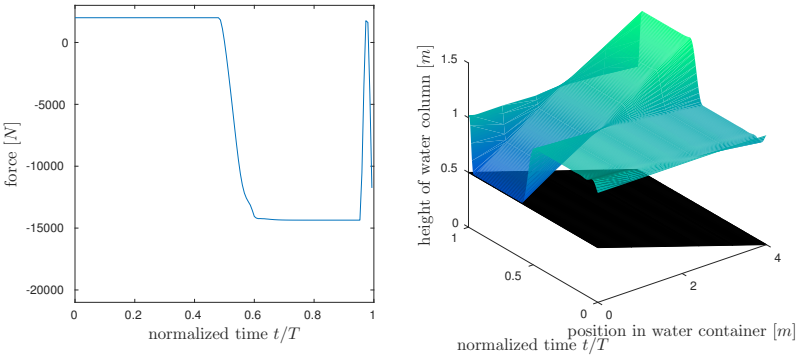

FIGURE 4. Optimal solution: control force $u(t)$ (left) and fluid column height $h(t,x)+b(x)$ (right) for $N = 150$, $M = 5$ (final time minimised, $T = 11.5s$). The bottom of the fluid tank is given by the linear function $b(x) = -x/8 + 1/2$, $0 \le x \le L = 4$.

accuracies can be obtained with Method (A). Furthermore, Method (A) succeeds in solving problems (1) and (2) with a great range of initial guesses and scaled objective functions [15, Tables B.2 – B.5]. For Problem (1) Method (B) can cope with a variety of initial guesses, too, yet the algorithm struggles to find feasible points in some cases for Problem (2). For the coarsest discretisation $N = 150$, $M = 5$ the algorithm breaks down without finding a feasible point. "Hot starting" (HS) the problem with the current iterate as the next initial guess, fixes the feasibility problems, and a solution is attained at the cost of increased computing times. Furthermore, Method (B) solves Problem (2) for finer discretisations much faster, when initialised by collocation. However, Method (B) is more sensitive to scaling in the objective function, sometimes leading to a breakdown of the method or yielding inaccurate solutions [15, Table B.10]. It can be concluded that Method (A) is more robust than Method (B), for which numerical difficulties concerning scaling and different initial guesses emerged more often.

ITER	QPIT	ALPHA	OBJ	CV	KKT
0	0	0.0000E+00	0.100000050000E+01	0.9893E+02	0.1000E+00
1	224	0.1000E+01	0.102710574197E+01	0.1811E+02	0.2648E-01
2	6	0.1000E+01	0.107819921922E+01	0.2086E+00	0.5244E-02
3	24	0.1000E+01	0.109015429623E+01	0.4723E-02	0.2969E-03
4	5	0.1000E+01	0.109049950972E+01	0.3198E-05	0.4477E-06
5	3	0.1000E+01	0.109049961114E+01	0.4253E-11	0.4570E-12

END OF SQP METHOD

TABLE 1. Solution output of Problem (1) with $M = 5$ and $N = 150$. CV and KKT denote the norm of the constraint violation and the norm of the gradient of the Lagrangian, respectively. QPIT is the number of QP iterations and OBJ the value of the objective function.

Moreover, due to the use of exact second derivatives, quadratic convergence to the solution can be achieved with Method (A), whereas Method (B) does not exhibit that property. In Table 1 an excerpt of the solution output of Problem (1), exhibiting the quadratic convergence, is depicted.

We conclude that, if exact second derivatives are available, Method (A) is the method of choice for the examined application as it is faster and more robust. Yet, extending the truck-container model to three dimensions would make the derivation of the exact Hessian cumbersome and prone to errors. In this case, Method (B) could be employed, which does not need exact second derivatives, at the cost of significantly higher computing times. The introduction of Method (B) as an alternative to (A) was also motivated by the assumption that the application of a direct method may no longer be viable for very large problems because of memory limitations of the LES solver. It was shown that Method (A) solved problems with up to 104487 variables on a grid with $N = 1200$ and $M = 40$ within about 6.2 hours, see Table 2. Based on the observation that Method (B) requires about a fifty times this computing time, computation times of more than 12 days for the solution of the same problem would be required. For even larger problems, the solution with Method (B) is unlikely to yield a result in an acceptable time.

Finally, we compare the performance of Method (A) with SNOPT. For these runs, a sequence of problems of type (1) is set up with gradually refined discretisation. Each problem except the first is initialised using collocation based on the previous solution on a coarser grid. Despite the fact that SNOPT does not rely on exact second derivative information, it requires much less computing time than Method (A) in most cases, as it can be observed in Table 2. However, the problem with $N = 1200$ and $M = 40$, which has 104487 variables, is solved much faster by Method (A). This can be ascribed to the superior matrix factoring technique for very large and sparse matrices employed by MA57.

6. Conclusion and future work.

In this article, an optimal control problem for an example of a fully coupled mechanical system, i.e., the truck-container system, was solved. The infinite-dimensional optimal control problem was fully discretised,

problem data:					
FEASTOL =	1.00E-008	α_0 =	1.00E-001		
OPTTOL =	1.00E-008	α_1 =	0		
μ =	1.00E-012	α_2 =	1.00E-007		
threshold =	1.00E-002	α_3 =	1.00E+000		
initialisation by collocation		α_4 =	1.00E-007		
		c_0 =	1.00E-003		

N	M	$\tilde{J}$	SQPIT	QPIT	$t[s]$	$t[s]$ SNOPT
150	5	1.0905	5	263	10.4	1.6
300	10	1.0870	3	47	19.4	7.8
450	15	1.0840	3	49	91.6	25.4
600	20	1.0810	3	49	281.9	99.0
900	30	1.0751	4	110	2824.7	1541.0
1200	40	1.0705	4	289	22333.2	162545.6

TABLE 2. Sequence of problems with gradually refined discretisations solved with `sqpfiltertoolbox` Method (A) and `SNOPT`. `SQPIT` states the number of SQP iterations.

where suitable integration schemes were employed for the discretisation of the coupled system of ordinary and partial differential equations that describe the motion of the truck, container, and fluid.

The finite-dimensional discretised optimal control problem, which is equivalent to an optimisation problem with nonlinear constraints, was solved with an SQP active-set method, as implemented in the software framework `sqpfiltertoolbox`. The article focused on the solution of the linear systems of equations, the so-called KKT systems, which arise in each QP iteration step. Due to the characteristic structure of a discretised optimal control problem with differential equations as constraints, these LES are large and sparse, which calls for solvers that exploit their structure for efficiency.

Two solution techniques were implemented and examined in the context of this study. Method (A) relies on the direct solution of the KKT systems, where both the Hessian of the Lagrangian and the Jacobian of the constraints were derived analytically. The resulting KKT systems were solved with `MA57`, a solver for symmetric large and sparse systems. Both primal and dual regularisation techniques were implemented that ensure the convergence of the method for a variety of initial guesses. In Method (B) the solution of the KKT system was split up in the solution of two smaller dimensional linear systems via block-elimination. The exact Hessian was replaced by a positive-definite Limited-Memory-BFGS approximation and the resulting linear systems were solved with the iterative conjugate gradient method. An efficient preconditioning method that takes advantage of the procedure of the QP active-set strategy was developed. Furthermore, several restarting and scaling techniques for the Limited-Memory-BFGS matrix representation were implemented and tested, which speed up the overall solution process significantly.

Numerical experiments showed the clear superiority of Method (A) in many respects. Not only did Method (A) solve the test problems about a fifty times faster than Method (B), but also found solutions with better accuracy. In some

cases, Method (B) struggled to find a feasible point and numerical difficulties were encountered for certain initial guesses and badly scaled objective functions. Nevertheless, Method (B) has the advantage of not being dependent on the availability of exact second derivatives. If the considered model of the mechanical system were extended to three dimensions and the Saint-Venant equations to two dimensions, the analytical derivation of the exact derivatives might not be desirable. Consequently, Method (B) is still relevant for applications where the exact Hessian is not available.

Method (A) outperforms the FDTO approach described in [7], yielding up to several thousand SQP iterations and computing times of about a day for $M = 20$. While the FOTD approach in [10] exhibits computing times of several hours for the same grid, it is limited to fixed terminal times. Method (A) exhibits computing times of a couple of minutes for the same grid and requires only up to 5 SQP iterations.

For further improvement of the overall solution process with Method (A), the efficient determination of the active set, which consumes a significant share of the computational effort, could serve as an approach. As proposed, e.g. in [1, Ch. 2.3] or [8], a sequence of similar LES, as they arise in the QP iterations, can be solved efficiently by means of the Schur-complement method. As a further possible improvement of our method, we could start with BFGS updates of the identity matrix as approximation to the Hessian, guaranteeing a positive definite matrix, and once the iterate is close to the solution, we switch to the exact Hessian. One way of reducing the number of state variables is to use a non-equidistant coarse grid, with a refinement only at the left and at the right end of the container.

Acknowledgements. J.-H. Webert would like to thank the University of California San Diego and especially Prof. Philip Gill for integrating him into the faculty and for establishing a creative working atmosphere during his stay.

References

[1] J. T. Betts, *Practical Methods for Optimal Control Using Nonlinear Programming*, 2^{nd} edition, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2010.

[2] M. Breuß, The correct use of the Lax–Friedrichs method, *ESAIM: Mathematical Modelling and Numerical Analysis*, **38** (2004), 519–540.

[3] R. Byrd, J. Nocedal and R. Schnabel, Representations of quasi-Newton matrices and their use in limited memory methods, *Mathematical Programming*, **63** (1994), 129–156.

[4] I. S. Duff, MA57—A code for the solution of sparse symmetric definite and indefinite systems, *ACM Trans. Math. Softw.*, **30** (2004), 118–144.

[5] A. Forsgren, Inertia-controlling factorizations for optimization algorithms, *Applied Numerical Mathematics*, **43** (2002), 91–107. 19th Dundee Biennial Conference on Numerical Analysis.

[6] M. Gerdts, *SQP filtertoolbox within software package OCPID-DAE1, Optimal Control and Parameter Identification with Differential-Algebraic Equations of Index 1. Users Guide (Online Documentation).* Universität der Bundeswehr München, Neubiberg/München, 2010. (Available from: `http://www.optimal-control.de/index.php/software`.)

[7] M. Gerdts and S.-J. Kimmerle, Numerical optimal control of a coupled ODE-PDE model of a truck with a fluid basin, *Discrete Contin. Dynam. Systems - A*, **2015, Suppl.** (2015), 515—524.

[8] P. E. Gill and E. Wong, Methods for convex and general quadratic programming, *Math. Prog. Comp.*, **7** (2015), 71–112.

[9] P. Gill, E. Wong, W. Murray and M. Saunders, *User's Guide for SNOPT Version 7.4: Software for Large-Scale Nonlinear Programming*, 2015.

[10] S.-J. Kimmerle and M. Gerdts, Necessary optimality conditions and a semi-smooth Newton approach for an optimal control problem of a coupled system of Saint-Venant equations and ordinary differential equations, *Pure Appl. Funct. Anal.*, **1** (2016), 231–256.
[11] D. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization, *Mathematical Programming*, **45** (1989), 503–528.
[12] D. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, 3^{rd} edition, Springer, US, 2008.
[13] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2^{nd} edition, Springer Series in Operations Research and Financial Engineering, Springer, New York, 2006.
[14] N. Petit and P. Rouchon, Dynamics and solutions to some control problems for water-tank systems, *IEEE Transactions on Automatic Control*, **47** (2002), 594–609.
[15] J.-H. Webert, *Structure-exploiting Optimization Algorithms for an Optimal Control Problem with Coupled Hyperbolic and Ordinary Differential Equation Constraints* M.Sc. thesis, Universität der Bundeswehr München, Neubiberg/München, 2015.

Received xxxx 20xx; revised xxxx 20xx.

E-mail address: jan_hendrik_webert@web.de
E-mail address: pgill@ucsd.edu
E-mail address: sven-joachim.kimmerle@unibw.de
E-mail address: matthias.gerdts@unibw.de