Zentrum für Technomathematik
Fachbereich 3 – Mathematik und Informatik

Nonlinear large-scale Optimization with WORHP

Tim Nikolayzik Christof Büskens
Matthias Gerdts

Report 10–08

Berichte aus der Technomathematik

Report 10–08 Dezember 2010

Nonlinear large-scale Optimization with WORHP

Tim Nikolayzik[*] and Christof Büskens[†]

Universität Bremen, 28359 Bremen, Germany

Matthias Gerdts[‡]

Universität Würzburg, 97074 Würzburg, Germany

December 20, 2010

In this paper we present the new nonlinear optimization solver WORHP which is capable of solving large-scale, sparse problems. A short introduction in nonlinear optimization and a discussion of details of the new solver is offered. At the end of this paper we state numerical results and present two applications to demonstrate the capabilities of the proposed method.

1 Introduction

Nonlinear optimization has grown to a key technology in many areas of aerospace industry, especially for solving discretized optimal control problems with ODEs, DAEs and PDEs. Examples for this applications are satellite control, shape-optimization, aerodynamamics, trajectory planning, reentry problems and interplanetary flights. One of the most extensive areas is the optimization of trajectories for aerospace applications. Nonlinear optimization problems arising from these applications typically are large and sparse. Previous methods for solving nonlinear optimization methods were developed for small to medium sized and dense problems. Using these kind of solvers for large-scale sparse problems leads to very high computational efforts and a higher risk of an unsuccessful termination.

[*]Research Associate, Zentrum für Technomathematik, University of Bremen, Bibliothekstraße 1, 28359 Bremen

[†]Professor, Zentrum für Technomathematik, University of Bremen, Bibliothekstraße 1, 28359 Bremen, Germany

[‡]Professor, Institut für Mathematik, Universität Würzburg, Würzburg, 97074, Germany

To solve these problems one has to exploit as much information of the problem as possible. This includes an efficient storing of the occurring matrices and vectors, special linear algebra for solving sparse, large-scale linear equations, an appropriate approximation of the Hessian, etc. Most of the available optimization methods are using update techniques, introduced by Broyden-Fletcher-Goldfarb-Shanno (BFGS). These update techniques have several advantages, for example they guarantee that the positive definiteness of the Hessian approximation such that further computations can be performed much easier. The biggest advantage is the efficiency of the calculation for small and medium sized problems. In case of large sparse optimization problems the sparsity can not be exploited and the approximation of the Hessian is getting dense.

These limitations motivates the idea of developing a new solver which is able to efficiently solve large sparse nonlinear optimization problems, by using the exact Hessian.

In this paper we first give a brief overview about nonlinear optimization and some background about methods for solving such problems. Then we introduce the general methodology of the new solver WORHP (We Optimize Really Huge Problems) and present its advantages and techniques in more detail. Numerical results from different applications demonstrates the capabilities of the new proposed method.

2 Nonlinear Optimization

We state the following nonlinear optimization problem (NLP)

$$\begin{aligned} \min_{x\in\mathbb{R}^N} \quad & F(x), \\ \text{subject to} \quad & G_i(x) = 0,\ i = 1,\dots,M_e, \\ & G_j(x) \le 0,\ j = M_e+1,\dots,M. \end{aligned} \tag{NLP}$$

Therein $x \in \mathbb{R}^N$ denotes the vector of optimization variables with objective function $F : \mathbb{R}^N \to \mathbb{R}$ and constraints $G : \mathbb{R}^N \to \mathbb{R}^M, G(x) = (G_1(x),\dots,G_M(x))^T$. All functions are assumed to be sufficiently smooth.

Special cases of (NLP) are linear optimization problems (linear programming), quadratic optimization problems (quadratic programming, QP), discrete optimal control problems, trajectory optimization problems or constrained nonlinear least-squares problems.

The aim is to find the vector $\overline{x} \in \mathbb{R}^N$, which satisfies the constraints G and uses the remaining degrees of freedom to minimize the given objective function F. The sets

$$\begin{aligned} I(\overline{x}) \quad & := \{i \in \{M_e+1,...,M\}\,|G_i(\overline{x}) = 0\}, \\ J(\overline{x}) \quad & := I(\overline{x}) \cup \{1,...,M_e\}, \end{aligned}$$

are called set of active indices. To find $\overline{x}$ it is necessary to introduce the Lagrangian

$$L(x,\lambda) := F(x) + \lambda^T G(x),$$

whereas $\lambda \in \mathbb{R}^M$ is the vector of the Lagrange multipliers. The necessary first order optimality conditions, also called KKT-conditions, guarantee that if $\overline{x}$ is a local minimum of (NLP) and moreover $\overline{x}$ is regular (cf. Mangasarian-Fromowitz[1]), then there exists $\overline{\lambda} \in \mathbb{R}^M$ such that hold:

$$\begin{aligned} \nabla_x L(\overline{x}, \overline{\lambda}) = \nabla_x F(\overline{x}) + \overline{\lambda}^T \nabla_x G(\overline{x}) = 0 \\ \lambda_i \geq 0,\ i \in I(\overline{x}) \\ \lambda_j = 0,\ j \notin J(\overline{x}) \\ \lambda^T G(\overline{x}) = 0 \end{aligned} \tag{1}$$

If actually $\nabla_x G_i(\overline{x}), i \in J(\overline{x})$ are linearly independent then also (1) holds but also $\overline{\lambda}$ is unique. This criterion is used to search for optimal solutions of (NLP).

3 Methods for solving NLP problems

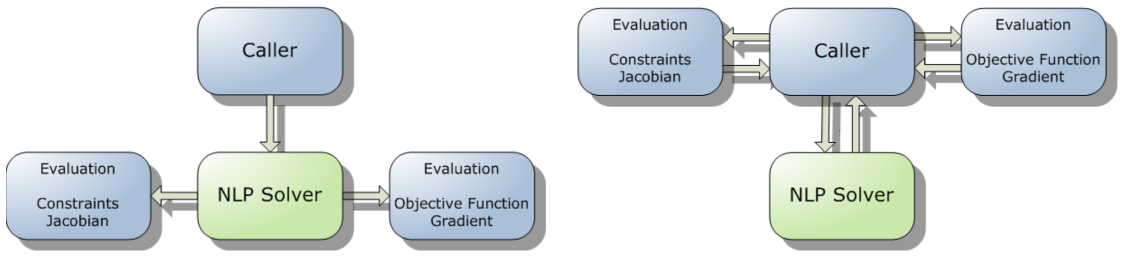
WORHP is a mixed SQP (Sequential Quadratic Programming) and IP (Interior-Point) method, which aim is to solve sparse large-scale NLP problems with more than 1,000,000 variables and constraints. The general idea of SQP methods was introduced by Han in 1977 (and earlier by Wilson in 1963). Since then they belong to the most frequently used algorithms for the solution of practical optimization problems due to their robustness and their good convergence properties (global convergence and locally superlinear convergence rate). The basic idea of interior-point methods is to handle inequality constraints by adding them with a weighted logarithmic barrier term to the objective function. Then, a sequence of equality constrained nonlinear programs is solved while simultaneously the weight parameter in the objective function tends to zero. Since WORHP is an iterative method. It generates a sequence of points $\{x^{[k]}\}_{k=0,1,2,\ldots}$ with $x^{[k]} \overset{k\to\infty}{\longrightarrow} \overline{x}$ by:

$$x^{[k+1]} = x^{[k]} + \alpha^{[k]} d^{[k]}, \tag{2}$$

whereas $d^{[k]} \in \mathbb{R}^N$ is an appropriate search direction and $\alpha^{[k]} \in (0, 1]$ a suitable step size. In each step of the method the search direction is determined by solving a quadratic optimization problem. Often, the Hessian of the Lagrangian used inside the quadratic subproblem is replaced by update formulas of BFGS type which have the additional benefit that only strictly convex quadratic programs have to be solved. This strategy works well for small to medium sized problems but it turns out to be crucial for large-scale problems as the update formulas lead to a fast fill-in of elements in the update matrices which in turn leads to dense matrices. Therefore, in the context of large-scale problems one is often forced to use the exact Hessian, which may be indefinite and leads to non-convex quadratic programs.

Alternative attempts use limited memory BFGS updates or sparse update formulas, compare Fletcher [2]. Both approaches are crucial in view of their convergence properties and their computational complexity, respectively. A handicap of these methods are their locally restricted properties, hence special globalization techniques have to be introduced to enlarge the radius of convergence.

One classical approach to promote global convergence for remote starting points is to perform a line-search for a merit function which is usually given by an exact penalty



(a) Traditional calling convention (“Fire and Forget”).

(b) Reverse Communication.

Figure 1: Different ways of calling an NLP solver.

function such as the L1-penalty function or the augmented Lagrangian function. For more details compare the work of Schittkowski [3], Gill, Murray and Wright [4] and Gill et al. [5].

The determination of derivatives is a crucial element in nonlinear optimization. Basically first derivatives as the gradient of the objective function or the Jacobian of the constraints are necessary in order to find a descent direction to the point where the next local minimum is expected. Second derivatives (Hessian of the Lagrangian) are used to guarantee a quadratic convergence behavior and to decide how far to follow the descent direction. There are different ways to calculate these derivative information. WORHP provides several of them: The solver includes for example a method using finite differences (FD) and WORHP can use special sparse BFGS update techniques. The FD module uses the so-called “group strategy” based on the graph coloring theory which speeds up the calculations extremely for sparse problems.

4 WORHP

As mentioned before WORHP is an iterative solver and produces a sequence of points $\{x^{[k]}\}_{k=0,1,2,\ldots}$. The basic scheme of the algorithm is the following:

i. Terminate if $x^{[k]}$ satisfies a termination criterion.

ii. Approximate the nonlinear problem by a quadratic subproblem in $x^{[k]}$ and use its solution $d^{[k]}$ as the search direction.

iii. Determine a step size $\alpha^{[k]}$ by applying a line search method to a merit function.

iv. Update the iterate by (2), increment k and go to i.

Instead of using the restrictive formulation used in (NLP) the more general description

$$\begin{aligned} \min_{x \in \mathbb{R}^N} \quad & F(x), \\ \text{subject to} \quad & l \leq G(x) \leq u, \end{aligned}$$

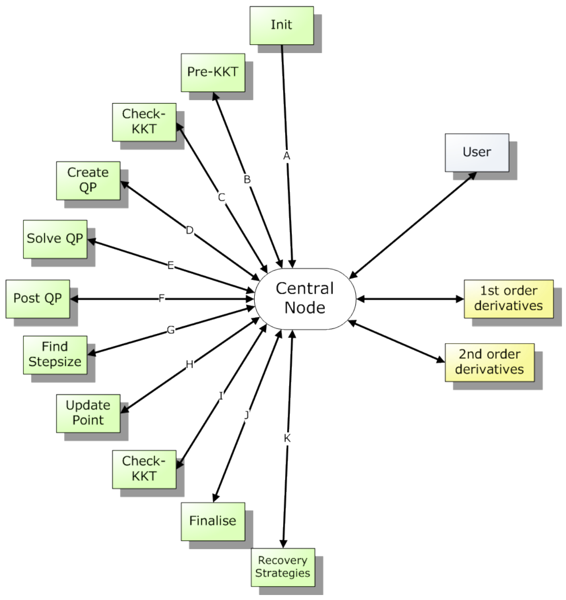


Figure 2: Data flow in WORHP

can be applied, whereas $G(x) = (G_1(x), \dots, G_M(x))^T$ and $l, u \in \mathbb{R}^M$.

In the next sections we will describe these basic steps of WORHP in more detail.

4.1 Architecture

WORHP is based on a reverse communication architecture that offers unique flexibility and control over the optimization process, see Figure 1.

The solver is aimed at the highest degree of control and possibilities of intervention. One central architectural principle is the complete disuse of internal (program flow) loops or jumps.

Each call of the solver carries out a defined minor iteration. They are grouped together to major NLP iterations. Among others these are (cf. Figure 2):

- Get objective function value, constraints, gradient, Jacobian or Hessian from user
- Update Hessian
- Check KKT conditions
- Create subproblem (QP or primal-dual system)
- Find step size $d^{[k]}$

These stages together form the SQP method. However, the general workflow of WORHP looks similar to other SQP methods, see Figure 3.

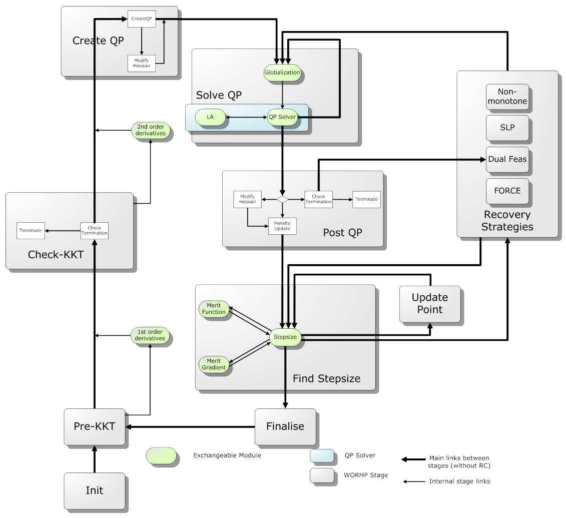


Figure 3: WORHP

4.2 Checking for optimality

For testing an iterate $x^{[k]}$ the first order necessary optimality conditions (1) have to be evaluated. For this it is necessary to calculate the first order derivatives. These derivatives can be provided by the user or WORHP calculates them by itself as mentioned before. The iterate $x^{[k]}$ is said to be optimal if the following holds

$$\nabla_x L(x^{[k]}, \lambda^{[k]}) = \nabla_x F(x^{[k]}) + \lambda^{[k]T} \nabla_x G(x^{[k]}) \leq \epsilon_{\text{opti}}, \tag{3}$$

$$\lambda_i^{[k]} \geq -\epsilon_{\text{comp}}, i \in I(x^{[k]}), \tag{4}$$

$$\lambda_j^{[k]} \leq \epsilon_{\text{comp}}, j \notin J(x^{[k]}). \tag{5}$$

and

$$\begin{aligned} |G_i(x^{[k]})| &\leq \epsilon_{\text{feas}},\ i = 1, \dots, M_e, \\ G_j(x^{[k]}) &\leq \epsilon_{\text{feas}},\ j = M_e + 1, \dots, M. \end{aligned} \tag{6}$$

WORHP also supports a scaled version of the original KKT-conditions. These conditions are motivated by the idea that the user hardly can interpret the numerical optimality by (3) but is interested in

$$|F(\overline{x}) - F(x^{[k]})| \leq \epsilon_{\text{tol}}.$$

This leads to

$$||\nabla_x L(x^{[k]}, \lambda^{[k]})||_\infty \leq \frac{\epsilon_{\text{opt}} \max\left(1, |F(x^{[k]})|\right) + ||\lambda_1 G_1(x^{[k]}), \dots, \lambda_M G_M(x^{[k]})||_\infty}{||d^{[k]}||_\infty}. \tag{7}$$

In order to prevent WORHP from iterating too long without bee able to fulfill (7), e.g. due to numerical reasons of inexact derivative approximations, a low pass filter is implemented. Among others this filter calculates two thresholds

$$\text{Filter}_{\text{obj}}^{[k]} = \alpha_{\text{f\_obj}} F(x^{[k]}) + (1 - \alpha_{\text{f\_obj}}) \text{Filter}_{\text{obj}}^{[k-1]}$$

and

$$\text{Filter}_{\text{con}}^{[k]} = \alpha_{\text{f\_con}} ||G(x^{[k]})||_\infty + (1 - \alpha_{\text{f\_con}}) \text{Filter}_{\text{con}}^{[k-1]}.$$

If an iterate is not satisfying the conditions (3)-(6) then for a given $\epsilon_{\text{filter}} > 0$ the conditions

$$\frac{|\text{Filter}_{\text{obj}}^{[k]} - \text{Filter}_{\text{obj}}^{[k-1]}|}{\max(1, |\text{Filter}_{\text{obj}}^{[k]}|)} < \epsilon_{\text{filter}} \quad \text{and} \quad \frac{|\text{Filter}_{\text{con}}^{[k]} - \text{Filter}_{\text{con}}^{[k-1]}|}{\max(1, |\text{Filter}_{\text{con}}^{[k]}|)} < \epsilon_{\text{filter}}$$

are checked. If both conditions are fulfilled and the current iterate $x^{[k]}$ is feasible, this point is assumed to be optimal. If the point is not feasible a feasibility mode is activated, cf. 4.6

4.3 Solving the QP-subproblem

Let $x^{[k]}$ be the approximation of the optimal solution in the k-th iteration and $B^{[k]}$ a suitable approximation of the Hessian of the Lagrangian, then the associated QP problem is:

$$\begin{aligned} \min_{d^{[k]} \in \mathbb{R}^N} \; & \nabla_x F(x^{[k]}) d^{[k]} + \frac{1}{2} d^{[k]T} B^{[k]} d^{[k]}, \\ \text{subject to } & G_i(x^{[k]}) + \nabla_x G_i(x^{[k]}) d^{[k]} = 0, \; i = 1, \ldots, M_e \\ & G_j(x^{[k]}) + \nabla_x G_j(x^{[k]}) d^{[k]} \leq 0, \; j = M_e + 1, \ldots, M \end{aligned} \tag{8}$$

Therefore the second order information, e.g. the exact Hessian or an approximation by BFGS update formulas is needed. Again, the user can provide this information or it is approximated by WORHP.

This QP subproblem is motivated by the fact that the KKT conditions for (8) can be written for $i \in J(x^{[k]})$ as

$$B^{[k]} d + \nabla_x F(x^{[k]}) + \nabla_x G_i(x^{[k]})^T \lambda_{QP}^{[k]} = 0 \tag{9}$$

$$G_i(x^{[k]}) + \nabla_x G_i(x^{[k]}) d = 0, \tag{10}$$

therein $\lambda_{QP}^{[k]}$ is the corresponding vector of the Lagrange multipliers of (8). (9) can be reformulated in the following way, with $\nabla_x G_a(x^{[k]})$ as the Jacobian of the active constraints

$$\begin{pmatrix} B^{[k]} & \nabla_x G_a^T(x^{[k]}) \\ \nabla_x G_a(x^{[k]}) & 0 \end{pmatrix} \begin{pmatrix} d^{[k]} \\ \lambda_{QP}^{[k]} \end{pmatrix} = - \begin{pmatrix} \nabla_x F(x^{[k]}) \\ G_i(x^{[k]}) \end{pmatrix}.$$

Applying the Newton method yields the same system.

The method implemented in WORHP for solving quadratic subproblems is a primal-dual interior-point method, cf. Gertz and Wright[6]. In most of the cases the solution $\overline{d}^{[k]}$ of the QP subproblem is an appropriate search direction while $\overline{\lambda}^{[k]}$ approximates the Lagrange multipliers.If the QP subproblem is non-convex regularization techniques of Levenberg-Marquardt type are used to convexify the problem, see Section 4.5.

Further problems which might occur are inconsistencies in the linearized constraints,hence for the practical realization the original QP problem (8) is solved but a relaxed formulation:

$$\begin{aligned}
\min_{d^{[k]}\in\mathbb{R}^N,\delta^{[k]}\in\mathbb{R}} \quad & \nabla_x F(x^{[k]})d^{[k]} + \tfrac{1}{2}d^{[k]T}B^{[k]}d^{[k]} + \tfrac{\eta_r}{2}\delta^{[k]}, \\
\text{subject to} \quad & G_i(x^{[k]})(1-\delta^{[k]}) + \nabla_x G_i(x^{[k]})d^{[k]} = 0, \ i = 1,\dots,M_e \\
& G_j(x^{[k]})(1-\sigma_i\delta^{[k]}) + \nabla_x G_j(x^{[k]})d^{[k]} \le 0, \ j = M_e+1,\dots,M
\end{aligned}$$

where $\delta^{[k]}$ denotes the relaxation variable,

$$\sigma_i = \begin{cases} 0, & \text{if } G_i(x^{[k]}) < 0, \\ 1, & \text{otherwise,} \end{cases}, i = M_e+1,\dots,M.$$

and $\eta_r \in \mathbb{R}^+$ is a penalty weight.

4.4 Merit Function

In order to achieve global convergence, we have to find an appropriate step size $\alpha^{[k]}$ to the solution $d^{[k]}$ of the QP subproblem. Therefore we use merit functions.

For determining a suitable step size one has to measure the progress of the optimization process, which consists of both, a quantification of the objective and the constraints. For this purpose a merit function is used. Merit functions supported by WORHP are e.g. the L_1-penalty function

$$\begin{aligned}
L_1(x;\eta) := F(x) + \sum_{i=1}^{M_e} \eta_i |G_i(x)| \\
+ \sum_{i=M_e+1}^{M} \eta_i \max\{0, G_i(x)\},
\end{aligned}$$

or the augmented Lagrangian

$$\begin{aligned}
L_a(x,\lambda;\eta) := f(x) + \sum_{i=1}^{M_e} \lambda_i G_i(x) + \frac{1}{2}\sum_{i=1}^{M_e} \eta_i G_i^2(x) \\
+ \frac{1}{2\eta_i} \sum_{i=M_e+1}^{M} \left(\left(\max\{0, \lambda_i + \eta_i G_i(x)\}\right)^2 - \lambda_i^2 \right),
\end{aligned}$$

whereas $\eta \in \mathbb{R}^M$, with $\eta_i \ge 0, i = 1,\dots,M$, is a penalty vector.

4.5 Hessian Regularization

To ensure that the solution of the QP subproblem is unique and reasonable one has to assure that the Hessian $H_L = (h_{ij})_{i,j} \in \mathbb{R}^{N \times N}$ is positive definite. To achieve this we use the modified Hessian

$$H = H_L + \tau(|\sigma| + 1)I, \tag{11}$$

cf Betts[7]. The parameter $\tau \in \mathbb{R}$ holds $0 \leq \tau \leq 1$, while σ is the Gerschgorin bound for the most negative of H_L, i.e.

$$\sigma = \min_{1 \leq i \leq n} \left\{ h_{ii} - \sum_{i \neq j}^{n} |h_{ij}| \right\}.$$

The original idea was suggested by Levenberg [8]. He used the matrix $\overline{\tau} I$ as an approximation of the Hessian for least squares problems.

The choice of τ is crucial for the rate of convergence of the overall algorithm. The setting $\tau = 1$ guarantees a positive definite Hessian but leads to a slower convergence since it causes a large deviation to the original Hessian. On the other hand, $\tau = 0$ leads to the described problems, since in this case the original Hessian without regularization is used. The idea of Betts [7] is to reduce τ when the predicted reduction in the merit function coincides with the actual one, and increase the parameter otherwise. In the following we denote with $M^{[k]}(x^{[k]}, \lambda^{[k]})$ the value of one of the merit functions introduced in section 4.4 in the k-th iteration. The following three quantities have to be computed. The actual reduction:

$$\rho_1 = M^{[k-1]}(x^{[k]}, \lambda^{[k]}) - M^{[k]}(x^{[k]}, \lambda^{[k]}), \tag{12}$$

where M^k is the value of the merit function in the k-th iteration. The predicted reduction:

$$\rho_2 = M^{[k-1]}(x^{[k]}, \lambda^{[k]}) - \tilde{M}^{[k]}(x^{[k]}, \lambda^{[k]}) = -M_0'(x^{[k]}, \lambda^{[k]}) - \frac{1}{2} d^{[k]T} H d^{[k]}, \tag{13}$$

with the predicted value of the merit function $\tilde{M}^{[k]}(x^{[k]}, \lambda^{[k]})$ and the derivative of the merit function $M_0'(x^{[k]}, \lambda^{[k]})$ with respect to the step size $\alpha^{[k]}$ evaluated at $\alpha^{[k]} = 0$. At last, we need to compute the rate of change in the norm of the gradient of the Lagrangian

$$\rho_3 = \frac{||\vartheta^k||_\infty}{||\vartheta^{k-1}||_\infty} \tag{14}$$

where the error in the gradient of the Lagrangian is

$$\vartheta = \nabla F + (\nabla G)^T \mu + \lambda.$$

4.6 Line Search

After determining the search direction $d^{[k]}$ from the QP subproblem, it is crucial to find a suitable step size $\alpha^{[k]}$. For this we use a line search method together with the Armijo Rule. We define

$$\phi(\alpha) := M(x^{[k]} + \alpha d^{[k]}, \lambda^{[k+1]}(\alpha), \eta),$$

whereas $\lambda(\alpha)$ is the new multiplier depending on α, e.g. $\lambda^{[k+1]}(\alpha) = (1-\alpha)\lambda^{[k]} + \alpha\lambda_{QP}$ and the function M is one of the merit functions introduced in Section 4.4. In general a good choice would be the α which minimizes $\phi(\alpha)$, unfortunately this defines another nonlinear optimization problem. Although it is one dimensional, it is too time consuming, especially for large-scale problems. So our goal is to find the largest step size α which fulfills the Armijo condition

$$\phi(\alpha) < \phi(0) = M(x^{[k]}, \lambda^{[k]}, \eta).$$

As we do not want to invoke another optimization, although it is one-dimensional, we do not apply an exact line search.
Starting with a maximum step size $\alpha_{\max} \in (0,1]$ and a factor $\beta_{\text{armijo}} \in (0,1)$, candidates for the step size are

$$\left\{\alpha_j = \beta^j \alpha_{\max} | j = 0, 1, 2, \dots, l_{\max}\right\},$$

whereas $l_{\max} = \max\{l \in \mathbb{N} | \beta^l \alpha_{\max} \geq \alpha_{\min}\}$ and $\alpha_{\min}$ is the smallest step size allowed. As the first step size we choose $\alpha_0 = \alpha_{\max}$. If the Armijo condition

$$\phi(\alpha_j) \leq \phi(0) + \sigma\alpha_j\phi'(0), \tag{15}$$

whereas $\sigma \in \mathbb{R}^+$ is a suitable factor and $\phi'(0)$ the derivative of ϕ with respect to α is not fulfilled, $\alpha_1 = \beta^1\alpha_0$ is calculated and again (15) is checked. This is done as long as a suitable α is found or the line search has failed, i.e. $i > l_{\max}$. If the line search has failed, WORHP uses recovery strategies to prevent the algorithm from failing. Several recovery strategies are implemented, e.g.

SLP This strategy is motivated by gradient methods. Instead of using a second-order approximation of the Hessian in (8), the identity matrix is used.

Feasible mode If the line search has failed at a point which is not feasible, this mode is a good choice for saving the algorithm. In this mode the QP-problem is modified in an extensive way. The new QP-problem is focused on the feasibility of the problem. The mode will be stopped after a feasible point is found. Afterwards the normal optimization is restarted at the new, now feasible, iterate.

4.7 The algorithm of WORHP

Next we state the algorithm in detail:
Algorithm: Given are a starting point (x_0) and a set of constants including e.g. $\epsilon_{\text{opti}},\ \epsilon_{\text{feas}} > 0,\ \epsilon_{\text{comp}} > 0,\ \varepsilon > 0,\ \beta_{\text{armijo}} \in (0,1),\ \alpha_{\max} \in (0,1]$.

A-1: *Initialize.* Set iteration counter $k = 0$.

A-2: *Check-KKT.* Check optimality conditions.

A-3: *Create-QP.* Set matrix $B^{[k]}$. If $B^{[k]} = I_N$ go to **A-5** else go to **A-4**.

A-4: *Hessian-Regularization.* Update of the Hessian according to section 4.5.

A-5: *Solve-QP.*

- **A-5.1:** If the QP-subproblem was not solved successfully, check if $\tau_{\sum} < 1$. If this is the case go to **A-4**.
- **A-5.2:** If $k = 0$ and $M > 0$ then set $\lambda_0 = \lambda_{QP}$ and $\mu_0 = \mu_{QP}$ and got to **A-3**.
- **A-5.3:** If QP-subproblem was solved successfully go to **A-6**.

A-6: *Post-QP.*

- **A-6.1:** If $\delta \geq \min(1, \delta_{\max})$ increase η_r by $\eta_r = \rho_{\text{relax}}\eta_r$ and go to **A-5**.
- **A-6.2:** If $||d^{[k]}||_2 < \sqrt{\varepsilon}$ try to activate feasibility mode and go to **A-7**, if this is not possible terminate with error.
- **A-6.4:** Go to **A-9**.

A-7: *Solve Feasibility QP.*

- **A-7.1:** Determine the set of current active constraints.
- **A-7.2:** Determine new $d^{[k]}$ by solving an equality constraint quadratic subproblem and go to **A-9**.

A-8: *Find step size.*

- **A-8.1:** Set $\alpha^{[k]} = \alpha_{\max}$ and go to **A-9**.
- **A-8.2:** Check if the Armijo condition (15) is fulfilled, if yes go to **A-11**
- **A-8.3:** Else set $\alpha^{[k]} = \alpha^{[k]} \cdot \beta_{\text{armijo}}$, if $\alpha^{[k]} \leq \alpha_{\min}$ go to **A-10** else go to **A-9**.

A-9: *Update Point.*

- **A-9.1:** Compute new iterate $x^{[k+1]} = x^{[k]} + \alpha d^{[k]}$.
- **A-9.2:** Update multipliers $\lambda^{[k]}$ and $\mu^{[k]}$.
- **A-9.3:** Go to **A-8.2**.

A-10: *Recovery Strategies.* Start selected recovery strategy and go to **A-8**.

A-11: *Finalize.*

- **A-11.1:** Compute ρ_1 by (12).
- **A-11.2:** Set $k = k + 1$.
- **A-11.3:** Go to **A-2**.

4.8 Interfaces

WORHP offers a wide spread of different interfaces, starting from the “Full-Feature-Interface” with reverse communication allowing close monitoring of, and control over all quantities involved in the optimization process for skilled users, down to others and finally ends with the conventional interface which is very similar to classical interfaces used by other solvers.

WORHP currently offers eight interfaces, divided into three classes for use in different environments.

The AMPL interface Executable for use with the AMPL modeling language.

The MATLAB interface Mex-object with a function interface to use inside the MATLAB and Simulink programming environment.

The Library interfaces WORHP has three interfaces, each for C/C++ and Fortran, to allow it to be included as optimization library into user code. They differ in their function signatures and the communication convention. All three are available as equivalent C/C++ and Fortran versions:

- Full Feature Interface: Reverse Communication, Unified Solver Interface.
- Basic Feature Interface: Direct Communication, Unified Solver Interface.
- Legacy Interface (also called Simple or Traditional Interface): Direct Communication, traditional interface.

Within the solver different linear algebra solvers can be used to solve the quadratic subproblem. WORHP offers the possibility to use these linear algebra packages

- MA48
- MA57
- PARDISO
- SuperLU
- MUMPS
- LAPACK
- WSMP

5 Results

The robustness of WORHP is proved by several test sets. We present numerical results for the CUTEr test set, which consists of a collection of 920 large-scale sparse and small dense problems. As reference solvers we use IPOPT 3.8.1 with MA57 [9] as well as KNITRO 6.0.0 and SNOPT 7.2.8 [10]. The test set is implemented in AMPL. Table 1 shows the results of the solvers.

The standard settings of the solvers were used while the scaling of the constraints was turned off for every solver, since it causes improper terminations for some of the solvers. The computational time for a single problem of the test set was limited to 5 hours and the precision for the constraints and the optimality conditions is set to 10^{-6}. The calculations were done on a Linux system with an Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz with 4GB RAM.

WORHP is capable of solving more than **99.5%** of the problems of the CUTEr test set. The 5 problems which WORHP was not able to solve were also not solved by any

Table 1: Comparison of the four solvers on the CUTEr test set.

	WORHP 1.0	IPOPT 3.8.1	KNITRO 6.0.0	SNOPT 7.2.8
Problems solved	915	883	885	825
Optimal solution found	910	866	879	809
Acceptable solution found	5	17	6	16
Not solved	5	37	35	95
Percentage	99.46	95.98	96.2	89.67
Time	$7450s$	$7146s$	$142200s$	$362823s$

of the other solvers. We think these problems are somehow formulated wrong or are not solvable. IPOPT was able to solve about **96**% of all the problems while KNITRO solves **96.2**% and SNOPT **89.7**%. The overall computational time for all the 920 test cases is similar for WORHP and IPOPT. WORHP seems to be faster for large-scale problems while IPOPT leads for small and medium sized problems. The relatively large overall computational times for KNITRO and SNOPT are a result of no termination within the given time limit for some of the problems. Figure 4 gives a more detailed overview about the efficiency of the solvers. The percentage of solved problems, sorted by the

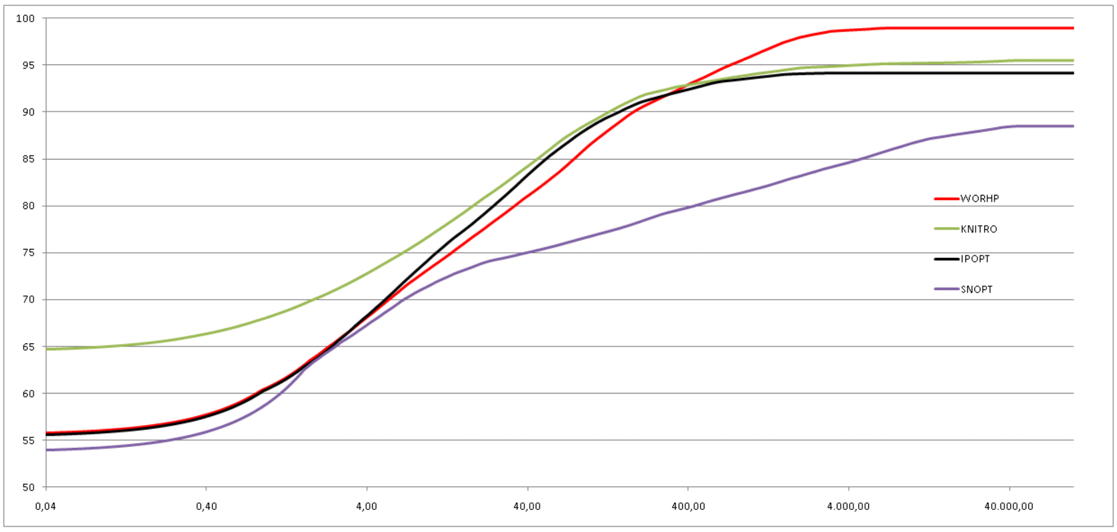


Figure 4: Percentage of optimally solved problems within given time frame.

computational time, is plotted against the accumulated time frame in seconds. Within the first second KNITRO is the fastest solver, the other three are very close. After ten seconds WORHP, KNITRO and IPOPT are close while SNOPT falls back. One should keep in mind, that within the first ten seconds only small and medium sized problems are calculated while the remaining larger problems are solved afterwards.

6 Two examples

The following two examples are full discretized optimal control problems. The calculations were also done on a Linux system with an Intel(R) Core(TM)2 Quad CPU Q6600 @ 2.40GHz and 4GB RAM.

6.1 Time optimal low-thrust planar transfer to a geosynchronous orbit

The aim of this task is to find a thrust direction control $u(t), 0 \leq t \leq t_f$, that minimizes the final time $F(x, u) = t_f$, subject to

$$\begin{aligned}
&\dot{x}_1 = x_2, && x_1(0) = 6.0, && x_1(t_f) = 6.6 \\
&\dot{x}_2 = \frac{x_3^2}{x_1} - \frac{r_\mu}{x_1^2} + 0.01\sin(u), && x_2(0) = 0.0, && x_2(t_f) = 0.0 \\
&\dot{x}_3 = -\frac{x_2 x_3}{x_1} + 0.01\cos(u), && x_3(0) = \sqrt{\frac{r_\mu}{x_1(0)}}, && x_3(t_f) = \sqrt{\frac{r_\mu}{x_1(t_f)}} \\
&\dot{x}_4 = \frac{x_3}{x_1}, && x_4(0) = 0.0,
\end{aligned}$$

where $x_1(t)$ represents the radial position, $x_2(t)$ the radial velocity, $x_3(t)$ the circumferential velocity and $x_4(t)$ the polar angle. The gravitational parameter for the earth is represented by $r_\mu = 62.5$. This problem is taken from Kluever[11].

This optimal control problem is fully discretized using Eulers method and hence lead to a sparse nonlinear optimization problem, cf. Büskens and Maurer[12]. The size of the problem is determined by the number of points used for the discretization of the problem. Using 101 discrete points the resulting nonlinear optimization problem consists of $506 = 5 \cdot 101 + 1$ optimization variables and $407 = 4 \cdot 100 + 7$ nonlinear constraints. The computational time for this problem was limited to 2 hours and the precision for the constraints was set to 10^{-6} and the optimality conditions ito 10^{-5}.

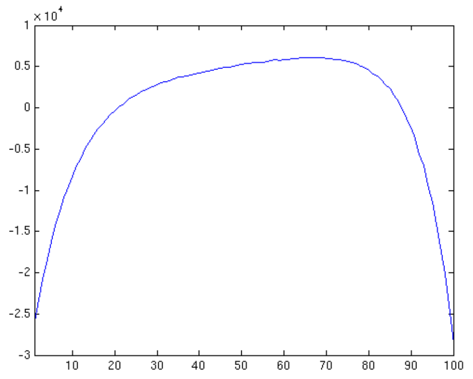
WORHP is able to solve this problem in 0.77 seconds to the final objective value of 17.153705844. The initial guess for the states was 1.0 and for the control 0.0. The optimal control and the optimal states are shown in figure 5. The results of all solvers are summarized in table 2.

Table 2: Comparison of the four solvers.

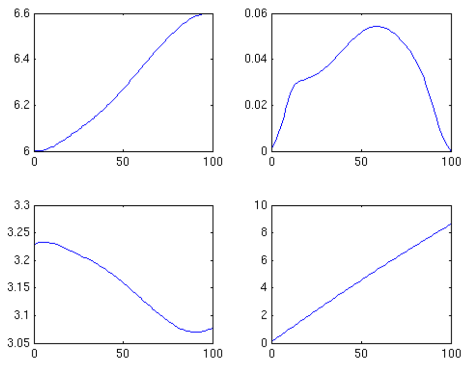
	WORHP 1.0	IPOPT 3.8.1	KNITRO 6.0.0	SNOPT 7.2.8
Objective function	17.153706	17.153706	MaxIter	17.153711
Time	$0.77s$	$0.91s$	$39.26s$	$0.63s$

6.2 Emergency landing of a hypersonic flight system

In this example the emergency landing of a two stage space transport vehicle is treated. After the separation of the first stage the engine of the upper stage can't be ignited.



(a) Optimal control $u(t)$.



(b) Optimal states $x_i(t).i = 1 \dots 4$.

Figure 5: Optimal control and optimal states of the low-thrust planar transfer problem.

Because of this propulsion damage the upper stage can't reach a safe orbit. For more details see Mayrhofer and Sachs [13] or Büskens and Gerdts [14][15].

For the description of the dynamic of the flight system a mass point model with six states and two control function is used. If one assumes a rotating, spherical earth, as the reference system, the equations of motion can be formulated as follows

$$
\begin{aligned}
\dot{v} &= -D(v, h; C_L)\frac{1}{m} - g(h)\sin\gamma + \\
&\quad \omega^2 \cos\Lambda(\sin\gamma\cos\Lambda - \cos\gamma\sin\chi\sin\Lambda + \cos\gamma\cos\Lambda)\frac{r(h)}{v}, \\
\dot{\gamma} &= L(v, h; C_L)\frac{\cos\mu}{mv} - \left(\frac{g(h)}{v} - \frac{v}{r(h)}\right)\cos\gamma + \\
&\quad 2\omega\cos\chi\cos\Lambda + \omega^2\cos\Lambda(\sin\gamma\sin\chi\sin\Lambda + \cos\gamma\cos\Lambda)\frac{r(h)}{v}, \\
\dot{\chi} &= L(v, h; C_L)\frac{\sin\mu}{mv\cos\gamma} - \cos\gamma\cos\chi\tan\Lambda\frac{v}{r(h)} + \\
&\quad 2\omega(\sin\chi\cos\Lambda\tan\gamma - \sin\Lambda) - \omega^2\cos\Lambda\sin\Lambda\cos\chi\frac{r(h)}{v\cos\gamma} \\
\dot{h} &= v\sin\gamma \\
\dot{\Lambda} &= \cos\gamma\sin\chi\frac{v}{r(h)} \\
\dot{\Theta} &= \cos\gamma\cos\chi\frac{v}{r(h)\cos\Lambda}.
\end{aligned}
$$

The appeared functions are defined by

$$
\begin{aligned}
r(H) &= r_0 + h, & g(h) &= g_0 \left(\frac{r_0}{r(h)}\right)^2, \\
D(v, h; C_l) &= q(c, h) F c_D(C_L), & \rho(h) &= \rho_0 e^{-\beta h}, \\
c_D(C_L) &= c_{D_0} + k C_L^2, & q(v, h) &= \frac{1}{c} \rho(h) v^2, \\
L(v, h; C_L) &= q(v, h) F C_L.
\end{aligned}
$$

The constants are chosen

$$
\begin{aligned}
c &= 2.0, & c_{D_0} &= 0.017, & r_0 &= 6.371 \cdot 10^6 \\
F &= 305.0, & g_0 &= 9.80665, & k &= 2.0, \\
\omega &= 7.270 \cdot 10^{-5}, & \beta &= \frac{1}{6900.0}, & \rho_0 &= 1.249512.
\end{aligned}
$$

The state variables consists of the velocity v, the flight path angle γ, the course angle χ, the altitude h, the longitude Λ and the latitude Θ. The control functions C_L (lift coefficient) and μ (angle of bank) are restricted by

$$0 \leq C_L \leq 1, \qquad 0 \leq \mu \leq 1.$$

The mass is supposed to be constant $m = 115000$. The initial values are given by

$$
\begin{pmatrix} v(0) \\ \gamma(0) \\ \chi(0) \\ h(0) \\ \Lambda(0) \\ \Theta(0) \end{pmatrix} = \begin{pmatrix} 2150.5452900 \\ 0.1520181770 \\ 2.2689279889 \\ 33900.000000 \\ 0.9268828079 \\ 0.1544927057 \end{pmatrix}
$$

which corresponds to a reentry point over Bremen. The initial values are also used as starting point for the optimization.

For safety reasons it is necessary to find a trajectory with maximum distance to the starting point over the rotating earth:

$$F(\mu, C_L, t_f) = \left(\frac{\Lambda(t_f) - \Lambda(t_0)}{\Lambda(t_0)}\right)^2 + \left(\frac{\Theta(t_f) - \Theta(t_0)}{\Theta(t_0)}\right)^2$$

As a final constraint a final altitude of 500 meters is required:

$$h(t_f) = 500.0$$

Figure 6 shows an example for an emergency trajectory.

This optimal control problem is again fully discretized using Eulers method and hence lead to sparse large-scale nonlinear optimization problem. The size of the problem is

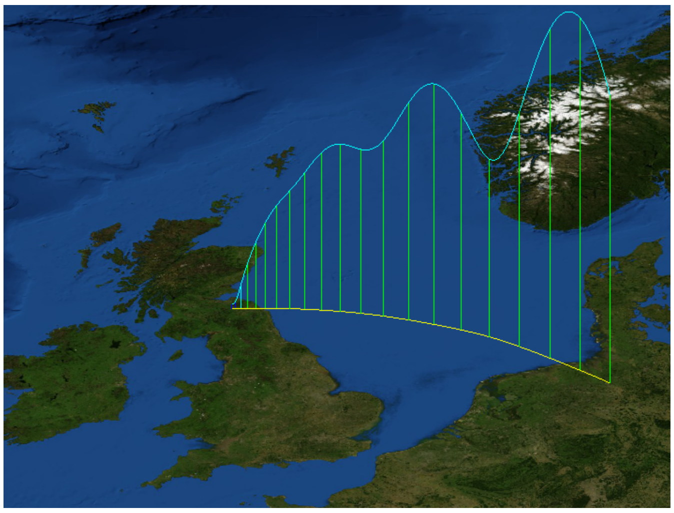


Figure 6: An emergency trajectory.

Table 3: List of different cases

n =	201	2001	5001	10001	20001	40001
N =	1609	16009	40009	80009	160009	320009
M =	2011	20011	50011	100011	200011	400011
WORHP 1.0	4.88s	45.47s	281.3s	424.25s	2261.35s	4898.53s
IPOPT 3.8.1	5.68s	Timeout	Timeout	382.57s	Timeout	5853.17s
KNITRO 6.0.0	MaxIter	MaxIter	Timeout	Timeout	Timeout	Timeout
SNOPT 7.2.8	28.73s	Error	Error	Timeout	Timeout	Timeout

determined by the number of points n used for the discretization of the problem. For the number of optimization variables N we get $N = 8 \cdot n + 1$ and for the number of constraints $M = 6 \cdot (n - 1) + 4 \cdot n + 7$. We tried different settings, the results are summarized in table 3. The computational time for this problem was limited to 2 hours and the precision for the constraints was set to 10^{-6} and the optimality conditions ito 10^{-5}.

WORHP is the only solver which is able to solve all variations of the problem. While IPOPT is slightly faster in the case of 10001 points WORHP is able to calculate a better objective function value and has also the best objective function value of all three solvers for 201 points.

These results can summarized in the following figure 7.

7 Conclusion

In this paper we presented the new nonlinear optimization solver WORHP. We described in a detailed way the main functionalities and advantages of WORHP. At the end of the

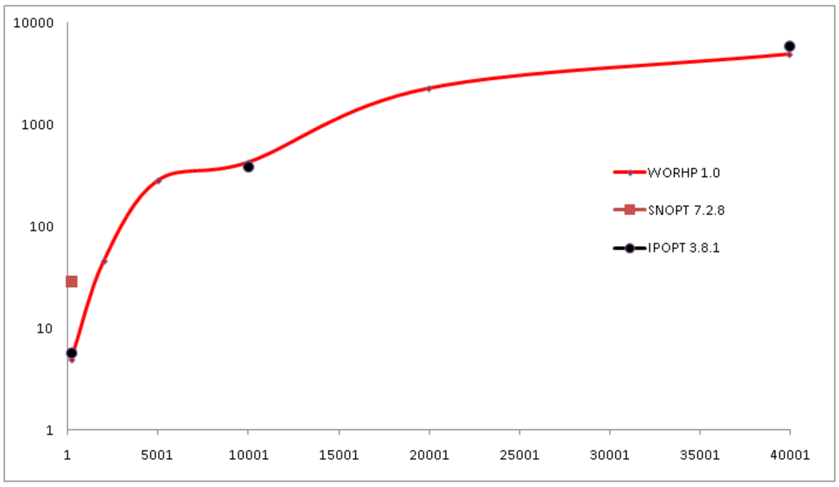


Figure 7: Number of points (x-axis) plotted against time in seconds (y-axis).

paper we have shown numerical test results and two applications, which demonstrated the capabilities of the new solver WORHP in comparison with the most used solvers for nonlinear optimization, showing that WORHP is not only able to solve academic test sets but also is very good in solving problems coming from applications.

Acknowledgments

WORHP was financially supported by the TEC-ECM group of the European Space Agency (ESA), the Bundesministerium für Wirtschaft und Technologie under contract 50 RL 0722 and the Steinbeis-Research Center (SFZ) Optimization and Optimal Control. The responsbility for the contents of this publication lies with the authors.

References

[1] Mangasarian, O. L. and Fromowitz, S., "The Fritz John Necessary Optimality Conditions in the Presence of Equality and Inequality Constraints." *Journal of Mathematical Analysis and Applications*, 1967, pp. 37–47.

[2] Fletcher, R., "An optimal positive definite update for sparse Hessian matrices," *SIAM Journal on Optimization*, Vol. 5, No. 1, 1995.

[3] Schittkowski, K., "On the convergence of a Sequential Quadratic Programming method with an augmented Lagragian line search function," *Mathematische Operationsforschung und Statistik, Series Optimization*, Vol. 14, 1983, pp. 197–216.

[4] Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization*, Academic Press, 1981.

[5] Gill, P. E., Murray, W., Saunders, M., and Wrigth, M. H., "Model Building and Practical Spects of Nonlinear Programming," *Computational Mathematical Programming*, edited by K. Schittkowski, Springer Berlin Heidelberg, 1985, pp. 209–47.

[6] Gertz, E. and Wright, S. J., "Object-oriented software for quadratic programming." *ACM Trans. Math. Softw.*, Vol. 29, No. 1, 2003, pp. 58–81.

[7] Betts, J. T., *Practical Methods for Optimal Control Using Nonlinear Programming*, SIAM Press, Philadelphia, Pennsylvania, 2001.

[8] Levenberg, K., "A Method for the Solution of Certain Non-linear Problems in Least-Squares," *Quarterly of Applied Mathematics*, Vol. 2, No. 2, jul 1944, pp. 164–168.

[9] Wächter, A. and Biegler, L. T., "On the Implementation of an Interior-Point Filter Line-Search Algorithm for Large-Scale Nonlinear Programming," *Mathematical Programming*, Vol. 106(1), 2006, pp. 25–57.

[10] Gill, P. E., Murray, W., and Saunders, M. A., "SNOPT: An SQP Algorithm For Large-Scale Constrained Optimization," *SIAM Journal on Optimization*, Vol. 12, 1997, pp. 979–1006.

[11] Kluever, C. A., "Optimal Feedback Guidance for Low-Thrust Orbit Insertion," *Optimal Control Applications and Methods*, Vol. 16, 1995, pp. 155–173.

[12] Büskens, C. and Maurer, H., "SQP-methods for Solving Optimal Control Problems with Control and State Constraints: Adjoint Variables, Sensitivity Analysis and real-Time Control." *Journal of Computational and Applied Mathematics*, 2000, pp. 85–108.

[13] Mayrhofer, M. and Sachs, G., "Notflugbahnen eines zweistufigen Hyperschall-Flugsystems ausgehend vom Trennmanöver," *Seminar des Sonderforschungsbereichs 255: Transatmospärische Flugsysteme*, München, 1996, pp. 109–118.

[14] Büskens, C. and Gerdts, M., "Emergency Landing of a Hypersonic Flight System: A Corrector Iteration Method for Admissible Real–Time Optimal Control Approximations," *Optimalsteuerungsprobleme in der Luft- und Raumfahrt, Workshop in Greifswald des Sonderforschungsbereichs 255: Transatmospärische Flugsysteme*, München, 2003, pp. 51–60.

[15] Büskens, C. and Gerdts, M., "Numerical Solution of Optimal Control Problems with DAE Systems of Higher Index," *Optimalsteuerungsprobleme in der Luft- und Raumfahrt, Workshop in Greifswald des Sonderforschungsbereichs 255: Transatmospärische Flugsysteme*, München, 2000, pp. 27–38.