ADAPTIVE PARETO FRONT SAMPLING BASED ON PARAMETRIC SENSITIVITY ANALYSIS IN A BI-OBJECTIVE SETTING

Arne Berger, Christof Büskens

University of Bremen
Center for Industrial Mathematics, WG Optimization and Optimal Control
Bibliothekstrae 5, 28359 Bremen, Germany
{aberger,bueskens}@math.uni-bremen.de

ABSTRACT

In practice different goals usually contradict each other. In this case one aims to find optimal compromises, which leads to the computation of the so called Pareto front. This paper shows how a discrete approximation of a Pareto front can be refined with polynomial interpolation. For this we exploit the information given by the discrete samples of the Pareto front and in addition we use parametric sensitivity information from these samples. The interpolation is afterwards used to determine the best point to compute another sample in order to get the most information. Results are shown for a bi-objective example.

Index Terms— Multiobjective optimization, Pareto front interpolation, nonlinear programming, adaptive stepsize

1. INTRODUCTION

Practical space engineering problems often deal with different, possibly conflicting objective functions. This leads to the formulation of multiobjective non-linear problems. In order to solve these problems, one usually solves multiple scalarized subproblems. This provides a discrete approximation of the Pareto front which gives useful information for the decision maker who, in praxis, has to select one single solution. If the desired solution is not part of the precomputed discrete approximation one needs to apply interpolation techniques.
This contribution shows a method which uses information from parametric sensitivity analysis of the scalarized subproblems in order to choose the stepsize between samples adaptively to obtain a better interpolation between precomputed solutions. The problems are solved with the NLP solver WORHP which provides sensitivity information in an efficient way by reusing the factorization of the KKT matrix of the last optimization iteration. We show the basic functionality of the presented method by applying it to a bi-objective optimization problem. The method can also be used for more than two objectives if one can identify the neighboring precomputed points which are then used for interpolation.

2. PRELIMINARIES

2.1. Pareto Optimality

Practical engineering problems often deal with different, possibly conflicting objective functions. This leads to the formulation of a multiobjective nonlinear program (MONLP) :

$$\begin{array}{ll} {''}\min\limits_{x\in\mathbb{R}^{N_x}}{''} & F(x) = (F_1(x), F_2(x), \dots, F_{N_F}(x))^T \\ \text{subject to} & g(x) \leq 0, \\ & h(x) = 0, \end{array} \tag{1}$$

with $N_F \geq 2$ and all $F_i : \mathbb{R}^{N_x} \to \mathbb{R}$, $g : \mathbb{R}^{N_x} \to \mathbb{R}^{N_g}$ and $h : \mathbb{R}^{N_x} \to \mathbb{R}^{N_h}$ are twice continuously differentiable. Since generally no single x would minimize all F_i simultaneously, an extended concept of optimality is needed. The most popular one is that of Pareto optimality that gives the definition for so called efficient points.
The vector $F(\hat{x})$ is said to dominate the vector $F(\bar{x})$ if and only if $F_i(\hat{x}) \leq F_i(\bar{x})$ for all $i = 1, \dots, N_F$, and $F_j(\hat{x}) < F_j(\bar{x})$ for at least one $j \in \{1, \dots, N_F\}$. We denote this by $F(\hat{x}) \prec F(\bar{x})$. Using this principle of dominance, a point $x^* \in \mathbb{R}^{N_x}$ is called *globally efficient* or *globally Pareto optimal* for (1) if and only if there exists no other point x that satisfies $F(x) \prec F(x^*)$. The vector $F(x^*)$ is called globally non-dominated. A point $x^* \in \mathbb{R}^{N_x}$ is called *locally efficient* or *locally Pareto optimal* for (1) if and only if there exists a subset $U_{x^*} \subset \mathbb{R}^{N_x}$, such that $x^* \in U_{x^*}$ and there exists no $x \in U_{x^*}$ with $F(x) \prec F(x^*)$. The vector $F(x^*)$ is called locally non-dominated.
The set $P = \{F(x) \;:\; x \text{ is globally efficient for (1)}\}$ is called the *Pareto front*.

2.2. Pascoletti-Serafini Scalarization Approach

One way to solve multiobjective optimization problems is to formulate a scalar valued objective problem which corresponds to the original problem. Usually the solution of the scalarized problem is at least guaranteed to be a weak Pareto optimal point. This sections explains one approach and how it is possible to sample the whole Pareto front with a scalarized problem.

The Pascoletti-Serafini approach is one way to scalarize problems of the form (1) [1]. It is a very broad approach where Pareto optimality is defined by convex cones $\mathcal{C}$. However if one uses

$$\mathcal{C} = \{v \in \mathbb{R}^{N_F} : v_i \geq 0,\ \forall\, i = 1, \ldots, N_F\} \tag{2}$$

the formulation is equivalent to the one in Section 2.1 [2]. Thus we will use this definition for the reminder of this paper. It leads to the scalar valued optimization problem

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^{N_x},\ t \in \mathbb{R}} & t \\ \text{subject to} & F(x) - (a + tr) \leq 0 \\ & g(x) \leq 0 \\ & h(x) = 0, \end{array} \tag{3}$$

where $a, r \in \mathbb{R}^{N_F}$ are hyperparameter which can be varied to optain different Pareto optimal solutions. Pascoletti and Serafini show that for a fixed parameter r^0 and for every solution $F(x^*)$ which is Pareto optimal there exists a parameter a^* such that $F(x^*)$ is the solution of (3) for these parameters. In other words, for a fixed value of r one is able to find every point on the Pareto front by varying a. It is even possible to find all points by varying a along a hyperplane which is the usual pratice (see [2] for details).

2.3. Parameter Perturbed Multiobjective Optimization

We use the Pascoletti-Serafini approach for the scalarization which depends on the hyper parameter a and r and introduce the perturbation parameter $p \in \mathbb{R}^{N_F}$ with $p_0 = 0$ which can be interpreted as a variation in a to the problem (3):

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^{N_x},\ t \in \mathbb{R}} & t =: \hat{f}(x, t, p) \\ \text{subject to} & F(x) - (a + p + tr) \leq 0 \\ & g(x) \leq 0 \\ & h(x) = 0 \end{array} \tag{4}$$

Based on the theory of parametric sensitivity analysis we get the following sensitivity differentials [3]

$$\frac{dx}{dp}(p_0), \frac{d\hat{f}}{dp}(p_0), \frac{dg}{dp}(p_0), \frac{dh}{dp}(p_0). \tag{5}$$

Together with equation (4) and E_{N_F} as the identity matrix of dimension N_F we also get (see [2])

$$\frac{dF}{dp}(p_0) = E_{N_F} - r\left(\frac{d\hat{f}}{dp}(p_0)\right)^T. \tag{6}$$

It is therefor possible to calculate the parametric sensitivities of F with respect to p and thus one gets the information of how F changes when a is varied. Before we show how this can be used to interpolate the Pareto front we first introduce multiobjective optimal control problems.

2.4. Multiobjective Optimal Control

Equivalent to (1) one can also formulate a multiobjective optimal control problem of the form

$$\begin{array}{lrcll} \min\limits_{u,x,t_f} & F(x(t)) & \in & \mathbb{R}^{N_F} & \\ \text{s.t.} & \dot{x}(t) & = & \xi(x(t), u(t), t), & t \in [0; t_f] \\ & \omega(x(0), x(t_f)) & = & 0 & \\ & C(x(t), u(t), t) & \leq & 0 & t \in [0; t_f], \end{array} \tag{7}$$

$$F_i = M_i(x(0), x(t_f)) + \int_0^{t_f} I_i(x(t, u(t), t))dt,\ i = 1, \ldots, N_F.$$

One approach to solve problems of the type (7) is to discretize them in order to optain a problem of the form (1) which than can be solved with the same methods. For our results we use the library for direct transcription methods TransWORHP [4].

3. ADAPTIVE PARAMETER CHOICE

In order to compute Pareto optimal solutions we apply the Pascoletti Serafini approach. Afterwards we interpolate the Pareto front based on sensitivity information and then calculate further solutions with an adaptive choice of the hyperparameter a. This section defines the necessary steps for this procedure.

3.1. Pareto front interpolation

With the results from Section 2.3 we can now interpolate the Pareto front. For this define $\alpha := a + p$ with a and p as in (4). Since $\frac{d\alpha}{dp} = 1$ we get that

$$\begin{aligned} \frac{dF}{dp} &= \frac{dF}{d\alpha} \text{ and} \\ \frac{dx}{dp} &= \frac{dx}{d\alpha}. \end{aligned} \tag{8}$$

Furthermore we define

$$\begin{aligned} &\hat{x}(\alpha) := x \text{ is the solution of (4) for } \alpha, \\ &\hat{F}(\alpha) := F(\hat{x}(\alpha)). \end{aligned} \tag{9}$$

In the following we consider only bi-objective settings ($N_F = 2$). In this case the variation of the hyperparameter a can be described as a function $a(t) : \mathbb{R} \to \mathbb{R}^2$. Depending on t the Pareto front is now a curve in $\mathbb{R}^2$ with coordinates $\xi_1 = \hat{F}_1(\alpha(t))$ and $\xi_2 = \hat{F}_2(\alpha(t))$. It follows that

$$\frac{d\xi_2}{d\xi_1} = \frac{\frac{dF_2}{d\alpha_1}}{\frac{dF_1}{d\alpha_1}}. \tag{10}$$

With this information we can approximate $\hat{F}$ by applying a polynomial interpolation of 3rd grade between each neighbouring samples. Based on the additional information about

$\frac{dx}{dp}$ the same procedure can be applied in variable spaces such that we get $\hat{x}(\alpha)$. The resulting approximations we call $\hat{F}^{\text{[approx]}}$ and $\hat{x}^{\text{[approx]}}$. If one aims not only for the shape of the Pareto front but also for explicit solutions it will be necessary to apply feasibility refinement afterwards since the interpolation is not guaranteed to yield feasible soultions. For details about this see [5].

3.2. Refine Interpolation Support Points

Based on the interpolation described in Section 3.1 our goal is now to find further Pareto optimal points which give the most additional information for a refined interpolation. To reach this goal we first make an observation:
If one wants to display the shape of the Pareto front using the interpolation of Section 3.1 there are two possibilities. The first is to display $\hat{F}^{\text{[approx]}}$ the second is to approximate $\hat{x}$ and afterwards calculate $F(\hat{x}^{\text{[approx]}})$. One can than observe that usually $\hat{F}^{\text{[approx]}}$ differs from $F(\hat{x}^{\text{[approx]}})$. The exception of this is when the objective functions in F are all identical to some value in x or the negative of one. In order to calculate the error made by these approximations we compute another set of $N^S \in \mathbb{N}_+$ samples, which we call S with F^i as the ith element in S. We define the error functions by calculating the difference between a point in H an. We define

$$\begin{aligned}
\alpha_i^* &:= \alpha \in \mathbb{R}^2 : \hat{F}(\alpha^*)_1 = F_1^i,\ i = 1, \dots, N^S \\
x_i^* &:= x \in \mathbb{R}^{N_x} : F(x^*) = F_1^i, i = 1, \dots, N^S \\
\Delta_i^{\hat{F}} &:= |\hat{F}(\alpha_i^*)_2 - F_2^i| \\
\Delta^{\hat{F}} &:= \{\Delta_i^{\hat{F}},\ i = 1, \dots, N^S\} \\
\Delta_i^{Fx} &:= |F(x_i^*)_2 - F_2^i|,\ i = 1, \dots, N^S \\
\Delta^{Fx} &:= \{\Delta_i^{Fx},\ i = 1, \dots, N^S\}
\end{aligned} \tag{11}$$

as the errors made by the approximation by comparing F_2 for specific values of F_1. Additionally we define the distance between both approximations as

$$\begin{aligned}
\delta_i &:= |F(x_i^*)_2 - \hat{F}(\alpha_i^*)_2|,\ i = 1, \dots, N^S \\
\delta &:= \{\delta_i,\ i = 1, \dots, N^S\}
\end{aligned} \tag{12}$$

If one take the signs of Δ^{Fx} and $\Delta^{\hat{F}}$ into account it follows that δ is the result of adding both errors. It follows that

$$\Delta_i^{\hat{F}} = 0 \ \&\ \Delta_i^{Fx} = 0 \Rightarrow \delta_i, \quad \forall\, i = 1 \dots, N^S, \tag{13}$$

leading to the idea that the point, where both approximations differ the most will be a good position to compute the next sample of the Pareto front.

4. IMPLEMENTATION

4.1. Example Problem

For the illustration of our approach we use a very simple model of a car which drives on a hilly territory decribed by:

$$\begin{array}{lrcl}
\min\limits_{u,x,t_f} & F(x,u,t_f) & = & (-x_1(t_f), x_3(t_f)^2)^T \\
\text{s.t.} & \dot{x}_1(t) & = & x_2(t) \\
& \dot{x}_2(t) & = & u(t) - \dot{\tau}(x(t)) \\
& \dot{x}_3(t) & = & u(t)^2 \\
& x_1(0) & = & 0 \\
& t_f & = & 1 \\
& u(t) & \geq & 0 \\
& u(t) & \leq & 5
\end{array} \tag{14}$$

The objetives are to maximize the distance the car can drive in a certain time and to minimize the energy needed. The territory is described by the differentiable function $\tau : [0, \infty) \to \mathbb{R}$. We implement this function as a piecewise polynomial function τ^p which is described on the interval $[0; 1]$ by support points at which τ^p and $\dot{\tau}^p$ are given. To cover the whole interval $[0, \infty)$ we continue the function as

$$\begin{aligned}
&\tau(t) = \ell\tau^p(1) + \tau^p(t - \ell), \\
&\text{with } \ell = \text{nearest integer to } t \text{ with } \ell \leq t.
\end{aligned} \tag{15}$$

For our results we use the support points given in Table 1 for τ^p. The example is implemented with the forthcoming C++

x	$\tau(x)$	$\dot{\tau}(x)$
0.0	0.0	0.0
0.5	1.5	0.0
1.0	0.5	0.0

Table 1: Support points for τ^p

interface of the NLP-Solver WORHP, which is described in the next section.

4.2. The Multiobjective Interface for WORHP

In order to solve problem (4) we use the NLP-Solver WORHP ("We Optimize Really Huge Problems") [6] which is designed to solve continous high-dimensional nonlinear problems. It also provides the module WORHP Zen [7] which integrates an interface for parametric sensitivity analysis so that we are able to compute the sensitivity differentials (5). The implementation is done within the forthcoming new C++ interface of WORHP which aims to provide a unified interface for parallelization approaches [8], the transcription software TransWORHP [4] and algorithms for multiobjective optimization.

In Figure 1 the general structure of the new interface is shown. One can see that input for optimal control problems (OCP) and multiobjective optimal control problems

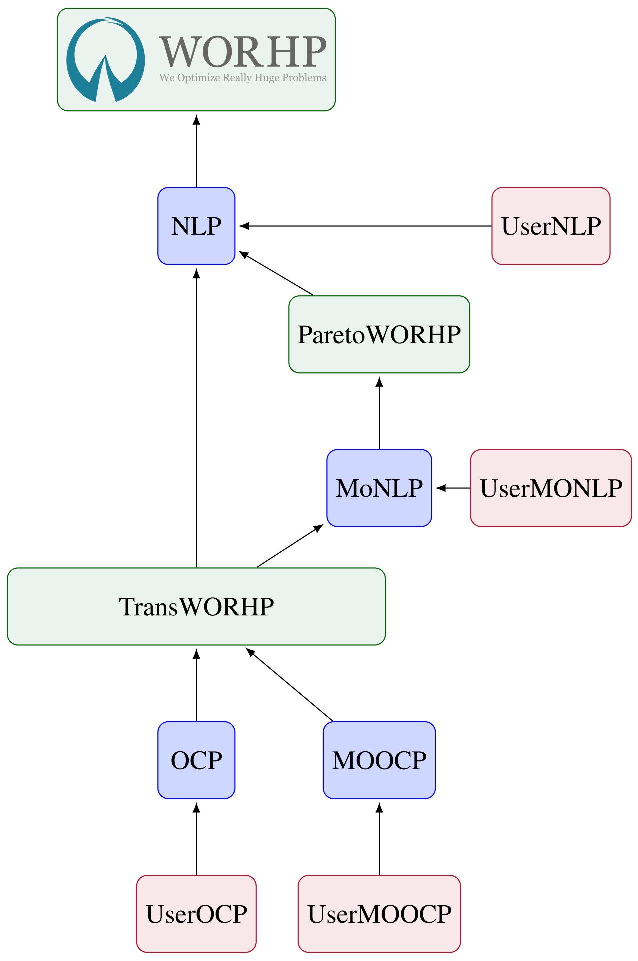

Fig. 1: Structure of the new interface for WORHP.
Blue: interface classes, red: possible user input, green: algorithms

(MOOCP) both go through the same transcription process, with a NLP or MONLP as the respective results. This makes it easy for the user to extend classical OCP to more than one objective. In addition, since every problem is in the end transformed into a NLP, all algorithms can benefit from advances in WORHP such as the above mentioned parallelization approaches.
The idea described in this paper will be fully integrated into the new interface and thus available to future users.

5. RESULTS

The violet line in Figure 2 shows the interpolation of $\hat{F}$ based on three samples which are shown as blue dots. One can see that it differs from the curve resulting from applying F to the approximation of $\hat{x}$ which is displayed by the orange line. Looking at the errors one might assume that the distance between the approximations and the approximation error do have a strong correlation and thus the assumption from Section 3.2 seems to be verifyed. For a closer look we display the errors in Figure 3.

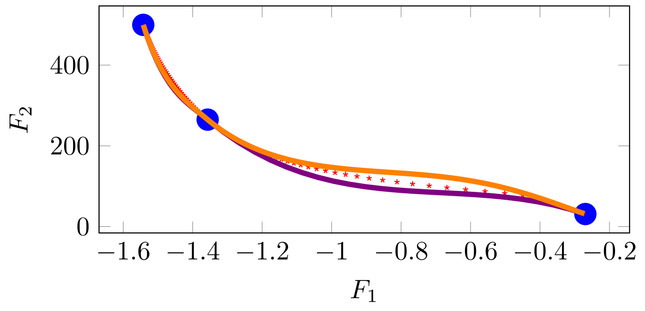

Fig. 2: Interpolated Pareto front. Blue: samples used for interpolation, violet: approximation of $\hat{F}$, orange: F applied to the approximation of $\hat{x}$, red: samples of the Pareto front with higher resultion

The interpolation errors are displayed in Figure 3. One can see that the maximum value of δ is in between the maxima of $\Delta^{\hat{F}}$ and Δ^{Fx} thus placing the next sample at the maximum of δ seems to be a compromise to reduce both approximation errors.

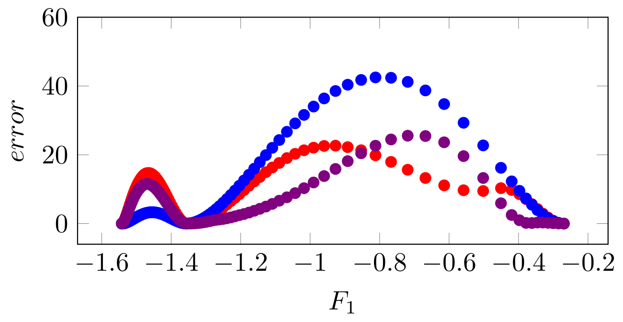

Fig. 3: Interpolation Error. Red: approximation error $\Delta^{\hat{F}}$, violet: approximation error Δ^{Fx}, blue: distance between both approximations δ.

After calculating the distance δ we aim to place the next sample at the point of the maximum value of δ. The result after a refined interpolation is shown in Figure 4 and Figure 5. One can see that the error is reduced by more than 50%.

One problem of the proposed idea can be seen in Figure 5 when looking at the left part of the Pareto front. Here both approximation have nearly the same error which results in a low distance δ but high error functions $\Delta^{\hat{F}}$ and Δ^{Fx}. If one now wants to know where the highest approximation error is over the whole Pareto front, the distance δ would be misleading. But if one only looks at the interval between both samples on the left side the distance is still a good indicator.

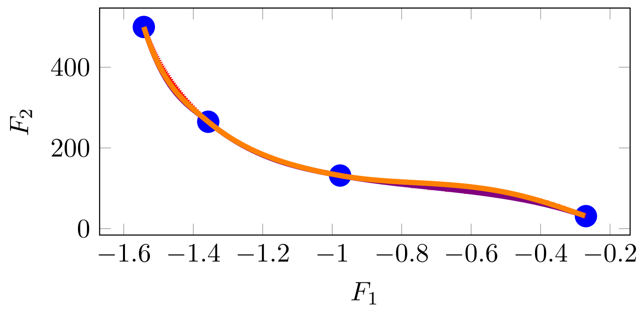

Fig. 4: Interpolated Pareto front with additional sample. Colors as in Figure 2.

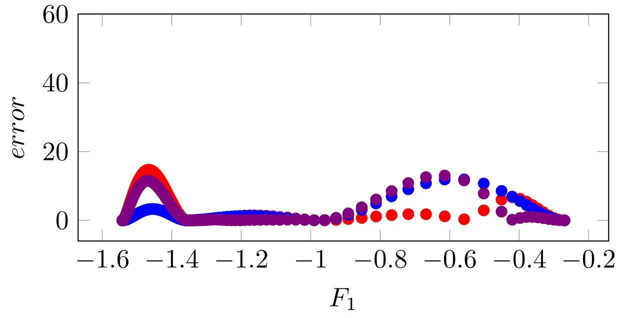

Fig. 5: Interpolation Error, after adding one additional sample. Colors as in Figure 3.

The result after adding four samples to the original three is displayed in Figure 6. The interval between -1.4 and -1.0 shows the advantage of the proposed method. Here the approximations are allready very good after adding one more sample (see Figure 5). As a result the distance between both is very low and thus no additional sample is placed here. Instead the errors on the right and left are reduced further to a small percentage of the original error which is shown in Figure 7.

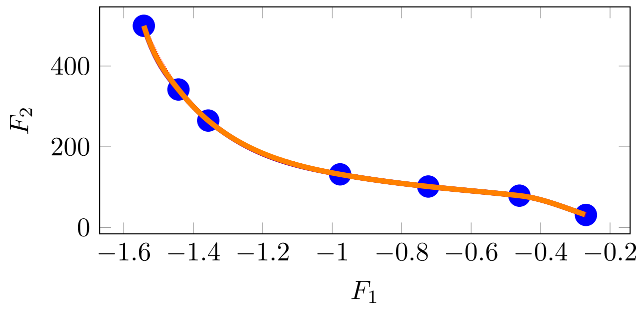

Fig. 6: Interpolated Pareto front after support point refinement. Colors as in Figure 2.

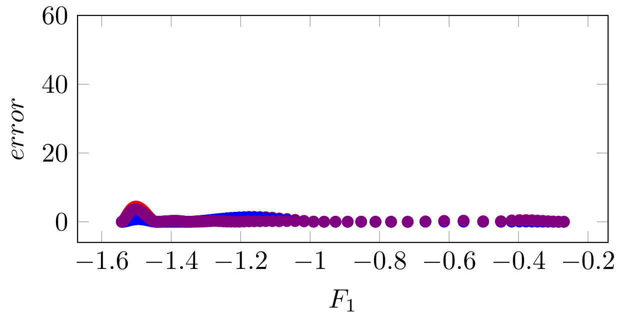

Fig. 7: Interpolation Error after support point refinement. Colors as in Figure 7.

6. CONCLUSION

We were able to exploit parametric sensitivity information to interpolate the Pareto front in objective and parameter space. Afterwards we used the distance between the interpolation in objective space and the curve resulting from applying the objective functions to the approximation in variable space. We showed that this distance could be used as an indicator where it would be usefull to compute another sample of the Pareto front. By applying this technique we reduced the approximation error significantly with only a few additional samples.
In the process we discovered the problem, that the distance between both approximations could be low even if the approximation errors are high. This leads to the necessity of further research regarding the approximation quality of the interpolations.

7. REFERENCES

[1] A. Pascoletti and P. Serafini, "Scalarizing vector optimization problems," *Journal of Optimization Theory and Applications*, vol. 42, no. 4, pp. 499–524, 1984.

[2] G. Eichfelder, *Adaptive scalarization methods in multiobjective optimization*, vol. 436, Springer, 2008.

[3] C. Büskens, "Echtzeitoptimierung und echtzeitoptimalsteuerung parametergestörter probleme," 2002.

[4] M. Knauer and C. Büskens, "From WORHP to TransWORHP," in *Proceedings of the 5th International Conference on Astrodynamics Tools and Techniques*, 2012.

[5] A. Berger, M. Knauer, and C. Büskens, "Pareto front interpolation based on parametric sensitivity anal- ysis in a bi-objective setting," *PAMM*, vol. 18, forthcoming 2018.

[6] C. Büskens and D. Wassel, "The esa nlp solver worhp," in *Modeling and optimization in space engineering*, pp. 85–110. Springer, 2012.

[7] R. Kuhlmann, S. Geffken, and C. Büskens, “Worhp zen: Parametric sensitivity analysis for the nonlinear programming solver worhp,” in *Operations Research Proceedings 2017*, pp. 649–654. Springer, 2018.

[8] S. Geffken and C. Büskens, “WORHP Multi-Core Interface, Parallelisation Approaches for an NLP Solver,” in *Proceedings of the 6th International Conference on Astrodynamics Tools and Techniques, 14.03. - 17.03.2016, Darmstadt, Germany*, 2016.