Zentrum für Technomathematik
Fachbereich 3 – Mathematik und Informatik

Nonlinear Optimization in Space Applications with WORHP

Tim Nikolayzik
Christof Büskens Dennis Wassel

Report 11–10

Berichte aus der Technomathematik

Report 11–10 December 2011

NONLINEAR OPTIMIZATION IN SPACE APPLICATIONS WITH WORHP

Tim Nikolayzik
Universität Bremen, Germany, timn@math.uni-bremen.de

Dennis Wassel*, Christof Büskens†

Abstract

Nonlinear optimization has grown to a key technology in many areas of aerospace industry, e.g. satellite control, shape-optimization, aerodynamics, trajectory planning, reentry problems and interplanetary flights. These problems typically are discretized optimal control problems that give rise to large sparse nonlinear optimization problems. In the end all these different problems from various areas can be described in the general formulation as nonlinear optimization problems. The success of the optimization process depends on a multitude of factors, beginning at the modeling phase with the choice of the modeling approach and ending in the final interpretation and application of the outcomes, one the most crucial choices being the choice of a suitable optimization method. Despite the increase of computational power in recent years, methods not exploiting the special structure of these problems are likely to fail.

WORHP is designed to solve nonlinear optimization problems with more then hundred of millions variables and constraints. The algorithm is an SQP method and is exploiting the sparsity of the problem on every possible level: It includes efficient routines for computing sparse derivatives, e.g. graph-coloring methods for finite differences, sparse BFGS update techniques for Hessian approximations and sparse linear algebra. Furthermore WORHP uses reverse communication which allows the user to have full control of the optimization process.

In this paper we are going to introduce WORHP and the features described. Afterwards the results from the test campaign are going to be presented. In this test campaign WORHP has proven to be the most robust of all tested state-of-the-art nonlinear optimization solvers. Furthermore we will present a time optimal low-thrust planar transfer to a geosynchronous orbit and an emergency landing of a hypersonic flight system, both computed with WORHP.

I. 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. 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 kinds of solvers for large-scale sparse problems leads to unacceptably high computational effort and a higher probability of unsuccessful terminations.

To solve large-scale sparse problems one has to exploit as much information of the problem as possible. This includes an efficient storage of the problem 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* and *Shanno* (BFGS). These update techniques have several advantages, such as guaranteeing the positive definiteness of the Hessian approximation, so that further computations can be performed much easier. The biggest advantage is the efficiency of the calculation for small and medium sized problems. However, no sparsity can be exploited because the approximation of the Hessian is generally dense.

These limitations motivate 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 demonstrate the capabilities of the new proposed method.

II. NONLINEAR OPTIMIZATION

We consider nonlinear optimization problems of the form

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

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

*Universität Bremen, Germany, dwassel@worhp.de
†Universität Bremen, Germany, bueskens@worhp.de

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

$$\begin{aligned} I(\overline{x}) &:= \left\{ i \in \{M_e + 1, \ldots, M\} \mid G_i(\overline{x}) = 0 \right\}, \\ J(\overline{x}) &:= I(\overline{x}) \cup \{1, \ldots, M_e\}, \end{aligned}$$

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

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

where $\lambda \in \mathbb{R}^M$ 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(12)), then there exists $\overline{\lambda} \in \mathbb{R}^M$ such that the following holds:

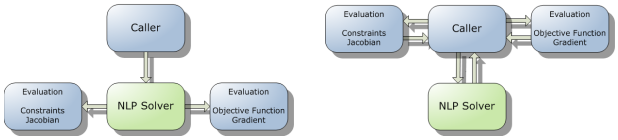
$$\begin{aligned} \nabla_x L(\overline{x}, \overline{\lambda}) = \nabla_x F(\overline{x}) + \overline{\lambda}^T \nabla_x G(\overline{x}) = 0 \\ \lambda_i \geq 0, \ i \in I(\overline{x}) \\ \lambda_j = 0, \ j \notin J(\overline{x}) \\ \lambda^T G(\overline{x}) = 0. \end{aligned} \qquad [1]$$

If additionally $\nabla_x G_i(\overline{x}), \ i \in J(\overline{x})$ are linearly independent then $\overline{\lambda}$ is unique. This criterion is called Linear Independence Constraint Qualification (LICQ) and is used to search for optimal solutions of [NLP].

III. METHODS FOR SOLVING NLP PROBLEMS

WORHP is a mixed SQP (Sequential Quadratic Programming) and IP (Interior-Point) method, that has been designed 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). SQP methods are producing a series of iterates by solving quadratic subproblems with linearized constraints. These subproblems are solved by using an interior-point method. 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



(a) Traditional calling convention ("Fire and Forget"). (b) Reverse Communication.

Figure 1: Different ways of calling an NLP solver.

an iterative method. It generates a sequence of points $\left\{x^{[k]}\right\}_{k=0,1,2,\ldots}$ with $x^{[k]} \overset{k\to\infty}{\longrightarrow} \overline{x}$ by:

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

where $d^{[k]} \in \mathbb{R}^N$ 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 infeasible for large-scale problems since the update formula generally yields dense update matrices, which in turn lead to dense Hessian approximations. Therefore, in the context of large-scale problems one is often forced to use the exact Hessian, which may be indefinite and leads to nonconvex quadratic programs.

Alternative attempts use limited memory BFGS updates or sparse update formulas, cf. Fletcher (5). Both methods are handicapped by their local character, hence 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 on a merit function, which is usually given by an exact penalty function such as the L_1-penalty function or the augmented Lagrangian function. For more details cf. Schittkowski (14), Gill, Murray and Wright (7) and Gill et al. (8).

Computation of derivatives is a crucial element in nonlinear optimization. Basically first derivatives, i.e. the gradient of the objective function and the Jacobian of the constraints are necessary in order to find a descent direction towards the point where the closest local minimum is expected. Second derivatives (Hessian of the Lagrangian) are used to enable quadratic convergence behavior, and to determine a stepsize. There are different ways to calculate these derivative information; WORHP provides several of them: The solver includes finite differences (FD) methods and WORHP can use truly sparse BFGS update techniques. The FD-module uses a so-called "group strategy" based on the graph-coloring theory to drastically speed up

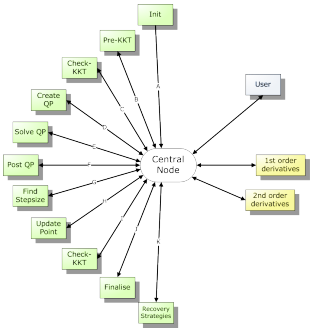


Figure 2: Data flow in WORHP

the calculations for sparse problems by requiring fewer (in many cases far fewer) function evaluations than naïve approaches.

IV. WORHP

Since WORHP is an iterative solver, it produces a sequence of points $\left\{x^{[k]}\right\}_{k=0,1,2,\ldots}$ by the following basic algorithmic scheme:

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 according to [2], increment k and go to i.

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

Instead of assuming the restrictive formulation used in [NLP], WORHP accepts the more flexible (but equivalent) problem formulation

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

where $G(x) = \left(G_1(x), \ldots, G_M(x)\right)^T$ and $l, u \in \mathbb{R}^M$.

IV.I 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.

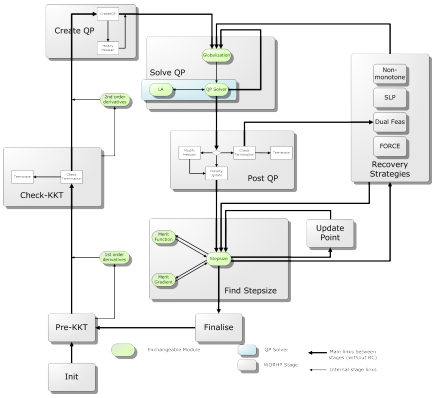


Figure 3: WORHP

Each call of the solver carries out part of a major NLP iteration, called *stage*. Forming a superset of the algorithmic steps outlined above, some central stages of WORHP 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 $d^{[k]}$.

All stages together form the SQP method. The general workflow of WORHP resembles that of other SQP methods, see Figure 3.

IV.II Checking for optimality

For testing an iterate $x^{[k]}$, the first order necessary optimality conditions [1] have to be evaluated. First order derivatives are required for this test. These derivatives can be provided by the user, or the user can have WORHP approximate them by finite differences. In the following we denote with $F^{[k]}$ and $G^{[k]}$ evaluations of the objective funtion F and the constraints G at the point $x^{[k]}$. The iterate $x^{[k]}$ is said to be optimal if the following holds

$$\nabla_x F^{[k]} + \lambda^{[k]T} \nabla_x G^{[k]} \le \epsilon_{\text{opti}}, \qquad [3]$$

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

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

and

$$\begin{aligned} |G_i^{[k]}| &\le \epsilon_{\text{feas}}, \ i = 1, \ldots, M_e, \\ G_j^{[k]} &\le \epsilon_{\text{feas}}, \ j = M_e + 1, \ldots, M. \end{aligned} \qquad [6]$$

WORHP also supports a scaled version of the original KKT-conditions. These conditions are motivated by the idea that the numerical optimality by [3] is difficult to interpret for the user, who is actually interested in

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

This leads to

$$||\nabla_x L^{[k]}||_\infty \leq \frac{\epsilon_{\text{opt}} \max\left(1, |F^{[k]}|\right) + ||\lambda^{[k]} G^{[k]}||_\infty}{||d^{[k]}||_\infty}. \quad [7]$$

In order to prevent WORHP from iterating too long without satisfying [7], e.g. due to inexact derivative approximations, a low pass filter is implemented. Among others, this filter calculates two thresholds

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

and

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

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

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

and

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

are checked. If both conditions are fulfilled and the current iterate $x^{[k]}$ is feasible, this point is assumed to be optimal, since no more progress has been achieved with the given inputs. If the current iterate $x^{[k]}$ is not feasible, a feasibility mode is activated, cf. VI.

IV.III 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 quadratic problem (QP) is

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

To formulate the QP as subproblem for solving [NLP], the exact Hessian or an approximation by BFGS update formulas is needed. Again, the user can provide this information or have WORHP approximate it.

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

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

$$G_i^{[k]} + \nabla_x G_i^{[k]} d = 0, \quad [10]$$

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

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

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 is an appropriate search direction and $\overline{\lambda}^{[k]}$ approximates the Lagrange multipliers of [NLP]. If the QP is non-convex, regularization techniques of Levenberg-Marquardt type are used to convexify the problem, cf. section V.

Further problems which might occur are inconsistencies in the linearized constraints, hence for the practical implementation a relaxed formulation of [8] is solved with respect to $z = (d^{[k]}, \delta^{[k]}) \in \mathbb{R}^{N+1}$:

$$\begin{aligned} \min_z \quad & \nabla_x F^{[k]} d^{[k]} + \tfrac{1}{2} d^{[k]T} B^{[k]} d^{[k]} + \tfrac{\eta_r}{2} \delta^{[k]}, \\ \text{s.t.} \quad & G_i^{[k]}(1 - \delta^{[k]}) + \nabla_x G_i^{[k]} d^{[k]} = 0, \; i \in \mathcal{I}, \\ & G_j^{[k]}(1 - \sigma_i \delta^{[k]}) + \nabla_x G_j^{[k]} d^{[k]} \leq 0, \; j \in \mathcal{J}, \end{aligned}$$

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

$$\sigma_j = \begin{cases} 0, & \text{if } G_j^{[k]} < 0, \\ 1, & \text{otherwise,} \end{cases}, j \in \mathcal{J},$$

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

$$\begin{aligned} \mathcal{I} &:= \{1, \ldots, M_e\}, \\ \mathcal{J} &:= \{M_e + 1, \ldots, M\}. \end{aligned}$$

IV.IV Merit Function

In order to achieve global convergence, we have to find an appropriate step size $\alpha^{[k]}$ for the solution $d^{[k]}$ of the QP. To this end, one has to measure the progress of the optimization process, which consists of a scalar quantification of both the objective and the constraints. For this purpose a merit function is used. Merit functions supported by WORHP are the L_1-penalty function

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

and the augmented Lagrangian

$$L_a(x,\lambda;\eta) := f(x) + \sum_{i=1}^{M_e} \lambda_i G_i(x) + \tfrac{1}{2}\sum_{i=1}^{M_e} \eta_i G_i^2(x) + \tfrac{1}{2\eta_i}\sum_{i=M_e+1}^{M} \Big(\big(\max\{0, \lambda_i + \eta_i G_i(x)\}\big)^2 - \lambda_i^2 \Big),$$

where $\eta \in \mathbb{R}^M$, with $\eta_i \geq 0, i = 1, \ldots, M$, is a penalty vector.

IV.V Hessian Regularization

To guarantee that the solution of the QP is unique and reasonable one has to ensure 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

$$H = H_L + \tau(|\sigma| + 1)I. \qquad [11]$$

The parameter $\tau \in [0, 1]$ is used to dampen the Hessian update, and σ is the Gerschgorin bound for the most negative eigenvalue 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 (11). 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 positive definiteness of the Hessian but impairs convergence, since it may cause a large perturbation of the original Hessian. On the other hand, $\tau = 0$ may cause problems in the QP solver, since in this case the original Hessian without regularization is used. The idea of Betts (1) 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 $M^{[k]} = M(x^{[k]}, \lambda^{[k]})$ the value of one of the merit functions introduced in section IV in the k-th iteration. Three quantities have to be computed.

The actual reduction:

$$\rho_1 = M^{[k-1]} - M^{[k]}. \qquad [12]$$

The predicted reduction:

$$\begin{aligned}\rho_2 &= M^{[k-1]} - \tilde{M}^{[k]} \\ &= -\tfrac{\mathrm{d}}{\mathrm{d}\alpha} M_0^{[k]} - \tfrac{1}{2} d^{[k]T} H d^{[k]},\end{aligned} \qquad [13]$$

where $\tilde{M}^{[k]}$ is the predicted value of the merit function, computed from the first derivative of the merit function $\frac{\mathrm{d}}{\mathrm{d}\alpha} M_0^{[k]}$ with respect to the step size α evaluated at $\alpha = 0$ and second-order information involving the Hessian matrix.

Finally, we need to compute the rate of change in the norm of the gradient of the Lagrangian

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

where the error in the gradient of the Lagrangian is

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

If $\rho_1 \leq 0,25\rho_2$ holds, than in the next iteration τ is increased by

$$\tau^{[k+1]} = \min(\theta \cdot \tau^{[k]}, 1),$$

with $\theta > 1$ a threshold which has to be chosen suitable. If instead $\rho_1 \geq 0,75\rho_2$ holds, τ is decreased by

$$\tau^{[k+1]} = \tau^{[k]} \min\left(\frac{1}{\theta}, \rho_3\right).$$

IV.VI Line Search

After determining the search direction $d^{[k]}$ from the QP, we have to find a suitable step size $\alpha^{[k]}$. To this end we use a line search method based on the Armijo Rule. We define

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

whereas $\lambda(\alpha)$ is an update of the multipliers depending on α, for instance $\lambda^{[k+1]}(\alpha) = (1-\alpha)\lambda^{[k]} + \alpha\lambda_{QP}$, and M is one of the merit functions introduced in section IV.

In general, a good choice would be an α which exactly minimizes $\phi(\alpha)$, but unfortunately this defines another nonlinear optimization problem. Although the original problem is reduced to a one-dimensional line search, it is too expensive to solve, especially with large-scale problems. Thus, a more realistic goal is to find the largest step size α that satisfies

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

by using an inexact line search:

Starting with a maximum step size $\alpha_{\max} \in (0, 1]$ and a decrease factor $\beta_{\text{armijo}} \in (0, 1)$, candidates for the step size are

$$\{\alpha_j = \beta^j \alpha_{\max} \mid j = 0, 1, 2, \ldots, l_{\max}\},$$

where $l_{\max} = \max\{l \in \mathbb{N} \mid \beta^l \alpha_{\max} \geq \alpha_{\min}\}$ defines the smallest step size allowed. As the first trial step size we choose $\alpha_0 = \alpha_{\max}$. If the Armijo condition

$$\phi(\alpha_j) \leq \phi(0) + \sigma\alpha_j\phi'(0), \qquad [15]$$

where $\sigma \in \mathbb{R}^+$ is a suitable factor and $\phi'(0)$ the derivative of ϕ with respect to α, is not satisfied, $\alpha_1 = \beta^1 \alpha_0$ is calculated and [15] is checked again. This is done iteratively until either a suitable α is found, or the line search fails when $i > l_{\max}$. In case of failure, WORHP uses recovery strategies to prevent the NLP algorithm from failing. Several recovery strategies are implemented.

SLP: This strategy is intended to recover from QP-solver failures and is motivated by gradient methods: Instead of using a second-order approximation of the Hessian in [8], the identity matrix is used, which has stabilizing properties.

Feasible mode: If the line search has failed at an infeasible point, this mode is a good choice for recovering. In this mode the QP is modified extensively to provide a search direction that focusses on the feasibility of the problem. The mode will be stopped after a feasible point has been found. Afterwards the normal optimization is resumed at the new, now feasible, iterate.

IV.VII The algorithm of WORHP

Next we state the algorithm in detail:
Algorithm: Given an initial guess $x^{[0]}$ and a set of constants including ϵ_{opti}, $\epsilon_{\text{feas}} > 0$, $\epsilon_{\text{comp}} > 0$, $\varepsilon > 0$, $\beta_{\text{armijo}} \in (0,1)$, $\alpha_{\max} \in (0,1]$, $\rho_r > 1$ and others,

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 V.

A-5: *Solve-QP.*

- **A-5.1:** If the QP was not solved successfully, check if $\tau < 1$. If this is the case go to **A-4**.
- **A-5.2:** If $k = 0$ and $M > 0$ set $\lambda^{[0]} = \lambda_{QP}$ go to **A-3**.
- **A-5.3:** If QP 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 $\eta_r = \rho_r \eta_r$ 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 currently 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 so 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 $\lambda^{[k]}$.
- **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**.

IV.VIII Interfaces

WORHP currently offers eight 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.

Interfaces are divided into three classes for use in different environments:

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

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

Library interfaces: Three interfaces for inclusion of WORHP 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.

To solve the sparse large-scale linear systems in the quadratic subproblem solver, various third-party linear algebra packages can be used by WORHP; their actual usability is subject to availability and licensing conditions of the third-party software providers:

- LAPACK (public domain, dense fallback solver)
- SuperLU (BSD-style license, default solver)
- MA48
- MA57
- MA86
- PARDISO
- MUMPS
- WSMP

V. RESULTS

The robustness of WORHP is demonstrated on several test sets. We present numerical results for the AMPL-version of the CUTEr test set, consisting of a collection of 920 medium-scale sparse and small dense problems. As reference solvers IPOPT 3.9.2 with MA57 (15) as well as KNITRO 7.0.0 and SNOPT 7.2-8 (9) were used. Test were performed using the AMPL optimization environment. Tables 1 and 2 show 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 problems. The computational time for a single problem of the test set was limited to 30 minutes, the maximum number of iterations limited by 10,000 and the tolerances for the constraints and the optimality conditions were both set to 10^{-6}. Testing was performed on a Linux system with an Intel Core2 Quad CPU Q6600 at 2.40GHz, with 4GB RAM.

	WORHP	IPOPT
Version	1.0	3.9.2
Problems solved	918	877
Optimal solution found	911	869
Acceptable solution found	7	8
Not solved	2	43
Percentage	99.78	95.33
Time	5,060s	27,056s

Table 1: Summary of the results of WORHP and IPOPT.

WORHP is capable of solving more than **99.7**% of the problems of the CUTEr test set. The 3 problems which WORHP was not able to solve were also not solved by any of the other solvers. We think that this is an issue with the problem formulation, such as an empty feasible set, which renders these problems unsolvable. IPOPT was able to solve about **95.3**% of all the problems while KNITRO solves **96.4**% and SNOPT **89.9**%. WORHP is the fastest of all solvers with respect to the overall computational time. The relatively large overall computational times for KNITRO and SNOPT are a result of unsuccessful terminations by timeout 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 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 and KNITRO are close while SNOPT and IPOPT are little bit behind. One should keep in mind, that within the first ten seconds only small and medium sized problems are solved while the remaining larger problems take more time. When it comes to the more larger problems we see that WORHP, KNITRO and IPOPT are almost at the same level, just SNOPT is falling behind, but this in not surprising since SNOPT is only solver using a BFGS approximation for the second derivative of the Hessian of the Lagrangian.

	KNITRO	SNOPT
Version	7.0.0	7.2-8
Problems solved	887	827
Optimal solution found	882	810
Acceptable solution found	5	17
Not solved	33	93
Percentage	96.41	89.89
Time	32,792s	49,569s

Table 2: Summary of the results of KNITRO and SNOPT.

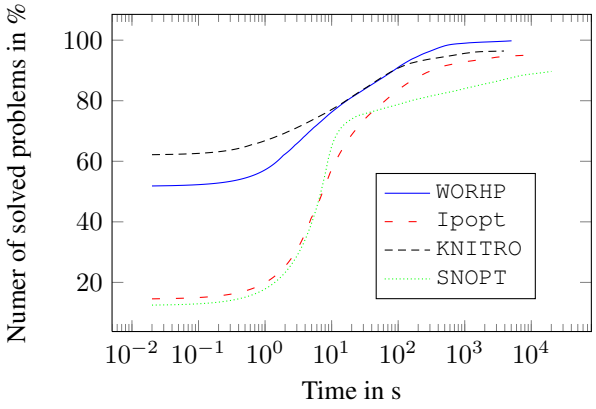


Figure 4: Percentage of optimally solved problems within given time frame.

VI. TWO EXAMPLES

The following two examples are full discretized optimal control problems. The calculations were also done on a Linux system with an Intel Core2 Quad CPU Q6600 at 2.40GHz with 4GB RAM.

VI.I Time-optimal low-thrust planar transfer to GEO

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

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

with the initial and final conditions

$$\begin{aligned}
x_1(0) &= 6.0, & x_1(t_f) &= 6.6 \\
x_2(0) &= 0.0, & x_2(t_f) &= 0.0 \\
x_3(0) &= \sqrt{\frac{r_\mu}{x_1(0)}}, & x_3(t_f) &= \sqrt{\frac{r_\mu}{x_1(t_f)}} \\
x_4(0) &= 0.0,
\end{aligned}$$

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 (10).

This optimal control problem is fully discretized using Euler's method, which transforms it into a sparse nonlinear optimization problem, cf. Büskens and Maurer (4). The size of the problem is determined by the number of points used for the discretization of the problem. Using n discrete points, the resulting nonlinear optimization problem has $5 \cdot n + 1$ optimization variables and $4 \cdot (n-1) + 7 + 2$ nonlinear constraints. We see in table 3 a summary of the different nonlinear optimization problems arising from different vlaues of n. The computational time for this problem was limited to 30 minutes and the precision for the constraints and the optimality conditions is set to 10^{-6}, the maximum number of iterations is again set to 10,000.

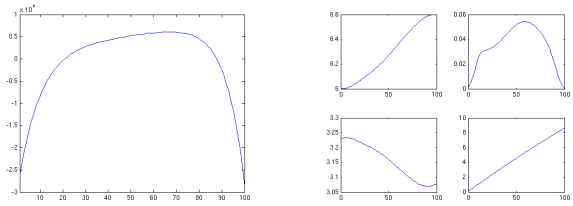
The optimal control and the optimal states of a solution are shown in figure 5.

The results of all solvers are summarized in the tables 4 and 5.

We see that WORHP together with KNITRP is only solver capable of solving all five problems. As also stated in the previous chapter the speed of WORHP is comparable to the speed of IPOPT and KNITRO, while SNOPT has problems with the high-dimensional problems.

n	N	M
101	506	409
1,001	5,006	4,009
5,001	25,006	20,009
10,001	50,006	40,009
40,001	200,006	160,009

Table 3: Number of optimization variables and constraints



(a) Optimal control $u(t)$. (b) Optimal states $x_i(t).i = 1 \ldots 4$.

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

n	WORHP	IPOPT
101	$0.88s$	$0.32s$
1,001	$3.83s$	$7.12s$
5,001	$145.55s$	$33.11s$
10,001	$194.73s$	Timeout
40,001	$939.58s$	$741.13s$

Table 4: Results of WORHP and IPOPT

n	KNITRO	SNOPT
101	$0.28s$	$0.54s$
1,001	$3.31s$	$102.68s$
5,001	$93.41s$	$374.30s$
10,001	$252.58s$	$1786.85s$
40,001	$312.62s$	Timeout

Table 5: Results of KNITRO and SNOPT

VI.II Emergency landing of a hypersonic flight system

In this example the emergency landing of a two-stage space transport vehicle is investigated. After the separation of the first stage the engine of the upper stage suffers an ignition failure, and due to this propulsion damage the upper stage cannot reach a safe orbit. For more details see Mayrhofer and Sachs (13) or Büskens and Gerdts (2, 3).

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

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

The abovementioned functions are defined as

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

The constants are chosen as

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

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

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

The mass is supposed to be constant m =115,000. 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 roughly 33 km 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.

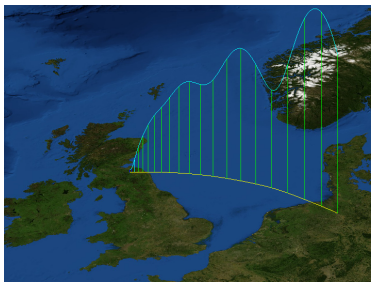


Figure 6: An emergency trajectory.

This optimal control problem is again fully discretized using Euler's method to transform it into a sparse large-scale nonlinear optimization problem. The size of the problem is determined by the number of points n used for the discretization of the problem. For the number of optimization variables N we have $N = 8\cdot n + 1$ and for the number of constraints $M = 6\cdot(n-1) + 4\cdot n + 7$. We used a various number of values for n. The size of the resulting nonlinear optimization problems can be found in table 6.

The reuslts for the four solvers can be found in the tables 7 and 8. The computational time for this problem was limited to 30 minutes and the precision for the constraints and the optimality conditions is set to 10^{-6}, the maximum number of iterations is again set to $10,000$.

n	N	M
201	1,609	2,011
2,001	16,009	20,011
5,001	40,009	50,011
10,001	80,009	100,011
20,001	160,009	200,011
40,001	320,009	400,011

Table 6: Number of optimization variables and constraints

n	WORHP	IPOPT
201	$2.49s$	$8.79s$
2,001	$59.30s$	MaxIter
5,001	$146.80s$	Timeout
10,001	$184.08s$	Timeout
20,001	$591.61s$	Timeout
40,001	$1477.32s$	Timeout

Table 7: Results of WORHP and IPOPT

WORHP is the only solver which is able to solve all variations of the problem. KNITRO is not able to solve any of the problems, while IPOPT and SNOPT are solving at least the smallest of the problems, but WORHP is by far the fastest solver.

VII. 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 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.

n	KNITRO	SNOPT
201	MaxIter	$28.73s$
2,001	Error	Error
5,001	Timeout	Error
10,001	Timeout	Timeout
20,001	Timeout	Timeout
40,001	Timeout	Timeout

Table 8: Results of KNITRO and SNOPT

Acknowledgments

WORHP was financially supported by the TEC-ECM group of the European Space Agency (ESA) in the project eNLP (contract number 21293/07/LL/ST together with Astos Solutions) and eNLPext (contract number 4000102529 together with Astos Solutions), the Steinbeis-Research Center (SFZ) Optimization and Optimal Control, Bundesministerium für Bildung & Forschung (grants 50RL0722 and 50JR0688)

References

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

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

[3] Christof Büskens and Matthias Gerdts. Numerical Solution of Optimal Control Problems with DAE Systems of Higher Index. In *Optimalsteuerungsprobleme in der Luft- und Raumfahrt, Workshop in Greifswald des Sonderforschungsbereichs 255: Transatmospärische Flugsysteme*, pages 27–38, München, 2000.

[4] Christof Büskens and Helmut Maurer. 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*, pages 85–108, 2000.

[5] Roger Fletcher. An optimal positive definite update for sparse Hessian matrices. *SIAM Journal on Optimization*, 5(1), 1995.

[6] E. Michael Gertz and Stephen J. Wright. Object-oriented software for quadratic programming. *ACM Trans. Math. Softw.*, 29(1):58–81, 2003.

[7] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Academic Press, 1981.

[8] Philip E. Gill, Walter Murray, M.A. Saunders, and Margaret H. Wrigth. Model building and practical

spects of nonlinear programming. In Klaus Schittkowski, editor, *Computational Mathematical Programming*, pages 209–47. Springer Berlin Heidelberg, 1985.

[9] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12:979–1006, 1997.

[10] Craig A. Kluever. Optimal feedback guidance for low-thrust orbit insertion. *Optimal Control Applications and Methods*, 16:155–173, 1995.

[11] Kenneth Levenberg. A method for the solution of certain non-linear problems in least-squares. *Quarterly of Applied Mathematics*, 2(2):164–168, jul 1944.

[12] Olvi L. Mangasarian and S. Fromowitz. The Fritz John necessary optimality conditions in the presence of equality and inequality constraints. *Journal of Mathematical Analysis and Applications*, pages 37–47, 1967.

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

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

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