Technical University of Kaiserslautern
Fraunhofer Institute for Industrial Mathematics ITWM

Test rig optimization

by

Alexander Belyaev

Supervisor 1

apl. Prof. Dr. Karl-Heinz Küfer

Supervisor 2

Prof. Dr. Martin Arnold

Vom Fachbereich Mathematik
der Technischen Universität Kaiserslautern
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften
(Doctor rerum naturalium, Dr. rer. nat.)
genehmigte Dissertation

December 2014

D-386

Acknowledgments

It would not have been possible to write this doctoral thesis without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

First of all, I would like to thank PD Dr. habil. Karl-Heinz Küfer for providing the opportunity of writing my PhD thesis at the Fraunhofer Institut for Industrial Mathematics in the department of Optimization.

I also thank Volker Maag, who has assisted me in numerous ways. I specially thank him for his infinite patience and help with C++ code. I will like to thank Richard Welke for the idea to use Asymptote for visualization. I thank Dimitri Nowak, Vladimir Shiryaev, Katrin Stöbener, Martin Obermayr and Michael Speckert for their corrections and hints.

Special thanks to Ivan Serna, Tigran Nagapetyan, Edward Toroshchin, Vladimir Shiryaev and Dmitry Chistikov – thank you for showing me how skilled and whip-smart should an applied mathematician be. I also thank my friends Yaroslav Melnyk, Naveen Gopinath, Dimitri Nowak, Tjorben Groß, Aleksandra Kovtanyuk and Vyacheslav Goncharenko.

Furthermore, I would also like to acknowledge the Optimization Department in the Fraunhofer Institut for Industrial Mathematics for their financial and technological support.

Lastly, I would like to thank my mother and granny for all their love and encouragement. I am deeply grateful to them for all that I am. To Aleksandra for her love and patience.

Abstract

Designing good test rigs for fatigue life tests is a common task in the automotive industry. The problem to find an optimal test rig configuration and actuator load signals can be formulated as a mathematical program. We introduce a new optimization model that includes multi-criteria, discrete and continuous aspects. At the same time we manage to avoid the necessity to deal with the rainflow-counting (RFC) method. RFC is an algorithm, which extracts load cycles from an irregular time signal. As a mathematical function it is non-convex and non-differentiable and, hence, makes optimization of the test rig intractable.

The block structure of the load signals is assumed from the beginning. It highly reduces complexity of the problem without decreasing the feasible set. Also, we optimize with respect to the actuators' positions, which makes it possible to take torques into account and thus extend the feasible set. As a result, the new model gives significantly better results, compared with the other approaches in the test rig optimization.

Under certain conditions, the non-convex test rig problem is a union of convex problems on cones. Numerical methods for optimization usually need constraints and a starting point. We describe an algorithm that detects each cone and its interior point in a polynomial time.

The test rig problem belongs to the class of bilevel programs. For every instance of the state vector, the sum of functions has to be maximized. We propose a new branch and bound technique that uses local maxima of every summand.

Zusammenfassung

Das Entwerfen guter Prüfstände für Ermüdungstests ist eine häufige Aufgabe in der Automobilindustrie. Das Problem, optimale Prüfstandskonfigurationen und Lastsignale für Aktoren zu finden, kann als mathematisches Optimierungsproblem formuliert werden. Wir führen ein neues Optimierungsmodel ein, das multikriterielle, diskrete und kontinuierliche Aspekte hat. Dabei vermeiden wir die Notwendigkeit, die Rainflow-counting-Methode (RFC) zu verwenden. RFC ist ein Algorithmus, der Lastzyklen aus irregulären Zeitreihen extrahiert. Betrachtet man ihn als eine mathematische Funktion, so ist diese weder konvex noch differenzierbar und macht daher die Prüfstandsoptimierung ist daher schwer zu bewältigen.

Wir nehmen durchweg an, dass die Lastsignale eine Blockstruktur haben. Dies reduziert die Komplexität des Problems stark ohne die zulässige Menge zu verkleinern. Außerdem optimieren wir bezüglich der Aktorenposition, was erlaubt. Drehmomente mit einzubeziehen und die zulässige Menge zu erweitern. Als Folge davon erzielt das neue Model wesentlich bessere Ergebnisse, verglichen mit anderen Ansätzen zur Prüfstandsoptimierung.

Unter bestimmten Umständen ist das nicht-konvexe Prüfstandsproblem eine Vereinigung von konvexen Problemen auf Kegeln. Numerische Optimierungsmethoden brauchen in der Regel Randbedingungen und einen Startpunkt. Wir beschreiben einen Algorithmus, der in polynomialer Zeit die einzelnen Kegel erkennt und jeweils einen inneren Punkt generiert.

Das Prüfstandsproblem gehört zu der Klasse von Bilevel-Optimierungsproblemen. Für jede Instanz eines Zustandsvektors muss die Summe von Funktionen maximiert werden. Wir schlagen eine neue branch-and-bound-Technik vor, die die lokalen Maxima jedes Summanden nutzt.

Contents

Chapter 1

Introduction

Education: the path from cocky ignorance to miserable uncertainty.
Mark Twain

Durability is one of the most important physical properties of vehicle components. The natural procedure to assess their structural damage is the test drive, but it is expensive, takes a lot of time and can be done only after the whole automobile is assembled. One of the possible ways to make the tests cheaper and faster is to use the test rigs, in which actuators create load signals to emulate the real test drive damage.

The goal of this work is to formulate and treat numerically an optimization problem that allows to design test rigs capable of approximating fatigue damage at several chosen hot spots, assuming that the loading signal is cyclic and consists of several blocks of constant amplitude and mean. We are going to study several models under different assumptions, varying in optimization parameters. The most general problem considers attachment point for fixation to the test rig, number and attachment points of the actuators, actuators alignment parameters, numbers of cycles and amplifying parameters of each actuator in every block.

Thesis structure

The chapter 2 starts with a brief description of fatigue damage calculation and criteria that can be used to assess the quality of the test. We consider a component that has several attachments for either an actuator or a fixation to the test rig. Hot spots, where we want to assess reference damage values, are predefined and given. Fig. 1.1 shows an example of such a component. The mathematical program that allows to find an optimal design of the test rig will be derived and verified.

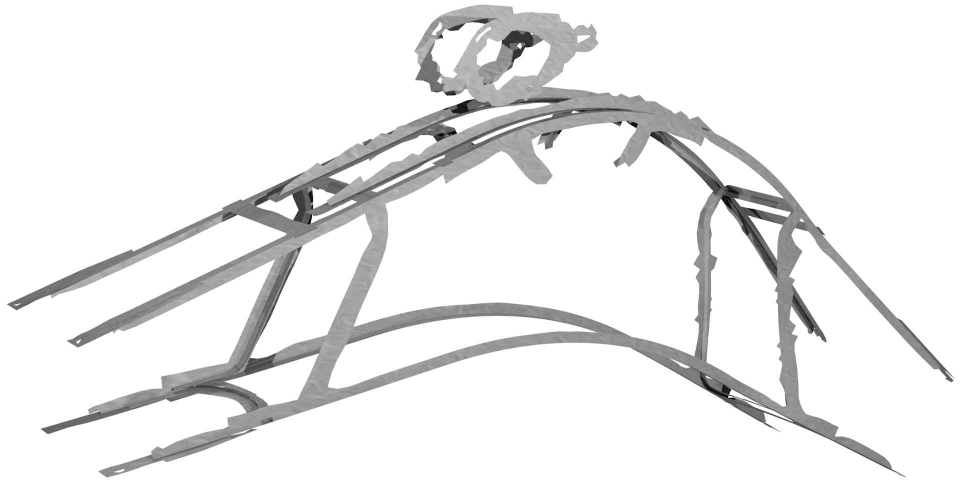


Figure 1.1: Geometry of the component

The chapter 3 contains the analysis of the single level formulation of the test rig model. Convexity of the damage function will be proven. Also it will be shown that under certain assumptions the test rig problem is a union of convex optimization problems on polyhedral cones. In order to detect such cones, we will present an efficient algorithm and prove its worst case polynomial time complexity.

In the chapter 4 the most generic formulation of the test rig problem will be considered. It belongs to the class of bilevel programs, since every instance of the state vector, which includes alignment parameters and applied forces, defines a one-dimensional inner maximization problem at each hot spot. We will propose a new branch-and-bound technique, which exploits the knowledge of the extrema of each summand to find the global maximum of the sum effectively. Convergence of the algorithm will be proven and its running time will be tested and compared with the running time of DIRECT algorithm.

Next, the regularity of the outer program will be studied. It is not convex in general, but we can convert the problem into a DC program. Also differentiability of the problem will be shown and the gradient with respect to all state variables, including alignment parameters, will be found. The possibility to solve the test rig problem by derivative-based global optimization methods will be demonstrated.

In the chapter 5 the bilevel formulation of the problem will be solved numerically for two components in order to address several important questions about the test rig design.

Remark 1.0.1. All computations in the current work were done on a Fedora 20 desktop with Intel Core i5-520M dual-core CPU and 4GB of RAM.

Chapter 2

Modeling

Let us start with a brief description of fatigue damage calculation and criteria that can be used to assess the quality of the test. We consider a component that has several attachments for either an actuator or a fixation to the test rig. Hot spots, where we want to assess reference damage values, are predefined and given. Finally, the mathematical program that allows to find an optimal design of the test rig will be derived and studied.

2.1 Damage Computation

The behavior of materials under repeated stresses or strains is usually referred to as fatigue. The formal definition of the process is currently provided by the American Society for Testing and Materials [2].

Definition 2.1.1. Fatigue is the process of progressive localized permanent structural change occurring in a material subjected to conditions that produce fluctuating stresses and strains at some point or points and that may culminate in cracks or complete fracture after a sufficient number of fluctuations.

Fatigue damage calculation is a quite challenging task that requires a number of assumptions, simplifications, approximations as well as empirical data. Fortunately, a hierarchy of models for the damage computation exists. Here we provide a brief overview of basic fatigue computation to describe necessary tools for later damage approximation and optimization.

2.1.1 Stress vector

Mechanical stress can be thought of as a measure of the intensity level of internal forces in a material. The fatigue damage models consider surface stresses, which are completely described by the two normal components σ_{xx}, σ_{yy} and a shear stress σ_{xy} in a two-dimensional setting. Fig. 2.1 shows the directions of all components.

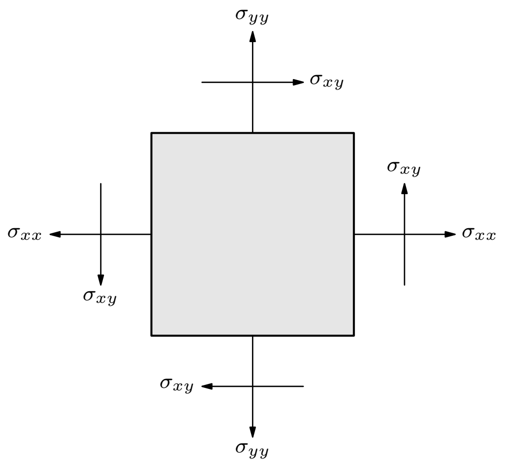


Figure 2.1: Cross section under stress in $\mathbb{R}^2$

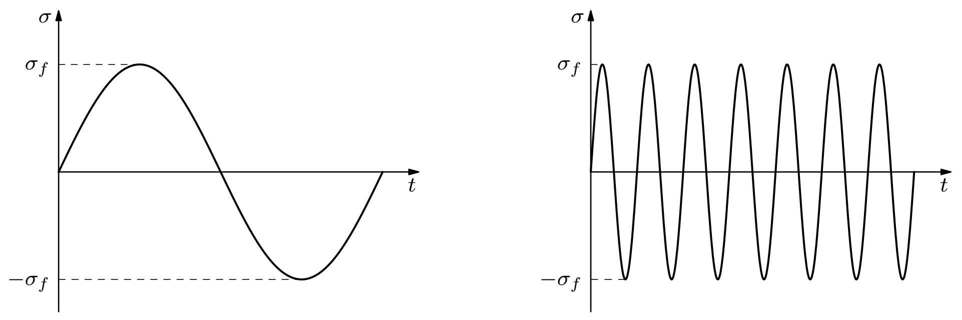
The normal stress components σ_{xx} and σ_{yy} are directed orthogonally to the cross section and, hence, correspond to the compression or the tension of the material. The shear stress σ_{xy}, on the other hand, acts along the cross section. We combine all components in a stress vector $\boldsymbol{\sigma} = [\sigma_{xx}, \sigma_{yy}, \sigma_{xy}]^T$.

2.1.2 Uniaxial constant fully reversed loading

The simplest damage model considers a uniaxial constant fully reversed signal, when a single force is acting on the surface element and oscillates about zero. Later it can be extended to cope with more general stress signals.

Definition 2.1.2. A stress vector $\boldsymbol{\sigma} = [\sigma_{xx}, \sigma_{yy}, \sigma_{xy}]^T$ is called uniaxial, if it can be represented as $\boldsymbol{\sigma} = \sigma \boldsymbol{e}_i$ for $i = 1, \ldots, 3$, or multiaxial otherwise.

A uniaxial stress loading history is a scalar signal that can be random, steady or periodic. Fig. 2.2 shows an example of a periodic loading with a constant amplitude and zero mean.



(a) Single cycle (b) Constant loading

Figure 2.2: Uniaxial fully reversed loading

Definition 2.1.3. Fully reversed cycle (Fig. 2.2a) is a stress-strain cycle that has a zero mean value. The amplitude σ_f of such cycle is a fully reversed stress.

Fatigue life of the component under a periodic loading can be described quantitavely in terms of the number of cycles at different amplitudes.

Definition 2.1.4. Fatigue life $N(\sigma_f)$ is a number of fully reversed cycles with an amplitude σ_f a component can survive without failure, i.e. fracture or crack.

The S-N curve [40], also known as a Wöhler curve, maps the fully reversed stress σ_f to the fatigue life N. The curve can be derived, applying the constant uniaxial fully reversed loading until the component fails.

Although the S-N curve is a very powerful tool, it is not convenient to use the test data directly in practice. Basquin [4] approximated the S-N curve with a power function $N(\sigma_f) = \sigma_f^{-k}$ with a positive k. The two-slope Basquin model's distinguishes between low and high cycle fatigue [40], hence achieves a better approximation:

$$N(\sigma_f) = N_s \begin{cases} (\sigma_f/\sigma_s)^{-k_1} & \text{for} \quad \sigma_f \leq \sigma_s, \\ (\sigma_f/\sigma_s)^{-k_2} & \text{for} \quad \sigma_f \geq \sigma_s, \end{cases} \tag{2.1}$$

where parameters N_s, σ_s, k_1, k_2 depend highly on the component geometry and its material.

Remark 2.1.5. One should remember that the frequency and mean of the stress-strain signal as well as temperature, component geometry, material properties, surface finish and corrosion affect durability of the component, see [40], and influence parameters of Basquin's model, defined by Eq. (2.1).

The number of cycles N needed to fail is given by the S-N curve for each amplitude σ_f. Miner [29] suggested that a constant loading signal with n cycles contributes $d = n/N$ portion of total damage. If d is greater than 1, then the component fails. We introduce a quantity $g(\sigma_f)$, which denotes a portion of damage induced by a single fully reversed cycle. The following definition uses the Basquin's model of S-N curve.

Definition 2.1.6. Damage curve $g(\sigma_f) : \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+$ is a mapping between stress amplitude σ_f of a single stress-strain cycle and a fracture of damage it produces:

$$g(\sigma_f) := \frac{1}{N(\sigma_f)} = \frac{1}{N_s} \begin{cases} (\sigma_f/\sigma_s)^{k_1} & \text{for} \quad \sigma_f \leq \sigma_s, \\ (\sigma_f/\sigma_s)^{k_2} & \text{for} \quad \sigma_f \geq \sigma_s, \end{cases} \tag{2.2}$$

where $k_1, k_2 > 2$.

Fig. 2.3 shows an example of a one-slope damage curve g, when $k_1 = k_2$, and a two-slope curve, when $k_1 \neq k_2$. Both curves are continuous and strictly monotonically increasing, although the two-slope curve is not differentiable at σ_s.

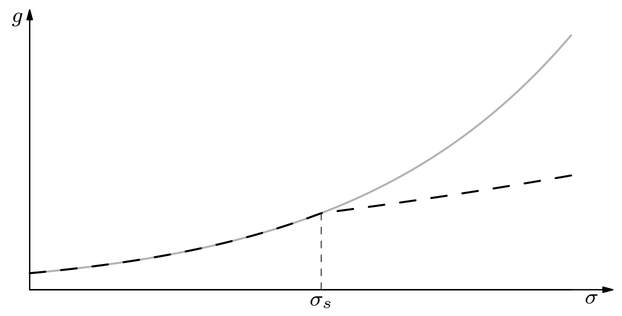


Figure 2.3: One-slope (gray) and two-slope (dashed) damage curves

Corollary 2.1.7. *Damage d under uniaxial constant fully reversed loading can be computed using the following formula:*

$$d(n, \sigma_f) = ng(\sigma_f), \tag{2.3}$$

where n is the number of cycles in the stress signal and σ_f is the signal amplitude.

Equation (2.3) models only a simple case, but it can be used to describe much more complicated situations that we will consider in the following sections.

2.1.3 Uniaxial constant general loading

Mean stress magnitude σ_m of a constant loading signal as in Fig. 2.4 affects fatigue life of a component. When σ_m is positive, then the stress is compressive and it increases the damage produced by the signal. If σ_m is negative, then the stress is tensile and, in general, decreases the damage.

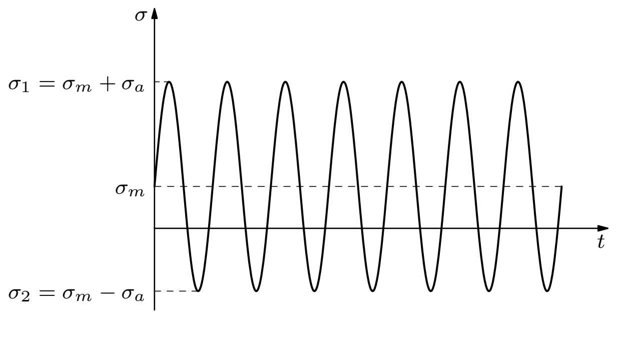


Figure 2.4: Stress-strain cycle with nonzero mean

There are several models that correct mean stress effects by mapping stress amplitude σ_a and mean σ_m to the corresponding fully reversed stress $\sigma_f(\sigma_a, \sigma_m)$, discussed in [40]. Gerber's equation approximates experimental data with parabola, but appears to predict compressive mean stress damage incorrectly. Goodman's relation and its modifications provide much better results, although they are based on linear interpolation:

$$\frac{\sigma_a}{\sigma_f} + \frac{\sigma_m}{\sigma_u} = 1, \tag{2.4}$$

where σ_u is the ultimate tensile strength. Fuchs [12] developed a more general and precise model to correct mean stress effects, see [40] for more details.

Remark 2.1.8. In this work we define the mean stress correction, depending on the turning points (local minima and maxima) of the stress-strain cycle σ_1 and σ_2 rather then on the amplitude $\sigma_a = |\sigma_1 - \sigma_2|/2$ and the mean $\sigma_m = (\sigma_1 + \sigma_2)/2$ to simplify the notation.

Definition 2.1.9. Mean stress correction is a function $h(\sigma_1, \sigma_2) : \mathbb{R}^2 \mapsto \mathbb{R}_0^+$ that maps the turning points of the stress cycle σ_1 and σ_2 to the corresponding fully reversed stress amplitude σ_f that produces the same damage.

Remark 2.1.10. A fully reversed stress-strain cycle always has turning points $\sigma_1 = -\sigma_2$ and hence $h(\sigma_1, -\sigma_1) = |\sigma_1|$. Also h is symmetric, i.e. $h(\sigma_1, \sigma_2) = h(\sigma_2, \sigma_1)$.

Example 2.1.11. One of the variants of the Goodman correction (2.4) is given by the following relation:

$$h(\sigma_1, \sigma_2) = \begin{cases} \dfrac{\sigma_a\, \sigma_u}{\sigma_u + \sigma_a} & \text{for} \quad \sigma_m \in (-\sigma_u, -\sigma_a] \\[2ex] \dfrac{\sigma_a\, \sigma_u}{\sigma_u - \sigma_m} & \text{for} \quad \sigma_m \in [-\sigma_a, \sigma_u) \end{cases} \tag{2.5}$$

where the amplitude $\sigma_a = |\sigma_1 - \sigma_2|/2$ and the mean $\sigma_m = (\sigma_1 + \sigma_2)/2$.

Isolines of the modified Goodman stress correction σ_f in Fig. 2.5 are piecewise linear functions that change their slope after the pulsating tension condition is achieved, i.e. values of the stress-strain cycle are non-positive and hence $\sigma_m = -\sigma_a$. Tensile mean stresses do not have any influence for $\sigma_m < -\sigma_a$ in this particular model. On the other hand mean stress influence is beneficial for $\sigma_m > -\sigma_a$.

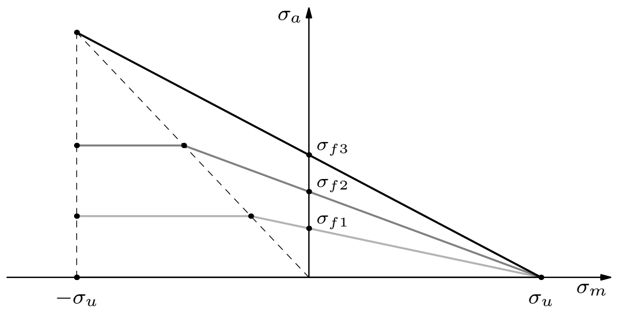


Figure 2.5: Isolines of the modified Goodman equation, corresponding to the values σ_{f1}, σ_{f2} and σ_{f3}

Given a method to calculate fully reversed stress for a cycle with arbitrary mean and amplitude it becomes possible to apply Equation 2.3.

Corollary 2.1.12. *Damage d in case of constant uniaxial loading signal with the turning points σ_1 and σ_2 can be obtained by the following relation:*

$$d = ng(h(\sigma_1, \sigma_2)), \tag{2.6}$$

where g is a damage curve, h is a mean stress correction and n is a number of cycles in the signal.

We have briefly discussed how tensile or compressive mean stresses affect produced damage and how to calculate damage when we have three degrees of freedom, namely amplitude and mean of the loading signal and number of cycles in it.

2.1.4 Uniaxial block general loading

Damage computation becomes much trickier in case of variable loading. Let us consider a block stress signal that consists of several blocks with constant amplitude and mean values. Fig. 2.6 shows an example of a signal with 3 blocks.

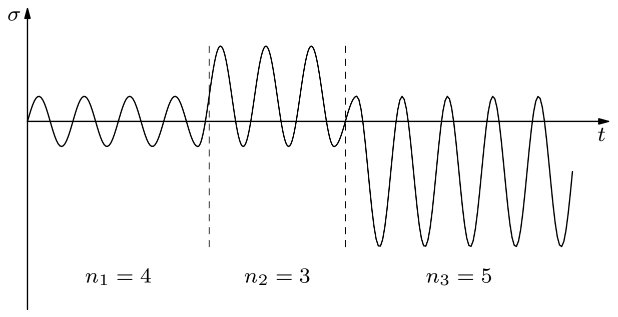


Figure 2.6: Block loading

Palmgren [34] and later Miner [29] suggested that the damage produced by such a block signal is equal to the sum of damages, produced by each block:

$$d = \sum_{b=1}^{B} d_b, \tag{2.7}$$

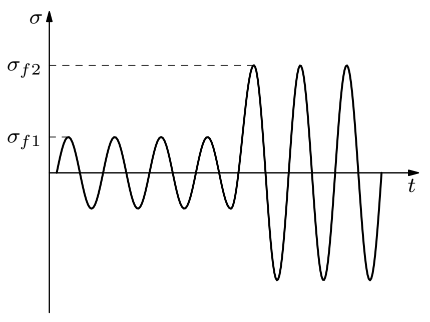
where B is the number of blocks, d_b is the damage by block $b = 1, \dots, B$. The Palmgren-Miner damage accumulation rule is linear and predicts failure, when $d \geq 1$.

Corollary 2.1.13. *Damage d in case of uniaxial block loading signal with B blocks can be computed using the Palmgren-Miner accumulation rule (2.7):*

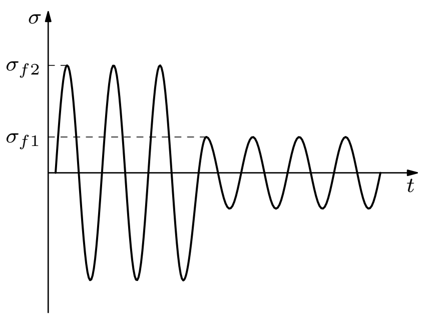
$$d = \sum_{b=1}^{B} n_b g(h(\sigma_{1,b}, \sigma_{2,b})), \tag{2.8}$$

where g is a damage curve, h is mean stress correction, vectors n_b, $\sigma_{1,b}$ and $\sigma_{2,b}$ contain numbers of cycles and the turning points for the stress-strain cycles in block $b = 1, \dots, B$.

Remark 2.1.14. Palmgren-Miner accumulation rule does not take the order of the generated stresses into account, so that the overall damage in Fig 2.7a and Fig. 2.7b are equal, which is not true in general.



(a)



(b)

Figure 2.7: Order effect

There are nonlinear fatigue models [40] that include stress order. However, in the case of random loading they predict fatigue damage with the same accuracy as the Palmgren-Miner rule [32, 40].

2.1.5 Multiaxial block general loading

The aim of the multiaxial loading analysis is to convert the stress vector $\boldsymbol{\sigma}$ into a corresponding scalar uniaxial stress σ and then calculate fatigue as above. The methods to assess the stress can be empirical [13, 15] or using the stress invariants, mesoscopic properties, energy based criteria, etc [42].

In this work we concentrate on the critical plane approach. It is widely used in industry and reflects the physical nature of the fatigue damage [40], since cracks and fractures nucleate on specific planes. The idea is to consider a plane, on which the failure is the quickest.

Definition 2.1.15. A scalar stress $\sigma(\alpha)$ at a given material point is the normal stress component of the multiaxial stress $\boldsymbol{\sigma} = [\sigma_{xx}, \sigma_{yy}, \sigma_{xy}]^T$ acting on the critical plane $\boldsymbol{c}(\alpha)$:

$$\sigma(\alpha) = \boldsymbol{c}(\alpha)^T \boldsymbol{\sigma}, \tag{2.9}$$

where the critical plane $\boldsymbol{c}(\alpha)$, passing through the material point, is a plane, defined by Mohr's equation [35]:

$$\boldsymbol{c}(\alpha) = [1 + \cos\alpha, 1 - \cos\alpha, 2\sin\alpha]/2 \tag{2.10}$$

Remark 2.1.16. The "classical" stress tensor is a symmetrical matrix with only six components, namely, σ_{xx}, σ_{yy}, σ_{zz}, σ_{xy}, σ_{xz}, σ_{yz} in the global coordinate system:

$$\Sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{xy} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{xz} & \sigma_{yz} & \sigma_{zz} \end{bmatrix}$$

Since we are interested only in surface stresses, the axes are rotated so that the new axis z is normal to the surface element. In this direction the components $\sigma_{zz}, \sigma_{xz}, \sigma_{yz}$ are negligible, so we only get three tensor components $\boldsymbol{\sigma} = (\sigma_{xx}, \sigma_{yy}, \sigma_{xy})^T$ or in the matrix form:

$$\Sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix} \tag{2.11}$$

The coordinate system xy of the surface element can be rotated again by an angle γ clockwise using the rotation matrix $\mathcal{R}(\gamma)$:

$$\mathcal{R}(\gamma) = \begin{bmatrix} \cos\gamma & \sin\gamma \\ -\sin\gamma & \cos\gamma \end{bmatrix}$$

The normal stress component of the transformed stress matrix in Eq. (2.11) is the first element of the resulting stress:

$$\begin{aligned} \sigma &= (\mathcal{R}(\gamma)\Sigma\mathcal{R}^T(\gamma))_{1,1} \qquad (2.12) \\ &= \frac{\sigma_{xx}}{2}(1 + \cos 2\gamma) + \frac{\sigma_{yy}}{2}(1 - \cos 2\gamma) + \sigma_{xy}\sin 2\gamma \end{aligned}$$

If we substite $\alpha = 2\gamma$ and write expression (2.12) as a scalar product, we obtain Mohr's equation (2.9)–(2.10).

We are looking for the plane, in which the fracture happens first. Hence, it maximizes the cumulative damage:

$$d = \max_{\alpha} \sum_{b=1}^{B} d(\sigma(\alpha)) \tag{2.13}$$

Combining the Eq. with the model for the uniaxial block general loading (2.8), we get the following result.

Corollary 2.1.17. *Damage d in case of multiaxial block loading signal with B blocks can be computed using the critical plane approach (2.9):*

$$d = \max_{\alpha} \sum_{b=0}^{B} n_b g(h(\boldsymbol{c}(\alpha)^T \boldsymbol{\sigma}_{1,b}, \boldsymbol{c}(\alpha)^T \boldsymbol{\sigma}_{2,b})), \tag{2.14}$$

where g is a damage curve, h is mean stress correction, vectors n_b, $\boldsymbol{\sigma}_{1,b}$ and $\boldsymbol{\sigma}_{2,b}$ contain numbers of cycles and the turning points for the stress-strain cycles in block $b = 1, \ldots, B$.

2.1.6 Multiaxial irregular loading

The arbitrary loading signal can be converted into the block loading, using the rainflow counting method, proposed by Endo et al. [10, 11]. The rainflow counting is an iterative algorithm that extracts the stress-strain cycles from the series of succesive turning points. The extensive description of the algorithm and its modifications can be found in [19].

Remark 2.1.18. Real service loads are irregular and, hence, the rainflow counting method is indispesible to damage computation. However, test rig can generate block loading using the hydraulic actuators, which neglects the necessity to use the rainflow counting algorithm in the optimization of the applied loading, reduces the number of parameters of the signal to the number of cycles and the turning points for the stress-strain cycles in every block only, instead of considering enormous time-series of the turning points. Thus, optimization with the block signal is more structured, easier to implement in the test rig and has nicer mathematical properties.

2.1.7 General damage computation

We provided all necessary tools for the damage computation in the most general case, namely, multiaxial irregular loading. At first the vector stress time-series $\boldsymbol{\sigma}_t$ (Fig. 2.8a) is converted into the scalar time-series $\sigma_t(\alpha)$ (Fig. 2.8b) using the critical plane $\boldsymbol{c}(\alpha)$. After applying the rainflow counting to extract stress-strain cycles (Fig. 2.8c) with the turning points $\sigma_{1,b}$ and $\sigma_{2,b}$ for $b = 1, \ldots, B$, it becomes possible to calculate the equivalent fully reversed cycles amplitudes $\sigma_{f,b} = h(\sigma_{1,b}, \sigma_{2,b})$ with the mean stress correction function (Fig. 2.8d). The amplitude of every cycle is mapped to its damage value $d_b = g(\sigma_{f,b})$ by the damage function. Finally, according to the Palmgren-Miner rule, the cumulative damage of the loading signal is the sum of each stress-strain cycle contributions, i.e. $d = d_1 + \ldots + d_B$.

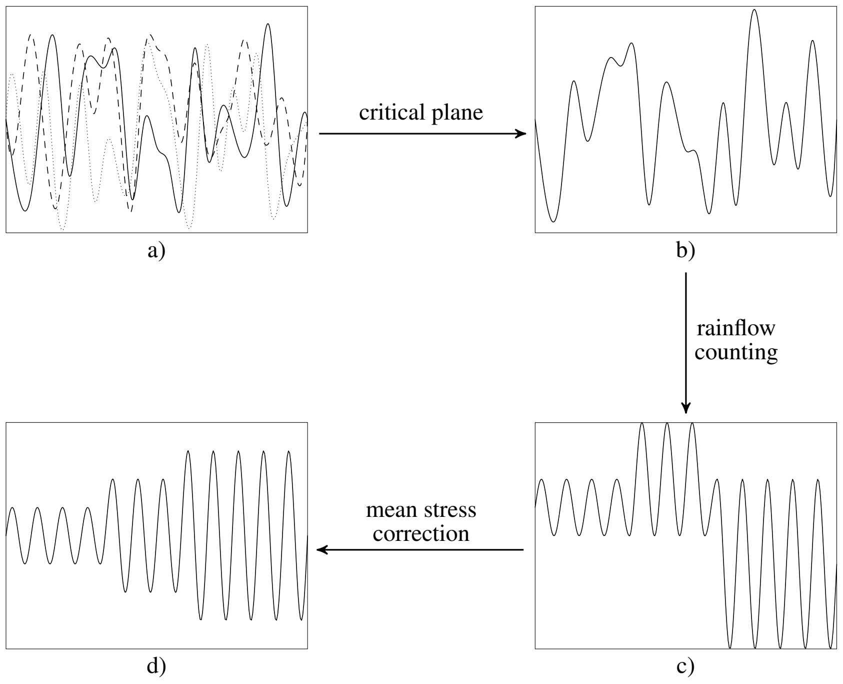


Figure 2.8: Damage calculation workflow

Remark 2.1.19. Damage calculation is usually performed for several equally spaced critical plane angles $\alpha_1, \ldots, \alpha_W$, where the number of planes W is often [27] 12, 18 or 36.

We discussed how to get damage values from the stress signal. Next section is devoted to the stress computation from the parameters of the test rig. These parameters will be used as optimization variables in the mathematical program.

2.2 Stress Computation

Eq. (2.14) allows to calculate damage in case of multiaxial block loading given turning points of stress-strain cycles. Computation of the stress vectors depends on many parameters of the test rig including the placement and alignment of actuators and the choice of fixation and loading signals applied in every block. In this section, we formulate these notions and provide a damage model that suits for optimization.

2.2.1 Equilibrium equation

We first define load, attachment and mount point. After that we will derive a load equilibrium equation for a chosen configuration of the test rig.

Definition 2.2.1. An attachment is a place, where the component can be either fixed or an actuator can be installed. Let $A+1$ be the number of attachments. Each attachment is characterized by its spatial coordinates $\boldsymbol{a}_i$ for $i = 0, \dots, A$.

Definition 2.2.2. A load $\boldsymbol{l}_i$ is a vector, which aggregates force $\boldsymbol{f}_i$ and moment $\boldsymbol{m}_i$, acting in the attachment i for $i = 0, \dots, A$:

$$\boldsymbol{l}_i = [\boldsymbol{f}_i; \boldsymbol{m}_i] = [f_i^x, f_i^y, f_i^z, m_i^x, m_i^y, m_i^z]^T$$

Lemma 2.2.3. *Let the component with A attachments be fixed at the mount point 0. Assume that every other attachment $i \in \{1, \dots, A\}$ has an actuator installed, which generates a load $\boldsymbol{l}_i$. Then the reaction load $\boldsymbol{l}_0$ at the mount point can be written as:*

$$\boldsymbol{l}_0 = -\sum_{i=1}^{A} \underbrace{\begin{bmatrix} \mathcal{I}_3 & 0 \\ \mathcal{R}_i & \mathcal{I}_3 \end{bmatrix}}_{=:\mathcal{D}_i} \boldsymbol{l}_i \tag{2.15}$$

where the matrix $\mathcal{R}_i \in \mathbb{R}^{3\times 3}$ is defined by

$$\mathcal{R}_i := \begin{bmatrix} 0 & -r_i^3 & r_i^2 \\ r_i^3 & 0 & -r_i^1 \\ -r_i^2 & r_i^1 & 0 \end{bmatrix} \tag{2.16}$$

and the lever $\boldsymbol{r}_i := \boldsymbol{a}_i - \boldsymbol{a}_0$.

Proof. The reaction force f_0 at the mount point 0 balances the forces f_i for $i = 1, \dots, A$, generated by the actuators.

$$\boldsymbol{f}_0 + \sum_{i=1}^{A} \boldsymbol{f}_i = 0 \tag{2.17}$$

The same is valid for the moments equilibrium, taking into account not only the moments of the actuators, applied directly at the attachments, but also the moments, which appear due to the forces $\mathbf{f}_i$ with the corresponding levers $\mathbf{r}_i$.

$$\boldsymbol{m}_0 + \sum_{i=1}^{A} \boldsymbol{m}_i + \sum_{i=1}^{A} \boldsymbol{r}_i \times \boldsymbol{f}_i = 0, \tag{2.18}$$

where $\boldsymbol{r}_i$ is a vector from $\boldsymbol{a}_0$ to $\boldsymbol{a}_i$.

Let us expand the expression for the moments $\mathbf{r}_i \times \mathbf{f}_i$:

$$\begin{aligned}
\mathbf{r}_i \times \mathbf{f}_i &= f_i^x \, \mathbf{r}_i \times \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + f_i^y \, \mathbf{r}_i \times \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + f_i^z \, \mathbf{r}_i \times \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\
&= f_i^x \begin{bmatrix} 0 \\ r_i^3 \\ -r_i^2 \end{bmatrix} + f_i^y \begin{bmatrix} -r_i^3 \\ 0 \\ r_i^1 \end{bmatrix} + f_i^z \begin{bmatrix} r_i^2 \\ -r_i^1 \\ 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & -r_i^3 & r_i^2 \\ r_i^3 & 0 & -r_i^1 \\ -r_i^2 & r_i^1 & 0 \end{bmatrix} \mathbf{f}_i =: \mathcal{R}_i \, \mathbf{f}_i \qquad (2.19)
\end{aligned}$$

Eq. (2.19) is used to formulate both forces (2.17) and moments (2.18) balance equations in terms of loads $\boldsymbol{l}_i$:

$$\begin{aligned}
\mathbf{l}_0 = \begin{bmatrix} \mathbf{f}_0 \\ \mathbf{m}_0 \end{bmatrix} &= -\sum_{i=1}^{A} \begin{bmatrix} \mathbf{f}_i \\ \mathbf{m}_i + \mathbf{r}_i \times \mathbf{f}_i \end{bmatrix} \\
&= -\sum_{i=1}^{A} \begin{bmatrix} \mathbf{f}_i \\ \mathbf{m}_i + \mathcal{R}_i \, \mathbf{f}_i \end{bmatrix} = -\sum_{i=1}^{A} \begin{bmatrix} \mathcal{I}_3 & 0 \\ \mathcal{R}_i & \mathcal{I}_3 \end{bmatrix} \mathbf{l}_i,
\end{aligned}$$

which proves the formula (2.15) for the reaction load $\boldsymbol{l}_i$. □

2.2.2 Unit load stresses

Here we describe stress calculation for a chosen configuration given unit load stresses.

Definition 2.2.4. A unit load stress $\boldsymbol{\sigma}_{ij}(\boldsymbol{x}) : \mathbb{R}^3 \mapsto \mathbb{R}^3$ at the spot $\boldsymbol{x}$ is a stress vector generated by the unit load $\boldsymbol{e}_j$ for $j = 1, \dots, 6$, acting in the attachment $i \in \{0, \dots, A\}$.

Definition 2.2.5. A unit load stress matrix $\mathcal{S}_i(\boldsymbol{x}) := [\boldsymbol{\sigma}_{i1}(\boldsymbol{x}), \dots, \boldsymbol{\sigma}_{i6}(\boldsymbol{x})] : \mathbb{R}^3 \mapsto \mathbb{R}^{3\times 6}$ encapsulates all unit load stresses in the spot $\boldsymbol{x}$, generated by the unit forces acting in the attachment $i \in \{0, \dots, A\}$.

The unit load stress matrices $\mathcal{S}_i(\boldsymbol{x}_k)$ for every attachment $i \in \{0, \dots, A\}$ at each hot spot $\boldsymbol{x}_k$ are given as the result of a finite element simulation performed on an unconstrained component using the inertia relief method [19].

Theorem 2.2.6. *The stress vector $\boldsymbol{\sigma}(\boldsymbol{x})$ under assumptions of Lem. 2.2.3 has the following form:*

$$\boldsymbol{\sigma}(\boldsymbol{x}) = \sum_{i=1}^{A} \big(\mathcal{S}_i(\boldsymbol{x}) - \mathcal{S}_0(\boldsymbol{x})\mathcal{D}_i\big)\, \boldsymbol{l}_i, \qquad (2.20)$$

where $\mathcal{S}_i(\boldsymbol{x})$ are unit load stress matrices for corresponding attachments and $\mathcal{D}_i$ are defined as in Eqs. (2.15), (2.16).

Proof. The stress can be written as a linear combination of unit load stresses:

$$\begin{aligned}\boldsymbol{\sigma}(\boldsymbol{x}) &= \sum_{i=0}^{A}\sum_{j=1}^{6} l_{i,j}\,\boldsymbol{\sigma}_{ij}(\boldsymbol{x}) = \sum_{i=0}^{A}\mathcal{S}_i(\boldsymbol{x})\,\boldsymbol{l}_i \\ &= \mathcal{S}_0(\boldsymbol{x})\,\boldsymbol{l}_0 + \sum_{i=1}^{A}\mathcal{S}_i(\boldsymbol{x})\,\boldsymbol{l}_i \qquad (2.21)\end{aligned}$$

Substitute Eq. (2.15) for the reaction load in Eq. (2.21) to prove the statement. □

2.2.3 Load parametrization

The alignment of actuators is not changing during the test, but the applied forces can be different for each block. Therefore we parametrize the load vector $\boldsymbol{l}$ as the product $p\,\boldsymbol{l}(\boldsymbol{\tau})$, where p and $\boldsymbol{\tau}$ are the following parameters:

1. $p \in \mathbb{R}$ - applied force that can be set for each actuator in every block separately
2. $\boldsymbol{\tau} \in \mathbb{R}^T$ - alignment parameters that are specified only once for each actuator and remain immutable during the test.

Definition 2.2.7. An alignment function $\boldsymbol{l}(\boldsymbol{\tau}) : \mathbb{R}^T \mapsto \mathbb{R}^6$ is a continuously differentiable function that maps alignment parameters $\boldsymbol{\tau} \in \mathbb{R}^T$ to load $\boldsymbol{l}$, where T is the number of alignment variables for the selected model (see Example 2.2.9).

Corollary 2.2.8. *The stress vector $\boldsymbol{\sigma}(\boldsymbol{x})$ under assumptions of Lem. 2.2.3 with load parameters p_i and $\boldsymbol{\tau}_i$ for the actuators $i = 1, \ldots, A$ has the following form:*

$$\boldsymbol{\sigma}(\boldsymbol{x}) = \sum_{i=1}^{A} p_i\,\mathcal{B}_i(\boldsymbol{x})\,\boldsymbol{l}(\boldsymbol{\tau}_i), \qquad (2.22)$$

where the matrix $\mathcal{B}_i(\boldsymbol{x}) = \mathcal{S}_i(\boldsymbol{x}) - \mathcal{S}_0(\boldsymbol{x})\mathcal{D}_i$ contains stresses generated by both actuators load and the reaction load of the mounted attachment.

Example 2.2.9 (Extender)**.** An actuator, installed in the attachment directly, can only apply forces, no moments. We can generate an additional torque if we use an extender $\boldsymbol{E}$ as a lever (Fig. 2.9) and then apply force $\boldsymbol{F}$ to its free end.

$$\begin{aligned}\boldsymbol{E}(r,\phi,\theta) &= [r\sin\theta\cos\phi,\ r\sin\theta\sin\phi,\ r\cos\theta]^T \\ \boldsymbol{F}(\phi',\theta') &= \left[\sin\theta'\cos\phi',\ \sin\theta'\sin\phi',\ \cos\theta'\right]^T \qquad (2.23)\end{aligned}$$

where r, ϕ and θ are the length, the azimuth and the inclination of the extender $\boldsymbol{E}$, respectively, and ϕ' and θ' are the azimuth and inclination of the force vector $\boldsymbol{F}$.

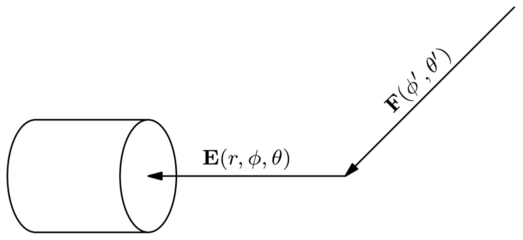


Figure 2.9: Extender installed in the attachment

Hence the resulting applied load can be expressed in the following form:

$$\boldsymbol{l}(\left[r,\phi,\theta,\phi',\theta'\right]) = \begin{bmatrix} \left(\boldsymbol{F}^T\boldsymbol{E} \;/\; |\boldsymbol{E}|^2\right)\boldsymbol{E} \\ \boldsymbol{E}\times\boldsymbol{F} \end{bmatrix} \tag{2.24}$$

Thus, the direction function in case of using extenders is given by Eqs. (2.23), (2.24) and depends on alignment variables $\boldsymbol{\tau} = [r,\phi,\theta,\phi',\theta'] \in \mathbb{R}^5$.

2.2.4 Damage simulation

The combination of the damage model defined by Eq. (2.14) and the stress model defined by Eq. (2.22) yields the formulas to compute damage values from the parameters of the test rig.

Corollary 2.2.10. *Damage d in case of multiaxial arbitrary mean stress signal with B blocks, generated by the actuators $i = 1,\dots,A$ with the alignment parameters $\boldsymbol{\tau}_i$ and the applied forces $p_{1,bi}$ and $p_{2,bi}$, corresponding to the turning points of the stress-strain cylces in block $b = 1,\dots,B$, can be computed as follows:*

$$d = \max_{\alpha} \sum_{b=1}^{B} n_b g\left[h(\sigma_{1,b}(\alpha),\sigma_{2,b}(\alpha))\right], \tag{2.25}$$

$$\sigma_{j,b} = \boldsymbol{c}(\alpha)^T \sum_{i=1}^{A} p_{j,bi}\,\mathcal{B}_i\,\boldsymbol{l}(\boldsymbol{\tau}_i), \quad \textit{for} \quad j = 1,\,2, \tag{2.26}$$

where g is a damage curve, h is a mean stress correction and $\boldsymbol{l}$ is an alignment function.

We finish the description of the physical model of the fatigue damage. Eqs. (2.25)–(2.26) fully describe the general damage computation that we will use in the optimization later.

2.3 Optimization Model

So far we have described all necessary components of the stress and damage simulation. Finally, we formulate an optimization problem and discuss what state variables, criteria and constraints it may have.

2.3.1 Optimization criterion

The quality of the damage approximation can be improved by minimizing the functional

$$f = \frac{d}{d^{\text{ref}}} + \frac{d^{\text{ref}}}{d}, \tag{2.27}$$

where d^{ref} is the reference damage value, that was acquired from FEM simulation using real-world load signals.

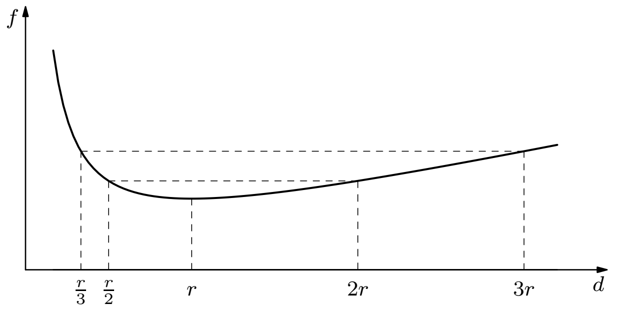


Figure 2.10: Quality criterion

Remark 2.3.1. The functional f considers the ratio of the achieved damage and the reference damage rather than the difference, which is closer to the logarithmic behavior of the fatigue life (see Fig. 2.10). In addition it treats underestimation and overestimation of the ratio in a similar way.

2.3.2 State variables

The component that we intend to test, is mounted to the test rig and has A actuators installed. Let the loading signal consist of B blocks. Then we have the following continuous variables:

- $\boldsymbol{p}_{1,b}$, $\boldsymbol{p}_{2,b} \in \mathbb{R}^A$ for $b = 1, \dots, B$ – applied forces in block b, corresponding to the first and second turning points of the load cycle in block b. The vector $\bar{\boldsymbol{p}} = [\boldsymbol{p}_{1,1}, \boldsymbol{p}_{2,1}, \dots, \boldsymbol{p}_{1,B}, \boldsymbol{p}_{2,B}] \in \mathbb{R}^{2AB}$ summarizes the applied forces.

- $\boldsymbol{\tau}_i \in \mathbb{R}^T$ for $i = 1, \dots, A$ – alignment parameters for each active actuator i. The vector $\bar{\boldsymbol{\tau}} = [\boldsymbol{\tau}_1, \dots, \boldsymbol{\tau}_A] \in \mathbb{R}^{AT}$ collects all alignment variables.

Let us introduce the state vector $\boldsymbol{z} = [\bar{\boldsymbol{p}}, \bar{\boldsymbol{\tau}}] \in \mathbb{R}^{A(2B+T)}$ that aggregates all load optimization parameters.

Remark 2.3.2. One could also take into account where the attachments with the mounts and where active actuators are. This would add further discrete variables to our model. However, there are not so many possibilities to vary the configuration of the test rig due to the low number of attachments and often there is only one obvious choice or few reasonable choices which are easy to enumerate.

2.3.3 Target functional

Let K denote the number of given hot spots $\boldsymbol{x}_k$ with reference damage values d_k^{ref} and the corresponding damage quality criterion f_k for $k = 1, \ldots, K$. We want to achieve the best quality of damage approximation at all hot spots, but there does not always exist a solution that minimizes every criterion f_k for $k = 1, \ldots, K$ simultaneously. Hence the test rig optimization problem is a multi-criteria minimization problem and we should look for Pareto optimal solutions [28], which are characterized by the fact that none of the criteria can be improved without worsening at least one of the others. Therefore, we use the criterion (2.27) to formulate the multi-criteria problem, depending on the load parameters $\boldsymbol{z}$:

$$\min_{\boldsymbol{z}} \, [f_1(\boldsymbol{z}), \ldots, f_K(\boldsymbol{z})], \tag{2.28}$$

$$f_k(\boldsymbol{z}) = \frac{d_k(\boldsymbol{z})}{d_k^{\text{ref}}} + \frac{d_k^{\text{ref}}}{d_k(\boldsymbol{z})}, \tag{2.29}$$

where f_k models the quality of the damage approximation at a hot spot $\boldsymbol{x}_k$ and damage $d_k(\boldsymbol{z})$ is produced by multiaxial arbitrary mean loading signal described above by the Eqs. (2.25)–(2.26):

$$d_k(\boldsymbol{z}) = \max_{\alpha_k} \sum_{b=1}^{B} n_b g\left[h(\sigma_{1,kb}(\alpha_k, \boldsymbol{z}), \sigma_{2,kb}(\alpha_k, \boldsymbol{z}))\right], \tag{2.30}$$

$$\sigma_{j,kb}(\alpha_k, \boldsymbol{z}) = \boldsymbol{c}(\alpha_k)^T \boldsymbol{\sigma}_{j,kb}(\bar{\boldsymbol{p}}, \bar{\boldsymbol{\tau}}), \tag{2.31}$$

$$\boldsymbol{\sigma}_{j,kb}(\bar{\boldsymbol{p}}, \bar{\boldsymbol{\tau}}) = \sum_{i=1}^{A} p_{j,bi} \, \mathcal{B}_{ki} \, \boldsymbol{l}(\boldsymbol{\tau}_i), \quad \text{for} \quad j = 1,\, 2, \tag{2.32}$$

where the state vector $\boldsymbol{z} = [\bar{\boldsymbol{p}}, \bar{\boldsymbol{\tau}}]$.

Remark 2.3.3. In general, there is no unique optimal solution in the multi-criteria setup. Instead we have the Pareto set, i.e. the set of Pareto-optimal solutions, which is usually of infinite size. These solutions can be obtained by an appropriate scalarization of the vector of criteria to a single new criterion [28]. This yields a non-linear program, which can be solved by standard techniques. In this work the multi-criteria problem (2.28) is scalarized using linear no-preference scalarization (each of the hot spots has the same importance) to concentrate on the properties of the nonlinear optimization:

$$\begin{aligned} &\min_{\boldsymbol{z}} F(\boldsymbol{z}) \\ &F(\boldsymbol{z}) = \sum_{k=1}^{K} f_k(\boldsymbol{z}) \end{aligned} \tag{2.33}$$

Although the target functional of the minimization problem is described by Eqs. (2.29)–(2.33), we need to provide constraints to set the problem.

2.3.4 Constraints and bounds

State variables in the test rig problem cannot be infinite and should be bounded to have a real world meaning. So we impose box constraints on the applied forces $\boldsymbol{p}_{1,b}$, $\boldsymbol{p}_{2,b}$ in each block $b = 1, \ldots, B$ as well as on actuators alignment variables $\boldsymbol{\tau}_i$ for every attachment $i = 1, \ldots, A$:

$$\begin{aligned} -\boldsymbol{p}^{\max} \leq \boldsymbol{p}_{j,b} \leq \boldsymbol{p}^{\max} \in \mathbb{R}^A \\ \boldsymbol{\tau}_i^{\min} \leq \boldsymbol{\tau}_i \leq \boldsymbol{\tau}_i^{\max} \in \mathbb{R}^T \end{aligned} \tag{2.34}$$

Note, that either positive or negative forces $\boldsymbol{p}_{j,b}$ can be applied and only the absolute value is bounded by $\boldsymbol{p}^{\max}$. Also we provide constraints for the components of the stress vectors $\boldsymbol{\sigma}_{j,kb}$:

$$-\boldsymbol{e}\sigma^{\max} \leq \boldsymbol{\sigma}_{j,kb} \leq \boldsymbol{e}\sigma^{\max} \in \mathbb{R}^3 \tag{2.35}$$

Contradictory constraints can lead to an empty feasible set, when there exists no state vector satisfying all constraints simultaneously.

Theorem 2.3.4. *The feasible set of the optimization problem Eqs. (2.28)–(2.33) is nonempty, bounded and convex.*

Proof. The optimization parameters $\bar{\boldsymbol{p}}$ and $\bar{\boldsymbol{\tau}}$ are linearly bounded by (2.34) from above and below and the stress constraints (2.35) are linear, hence the feasible set is convex and bounded. The applied powers vector $\bar{\boldsymbol{p}} = 0$ belongs to the feasible set, since all vector stresses $\boldsymbol{\sigma}_{j,kb}(\boldsymbol{0}, \bar{\boldsymbol{\tau}}) = 0$ and, therefore, satisfy the inequalities (2.35). □

Eqs. (2.28)–(2.33) provide a feasible formulation of the test rig optimization problem. The next section is devoted to the validation of the model by studying the behaviour of a simple cases.

2.4 Single Hotspot Solution

Previously, we combined all necessary tools in one complicated nonlinear optimization program, defined by Eqs. (2.28)–(2.33). In this section we check, whether the derived model's behaviour is acceptable and realistic. The validation of the model is done by considering solutions of the simple two-dimensional case with only one hot spot, one actuator and a constant fully reversed loading. The unit load stresses are computed by solving the linear elasticity equation with the finite element method. The reference damage field is generated from randomly created sequence of applied powers. We analyze this model both theoretically and numerically.

2.4.1 Two-dimensional model

Assume, that the number of hotspots $K = 1$ and damage approximation is done by a single actuator, generating the fully reversed constant loading, i.e. the number of blocks in the signal $B = 1$ and stress-strain cycles have zero mean. Selected assumptions simplify the general test rig problem (2.28)–(2.33) as follows:

$$\min_{p,\tau} f(p,\tau), \tag{2.36}$$

$$f(p,\tau) = \frac{d(p,\tau)}{d^{\text{ref}}} + \frac{d^{\text{ref}}}{d(p,\tau)}, \tag{2.37}$$

$$d(p,\tau) = n \max_{\alpha} g\left(p\left|\boldsymbol{c}(\alpha)^T \mathcal{B}\,\boldsymbol{l}(\tau)\right|\right), \tag{2.38}$$

where the matrix $\mathcal{B} \in \mathbb{R}^{3\times 2}$ contains two stress vectors, corresponding to the applied unit loads f_x and f_y, the function g is a damage curve (see Def. 2.1.6) and $\boldsymbol{c}(\alpha)$ denotes the critical plane normal (see Def. 2.1.15):

$$\boldsymbol{c}(\alpha) = [1+\cos\alpha, 1-\cos\alpha, 2\sin\alpha]/2 \tag{2.39}$$

The constants d^{ref} and n are the reference damage to achieve and the number of stress-strain cycles in the loading signal. The alignment function $\boldsymbol{l}(\tau)$ does not take the moments m_x and m_y into account, so, it balances only the ratio of the unit forces f_x and f_y:

$$\boldsymbol{l}(\tau) = [\sin\tau, \cos\tau]^T \tag{2.40}$$

The number of optimization parameters is, hence, reduced to only two:

- $p \in [0, p^{\max}]$ – applied force for the actuator
- $\tau \in [0, 2\pi]$ – angle of the actuator

The problem (2.36)–(2.38) belongs to the class of bilevel programs, i.e. for every instance of the outer state variables p and τ we have to solve an inner maximization problem with respect to α. In this particular case the bilevel formulation can be converted to the single level program by solving the inner problem analytically.

2.4.2 Single level formulation

The original formulation (2.36)–(2.38) can be simplified, if we get rid of the inner maximization problem. Luckily, it is possible in this case.

The critical plane optimization program for the fixed outer variables p and τ reads as follows:

$$\max_{\alpha} d(\alpha), \tag{2.41}$$

$$d(\alpha) = g\left(p\,|\boldsymbol{c}(\alpha)^T\boldsymbol{\sigma}|\right), \tag{2.42}$$

where the stress vector $\boldsymbol{\sigma} = \mathcal{B}\,\boldsymbol{l}(\tau)$. The physical meaning of $d(\alpha)$ here is the damage, induced by a single stress-strain cycle in loading signal.

Theorem 2.4.1. *The following holds for the critical plane optimization problem (2.41)–(2.42):*

$$\max_{\alpha} d(\alpha) = g\left(\frac{1}{2}p\,\max\{|d_1|, |d_2|\}\right)$$

where the constants d_1 and d_2 are defined as:

$$d_1 := \sigma_1 + \sigma_2 + \sqrt{(\sigma_1 - \sigma_2)^2 + 4\sigma_3^2}$$

$$d_2 := \sigma_1 + \sigma_2 - \sqrt{(\sigma_1 - \sigma_2)^2 + 4\sigma_3^2}$$

Proof. The monotonicity of the damage curve g and positivity of applied power p yields the expression:

$$\max_{\alpha} d(\alpha) = \max_{\alpha} g\left(p\,|\boldsymbol{c}(\alpha)^T\boldsymbol{\sigma}|\right) = g\left(p\,\max_{\alpha}|\boldsymbol{c}(\alpha)^T\boldsymbol{\sigma}|\right) \tag{2.43}$$

Let us define the function $q(\alpha) := \boldsymbol{c}(\alpha)^T\boldsymbol{\sigma}$ and find its extrema. We differentiate $q(\alpha)$ and substitute the critical plane normal definition Eq.(2.39) to get the equation:

$$q'(\alpha) = \boldsymbol{c}'(\alpha)^T\boldsymbol{\sigma} = \frac{1}{2}(\sigma_2 - \sigma_1)\sin\alpha + \sigma_3\cos\alpha = 0 \tag{2.44}$$

Eq. (2.44) has two roots:

$$\alpha_1 = \arctan(A/B), \tag{2.45}$$

$$\alpha_2 = \arctan(A/B) + \pi, \tag{2.46}$$

where $A := 2\sigma_3$ and $B := \sigma_1 - \sigma_2$. Hence, the maximum is given by the absolute value of either $q(\alpha_1)$ or $q(\alpha_2)$. Let us expand the definition of $q(\alpha)$ and write it in terms of A and B for convenience.

$$\begin{aligned} q(\alpha) = \boldsymbol{c}(\alpha)^T\boldsymbol{\sigma} &= 0.5\,[\sigma_1 + \sigma_2 + (\sigma_1 - \sigma_2)\cos\alpha + 2\sigma_3\sin\alpha] \\ &= 0.5\,[\sigma_1 + \sigma_2 + B\cos\alpha + A\sin\alpha] \\ &= 0.5\left[\sigma_1 + \sigma_2 + B\cos\alpha\left(1 + \frac{A}{B}\tan\alpha\right)\right] \end{aligned} \tag{2.47}$$

Substitute the expression (2.45) and (2.46) for the local optimal points α_1 and α_2 in (2.47):

$$\begin{aligned} q_{1,2} &= 0.5\left[\sigma_1 + \sigma_2 \pm \frac{B}{\sqrt{1 + \frac{A^2}{B^2}}}\left(1 + \frac{A^2}{B^2}\right)\right] \\ &= 0.5\left[\sigma_1 + \sigma_2 \pm \sqrt{B^2 + A^2}\right] \\ &= 0.5\left[\sigma_1 + \sigma_2 \pm \sqrt{(\sigma_1 - \sigma_2)^2 + 4\sigma_3^2}\right] \end{aligned} \tag{2.48}$$

Combining the result (2.43) and the formula (2.48) for the extrema of the function $q(\alpha) := \boldsymbol{c}(\alpha)^T\boldsymbol{\sigma}$, we prove the statement. □

Lemma 2.4.2. *The alignment function $\boldsymbol{l}(\tau) = [\sin\tau, \cos\tau]^T$ for $\tau \in [0, 2\pi]$ in problem (2.36)–(2.38) can be restricted to a semicircle for $\tau \in [\tau_0, \tau_0 + \pi]$ without decreasing the feasible set.*

Proof. It is enough to prove that $d(p, \tau) = d(p, \tau + \pi)$:

$$\begin{aligned} d(p, \tau) &= n \max_{\alpha} g\left(p\,|\boldsymbol{c}(\alpha)^T\mathcal{B}\,\boldsymbol{l}(\tau)|\right), \\ &= n \max_{\alpha} g\left(p\,| - \boldsymbol{c}(\alpha)^T\mathcal{B}\,\boldsymbol{l}(\tau + \pi)|\right) \\ &= d(p, \tau + \pi), \end{aligned}$$

using the fact that $[\sin\tau, \cos\tau]^T = -[\sin(\tau + \pi), \cos(\tau + \pi)]^T$. □

Remark 2.4.3. We have complete freedom to choose τ_0 of our interest. The following theorems give an idea, how to do that.

Lemma 2.4.4. *For the stress vector $\boldsymbol{\sigma}(\tau) = \mathcal{B}\,\boldsymbol{l}(\tau)$ with the alignment function $\boldsymbol{l}(\tau) = [\sin\tau, \cos\tau]^T$ the sum of first two components $\sigma_1(\tau) + \sigma_2(\tau)$ is non-negative, if one of the following statements holds:*

1. *$\mathcal{B}_{11} + \mathcal{B}_{21} > 0$ and $\tau \in [\tau_0, \tau_0 + \pi]$,*
2. *$\mathcal{B}_{11} + \mathcal{B}_{21} < 0$ and $\tau \in [\tau_0 + \pi, \tau_0 + 2\pi]$,*

where $\tau_0 = -\arctan\left[(\mathcal{B}_{12} + \mathcal{B}_{22})/(\mathcal{B}_{11} + \mathcal{B}_{21})\right]$.

Proof. Consider the sum of the first two stress components:

$$\sigma_1(\tau) + \sigma_2(\tau) = (\mathcal{B}_{11} + \mathcal{B}_{21}) \sin\tau + (\mathcal{B}_{12} + \mathcal{B}_{22}) \cos\tau = 0 \tag{2.49}$$

We get two solutions of Eq. (2.49):

$$\begin{aligned} \tau_1 &= -\arctan B/A =: \tau_0, \\ \tau_2 &= -\arctan B/A + \pi = \tau_0 + \pi, \end{aligned}$$

where the constants $A := \mathcal{B}_{11} + \mathcal{B}_{21}$ and $B := \mathcal{B}_{12} + \mathcal{B}_{22}$. The roots τ_0 and $\tau_0 + \pi$ define intervals $[\tau_0, \tau_0 + \pi]$ and $[\tau_0 + \pi, \tau_0 + 2\pi]$, but we do not know, where $\sigma_1(\tau) + \sigma(\tau)$ is non-negative. We calculate the expression at $\tau_0 + \pi/2$ to check its sign:

$$\begin{aligned} \sigma_1(\tau + \pi/2) + \sigma_2(\tau + \pi/2) &= A\cos\tau_0 - B\sin\tau_0 \\ &= A\cos\tau_0\,(1 + B^2/A^2) \\ &= A\sqrt{1 + B^2/A^2} \\ &= \operatorname{sgn} A\sqrt{A^2 + B^2} \end{aligned}$$

Hence $\sigma_1(\tau) + \sigma(\tau)$ is non-negative on the interval $[\tau_0, \tau_0 + \pi]$, when $\operatorname{sgn} A = \operatorname{sgn}(\mathcal{B}_{11} + \mathcal{B}_{21}) \geq 0$, which proves the statement of the lemma. □

Theorem 2.4.5. *For the critical plane optimization problem (2.41)–(2.42) with the stress vector $\boldsymbol{\sigma}(\tau) = \mathcal{B}\boldsymbol{l}(\tau)$ and the alignment function $\boldsymbol{l}(\tau) = [\sin\tau, \cos\tau]^T$ the following holds:*

$$d(p,\tau) = ng\left(\frac{p}{2}\left|\sigma_1(\tau) + \sigma_2(\tau) + \sqrt{(\sigma_1(\tau) - \sigma_2(\tau))^2 + 4\sigma_3^2(\tau)}\right|\right),$$

if one of the statements below is true:

1. $\mathcal{B}_{11} + \mathcal{B}_{21} > 0$ *and* $\tau \in [\tau_0, \tau_0 + \pi]$,
2. $\mathcal{B}_{11} + \mathcal{B}_{21} < 0$ *and* $\tau \in [\tau_0 + \pi, \tau_0 + 2\pi]$,

where $\tau_0 = -\arctan\left[(\mathcal{B}_{12} + \mathcal{B}_{22})/(\mathcal{B}_{11} + \mathcal{B}_{21})\right]$.

Proof. From the Theorem 2.4.1 we know, that

$$\max_{\alpha} d(p,\tau) = n\,g\left(\frac{1}{2}p\,\max\{|d_1(\tau)|, |d_2(\tau)|\}\right)$$

where the functions $d_1(\tau)$ and $d_2(\tau)$ are defined as:

$$\begin{aligned} d_1 &= \sigma_1(\tau) + \sigma_2(\tau) + \sqrt{(\sigma_1(\tau) - \sigma_2(\tau))^2 + 4\sigma_3^2(\tau)} \\ d_2 &= \sigma_1(\tau) + \sigma_2(\tau) - \sqrt{(\sigma_1(\tau) - \sigma_2(\tau))^2 + 4\sigma_3^2(\tau)} \end{aligned}$$

Obviously, $|d_1(\tau)| \geq |d_2(\tau)|$, when $\sigma_1(\tau) + \sigma_2(\tau)$ is non-negative. Lemma 2.4.4 provides the desired conditions. □

Remark 2.4.6. The above theorem combined with Lemma 2.4.2 makes it possible to forget about the critical plane maximization problem in the formulation (2.36)–(2.38).

Corollary 2.4.7. *The optimization problem (2.36)–(2.38) is equivalent to the following one:*

$$\min_{p,\tau} f(p,\tau), \tag{2.50}$$

$$f(p,\tau) = \frac{d(p,\tau)}{d^{ref}} + \frac{d^{ref}}{d(p,\tau)}, \tag{2.51}$$

$$d(p,\tau) = ng\left(p|\sigma_f(\tau)|\right), \tag{2.52}$$

$$\sigma_f(\tau) = \frac{1}{2}\left(\sigma_1(\tau) + \sigma_2(\tau) + \sqrt{(\sigma_1(\tau) - \sigma_2(\tau))^2 + 4\sigma_3^2(\tau)}\right), \tag{2.53}$$

$$\boldsymbol{\sigma}(\tau) = \mathcal{B}\,[\sin\tau, \cos\tau]^T, \tag{2.54}$$

where the state variables have box constraints:

$$\begin{aligned} &p \in [0, p^{max}], \\ &\tau \in [\tau_0, \tau_0 + \pi], \quad \text{if} \quad \mathcal{B}_{11} + \mathcal{B}_{21} > 0 \\ &\tau \in [\tau_0 + \pi, \tau_0 + 2\pi], \quad \text{if} \quad \mathcal{B}_{11} + \mathcal{B}_{21} < 0 \\ &\tau_0 = -\arctan\left[(\mathcal{B}_{12} + \mathcal{B}_{22})/(\mathcal{B}_{11} + \mathcal{B}_{21})\right] \end{aligned} \tag{2.55}$$

Theorem 2.4.8. *If the scalar stress $\sigma_f(\tau^*) \neq 0$ for a fixed direction τ^*, then the model (2.50)–(2.54) has the following minimum point:*

$$p^* = \min\left\{p^{max}, \frac{1}{|\sigma_f(\tau)|}\, g^{-1}\left(\frac{d^{ref}}{n}\right)\right\}, \tag{2.56}$$

where g^{-1} is the inverse of the damage curve g.

Proof. The KKT necessary optimality conditions [5] for the problem (2.50)–(2.54) read as follows:

$$\left(\frac{1}{d^{\mathrm{ref}}} - \frac{d^{\mathrm{ref}}}{d(p)}\right) ng'(p|\sigma_f(\tau)|)\,|\sigma_f(\tau)| + \lambda_1 - \lambda_2 = 0 \tag{2.57}$$

$$\lambda_1(p - p^{\max}) = 0 \tag{2.58}$$

$$\lambda_2\, p = 0 \tag{2.59}$$

$$\lambda_1,\, \lambda_2 \geq 0 \tag{2.60}$$

where λ_1 and λ_2 are Lagrangian multipliers. Considering different index sets of active constraints of the system (2.57)–(2.60), we get the minimizer:

Case 1. If $\lambda_1 = 0$ and $\lambda_2 = 0$, then the resulting equation $d(p) = d^{\text{ref}}$, combined with the damage formula 2.52, leads to the following minimum point:

$$p_1^* = \frac{1}{|\sigma_f(\tau)|}\, g^{-1}\left(\frac{d^{\text{ref}}}{n}\right), \tag{2.61}$$

where p_1^* is, obviously, non-negative, but it can be greater than $p_{\max}$ and, hence, infeasible.

Case 2. If $\lambda_1 = 0$ and $\lambda_2 \neq 0$, then the optimum is attained at $p_2^* = p^{\max}$.

Case 3. If $\lambda_1 \neq 0$ and $\lambda_2 = 0$, then the force p_3^* and the damage $d(p_3^*)$ are both equal to zero. Thus, the target functional $f(p_3^*, \tau^*)$ is infinite.

Therefore, the combination of cases 1 and 2 proves the theorem. □

We have successfully converted the original bilevel problem (2.36)–(2.38) into the single-level formulation (2.50)–(2.54) that can be solved, using Theorem 2.4.8 for a discretized set of angles τ. Before solving the problem we have to generate the stress matrix $\mathcal{B}$ and the reference damage d^{ref}.

2.4.3 Numerical Results

This section is devoted to the numerical investigation of the optimization problem behaviour with the example of two components. We calculate unit load stresses from the linear elasticity equation and generate reference damage field from a randomly created sequence of applied powers. Choosing a spot with the maximal reference damage as a hot spot, we get all necessary data to solve the optimization problem.

Unit load stresses computation

Based on [39], we make a brief description of the elasticity equation and its weak formulation to compute unit load stresses for two elastic bodies.

Strong formulation Deformation of the linearly elastic body Ω, fixed on $\Gamma_0 \subset \partial\Omega$ under the load acting on $\Gamma \subset \partial\Omega$ is described by the equations of linear elasticity in 2D:

$$\operatorname{div} \sigma(\boldsymbol{u}(\boldsymbol{x})) = 0, \tag{2.62}$$

$$\sigma(\boldsymbol{u}(\boldsymbol{x}))\, \boldsymbol{n}(\boldsymbol{x}) = \boldsymbol{\beta}(\boldsymbol{x}) \quad \text{on} \quad \Gamma, \tag{2.63}$$

$$\boldsymbol{u}(\boldsymbol{x}) = 0 \quad \text{on} \quad \Gamma_0, \tag{2.64}$$

where $\boldsymbol{u} := (u, v) \in \mathbb{R}^2$ is the displacement of a point $\boldsymbol{x}$ relative to its origin, $\boldsymbol{\beta} \in \mathbb{R}^2$ is the stress acting on the boundary Γ and $\sigma(\boldsymbol{u}(\boldsymbol{x})) \in \mathbb{R}^{2\times 2}$ is the stress tensor, which satisfies Hooke's law:

$$\sigma = 2\mu\varepsilon + \lambda \operatorname{tr}(\varepsilon)\mathcal{I}_2, \tag{2.65}$$

where λ and μ are Lamé parameters and $\varepsilon \in \mathbb{R}^{2\times 2}$ is the strain tensor, indicating the ratio of the change in length of the component to its original length [31]:

$$\varepsilon = \frac{1}{2}((\nabla \boldsymbol{u})^T + \nabla \boldsymbol{u}) = \frac{1}{2}\begin{bmatrix} 2\partial_x u & \partial_y u + \partial_x v \\ \partial_y u + \partial_x v & 2\partial_y v \end{bmatrix} \tag{2.66}$$

Remark 2.4.9. Eq. (2.65) implies the formula for the stress tensor components:

$$\begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix} = \begin{bmatrix} \partial_x u(2\mu + \lambda) + \lambda \partial_y v & \mu(\partial_y u + \partial_x v) \\ \mu(\partial_y u + \partial_x v) & \partial_y v(2\mu + \lambda) + \lambda \partial_x u \end{bmatrix} \tag{2.67}$$

Weak formulation The finite element method needs a variational formulation, so, we multiply the linear elasticity equation (2.62) with the test function $\boldsymbol{v}$, that satisfies the homogeneous boundary condition $\boldsymbol{v} = 0$ on Γ_0 and then integrate, using the Green's formula and the expression for the strain tensor (2.66):

$$\begin{aligned} 0 &= \int\limits_{\Omega} \operatorname{div} \sigma \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} = \int\limits_{\partial\Omega} \boldsymbol{v}^T \sigma \boldsymbol{n} \, \mathrm{d}s - \int\limits_{\Omega} \sigma(\boldsymbol{u}) : \nabla \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} \\ &= \int\limits_{\Gamma} \boldsymbol{v}^T \boldsymbol{\beta} - \int\limits_{\Omega} \sigma(\boldsymbol{u}) : \varepsilon(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x} \\ &= \int\limits_{\Gamma} \boldsymbol{v}^T \boldsymbol{\beta} - \int\limits_{\Omega} \lambda(\operatorname{tr}(\varepsilon)\mathcal{I}) : \varepsilon(\boldsymbol{v}) + 2\mu\varepsilon(\boldsymbol{u}) : \varepsilon(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x} \\ &= \int\limits_{\Gamma} \boldsymbol{v}^T \boldsymbol{\beta} - \lambda \int\limits_{\Omega} \operatorname{div} \boldsymbol{u} \operatorname{div} \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} - 2\mu \int\limits_{\Omega} \varepsilon(\boldsymbol{u}) : \varepsilon(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x} \end{aligned} \tag{2.68}$$

The weak formulation (2.68) can be solved with FEM. In this work we use FreeFEM++ solver [16].

Unit load stresses Let us split the boundary Γ, where the load is applied, into several attachments Γ_i, such that $\Gamma = \bigcup\limits_{i=1}^{A} \Gamma_i$ and $\Gamma_i \cap \Gamma_j = \emptyset$. Therefore, Eq. (2.68) can be rewritten:

$$\sum_{i=1}^{A} \int\limits_{\Gamma_i} \boldsymbol{v}^T \boldsymbol{\beta}_i - \lambda \int\limits_{\Omega} \operatorname{div} \boldsymbol{u} \operatorname{div} \boldsymbol{v} \, \mathrm{d}\boldsymbol{x} - 2\mu \int\limits_{\Omega} \varepsilon(\boldsymbol{u}) : \varepsilon(\boldsymbol{v}) \, \mathrm{d}\boldsymbol{x} = 0 \tag{2.69}$$

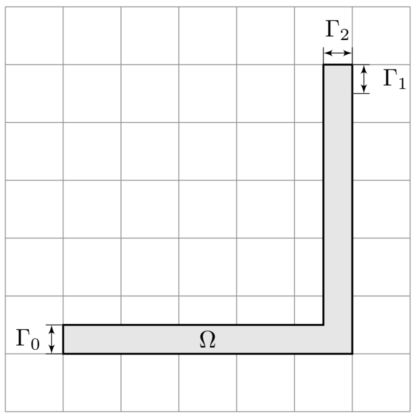
We are not particularly interested in the displacements $\boldsymbol{u}$, but rather in the stress tensors $\boldsymbol{\sigma}(\boldsymbol{u})$. That means we need to solve the weak formulation (2.69) and then apply Eq. (2.67) to evaluate the tensors.

Corollary 2.4.10. *Since the component is already fixed at Γ_0, the unit load stresses solutions $\boldsymbol{\sigma}(\boldsymbol{x}, \boldsymbol{\beta})$ for the surface traction $\boldsymbol{\beta}_1 = [1, 0]$ and $\boldsymbol{\beta}_2 = [0, 1]$ applied at the boundary $\Gamma_j(\boldsymbol{x})$ correspond to the first and the second columns of the stress matrices $\mathcal{B}_j(\boldsymbol{x}, \boldsymbol{\beta})$ in the node $\boldsymbol{x}$ for the j^{th} actuator, i.e.:*

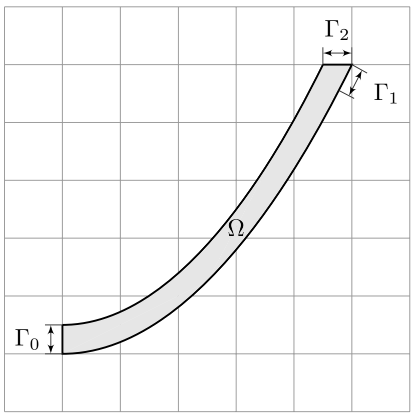
$$\mathcal{B}_j(\boldsymbol{x}) = (\boldsymbol{\sigma}_j(\boldsymbol{x}, \boldsymbol{\beta}_1), \boldsymbol{\sigma}_j(\boldsymbol{x}, \boldsymbol{\beta}_2)). \tag{2.70}$$

Eq. (2.70) allows us to calculate the unit load stress matrices, solving the elasticity equation for different boundary conditions.

Example 2.4.11. We consider the qualitative behavior of two components, namely, angle bar (see Fig. 2.11a) and countertop bar (see Fig. 2.11b). Both components have two attachments Γ_1 and Γ_2 and they are fixed at Γ_0.



(a) Angle bar



(b) Countertop bar

Figure 2.11: Cross sections of considered steel bars

We take the Lamé parameters of steel $\lambda = 1.07051 \cdot 10^{11}$ N/m^2 and $\mu = 7.75194 \cdot 10^{10}$ N/m^2 and rescale the magnitude of $\boldsymbol{\beta}$ to 0.02. These parameters are valid for both components.

The displaced mesh with the color map, showing the scalar stress field, generated by the different unit load cases, consists of 714 triangles for the angle bar (see Fig. 2.12a–2.15b) and 403 triagles for the countertop bar (see Fig. 2.16a–2.19b). The displacement $\boldsymbol{u}$ is multiplied by 10^8 for the angle bar and 10^7 for the countertop bar for better perception.

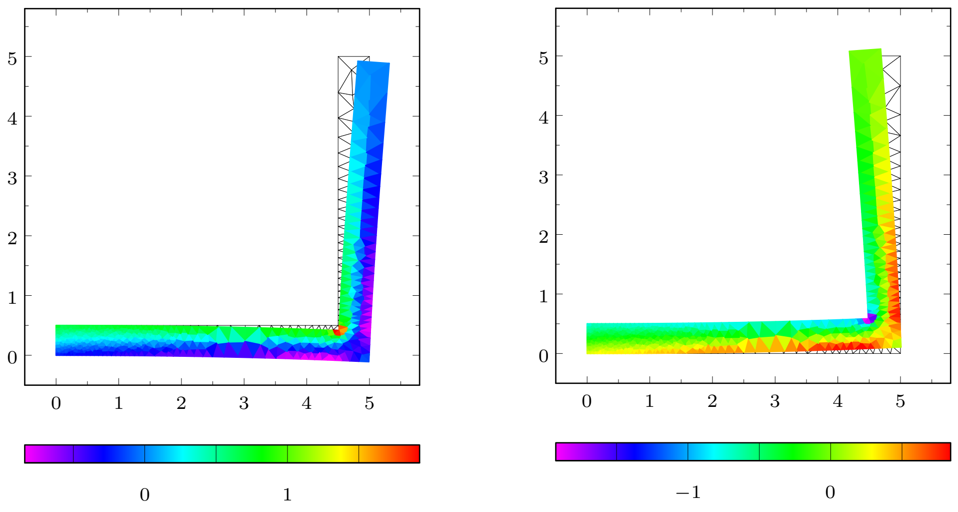


Figure 2.12: Scalar stress field $\boldsymbol{\sigma}^T \boldsymbol{e}$ for $\boldsymbol{\beta}_1 = [1, 0]^T$ and $\boldsymbol{\beta}_1 = [-1, 0]^T$

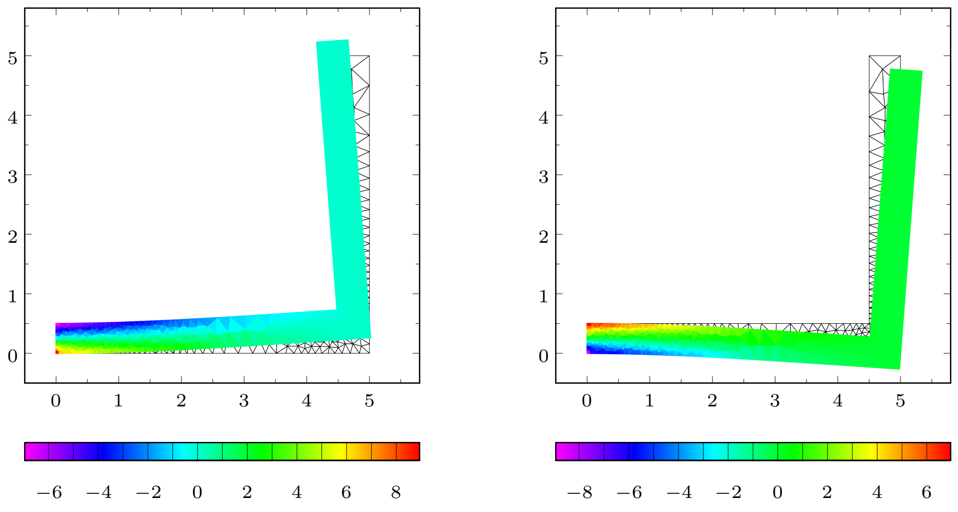


Figure 2.13: Scalar stress field $\boldsymbol{\sigma}^T \boldsymbol{e}$ for $\boldsymbol{\beta}_1 = [0, 1]^T$ and $\boldsymbol{\beta}_1 = [0, -1]^T$

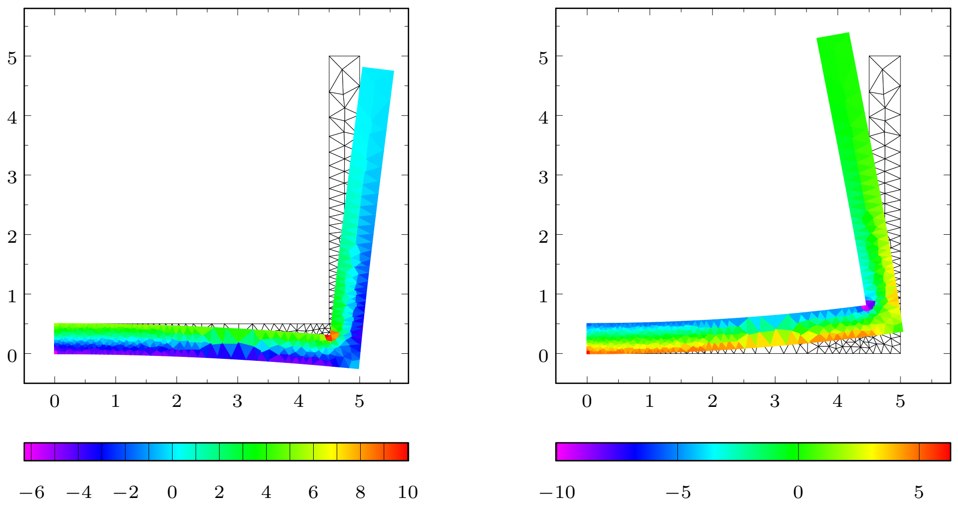


Figure 2.14: Scalar stress field $\boldsymbol{\sigma}^T \boldsymbol{e}$ for $\boldsymbol{\beta}_2 = [1, 0]^T$ and $\boldsymbol{\beta}_2 = [-1, 0]^T$

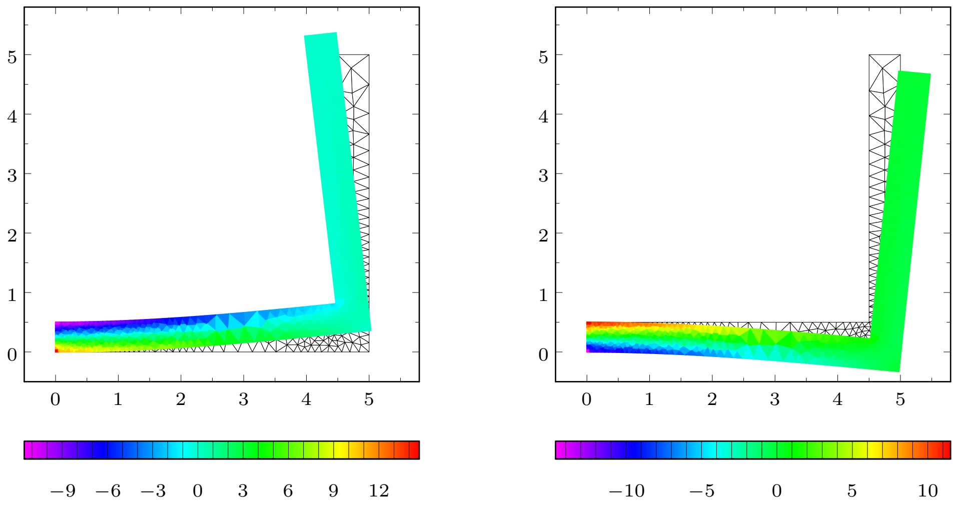


Figure 2.15: Scalar stress field $\boldsymbol{\sigma}^T \boldsymbol{e}$ for $\boldsymbol{\beta}_2 = [0, 1]^T$ and $\boldsymbol{\beta}_2 = [0, -1]^T$

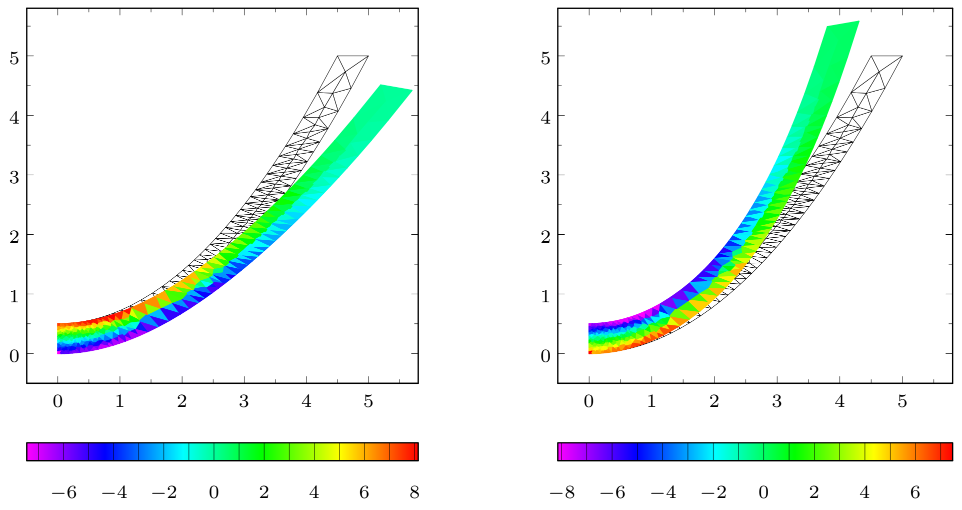


Figure 2.16: Scalar stress field $\boldsymbol{\sigma}^T \boldsymbol{e}$ for $\boldsymbol{\beta}_1 = [1, 0]^T$ and $\boldsymbol{\beta}_1 = [-1, 0]^T$

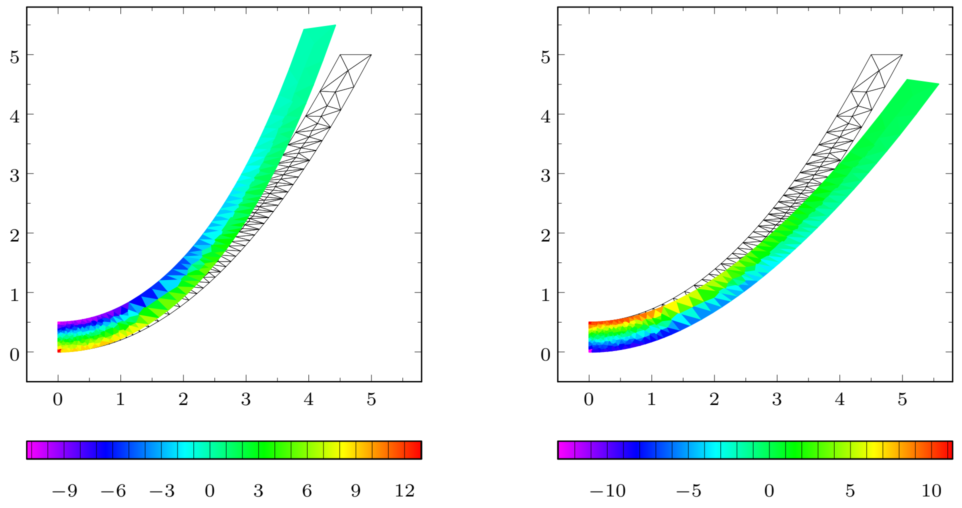


Figure 2.17: Scalar stress field $\boldsymbol{\sigma}^T \boldsymbol{e}$ for $\boldsymbol{\beta}_1 = [0, 1]^T$ and $\boldsymbol{\beta}_1 = [0, -1]^T$

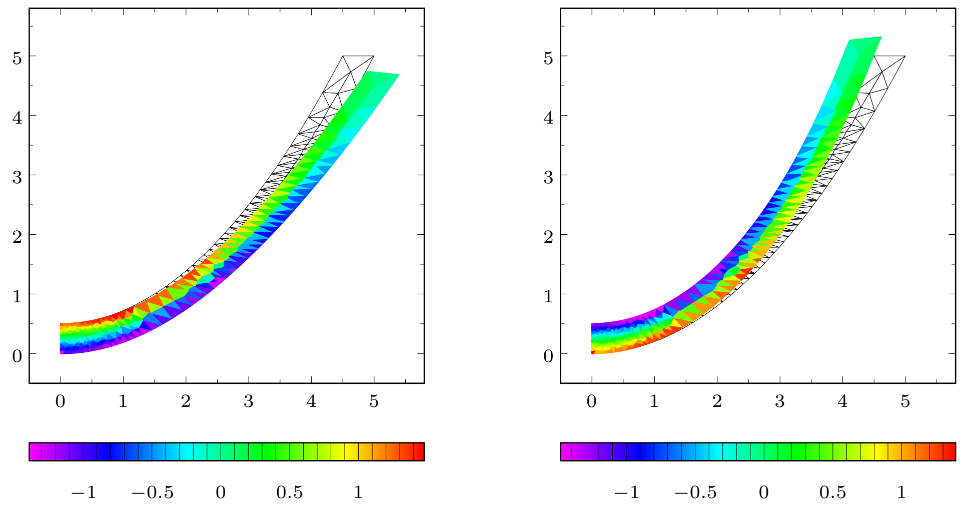


Figure 2.18: Scalar stress field $\boldsymbol{\sigma}^T \boldsymbol{e}$ for $\boldsymbol{\beta}_2 = [1, 0]^T$ and $\boldsymbol{\beta}_2 = [-1, 0]^T$

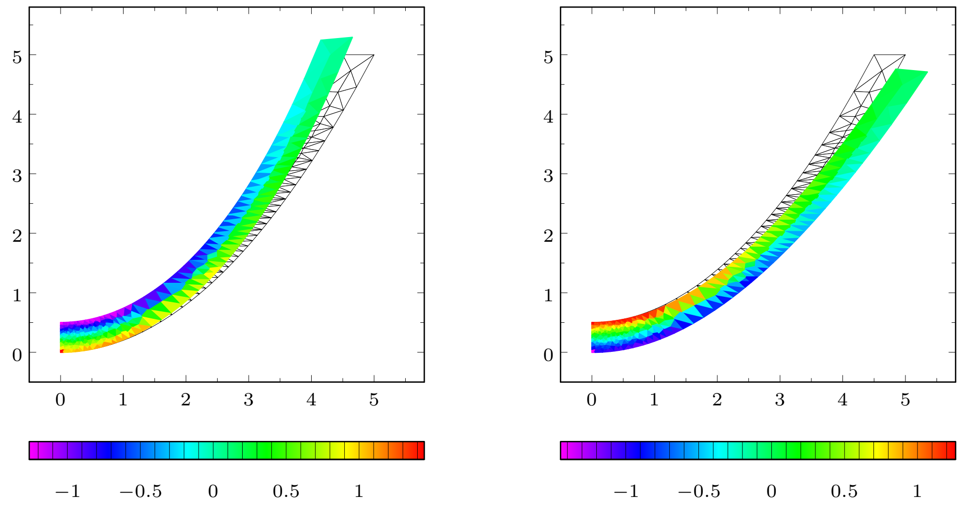


Figure 2.19: Scalar stress field $\boldsymbol{\sigma}^T \boldsymbol{e}$ for $\boldsymbol{\beta}_2 = [0, 1]^T$ and $\boldsymbol{\beta}_2 = [0, -1]^T$

Reference damage generation

In the previous section we discussed the unit load stresses calculation, but we still need the reference damage values to collect all necessary data for optimization.

The randomly generated multiaxial block general loading (for details, see Section 2.1.5 and Eqs. (2.25)–(2.26)), created by two actuators, is used to mimic the real world loading signals:

$$d^{\text{ref}} = \max_{w=1,\dots,W} \sum_{b=1}^{B} n_b g\left[h(\sigma_{1,b}(\alpha), \sigma_{2,b}(\alpha))\right], \tag{2.71}$$

$$\sigma_{j,b} = \boldsymbol{c}(\frac{2\pi w}{W})^T \left(p_{j,b1}\, \mathcal{B}_1\, \boldsymbol{l}(\tau_{b1}) + p_{j,b2}\, \mathcal{B}_2\, \boldsymbol{l}(\tau_{b2})\right), \quad \text{for} \quad j = 1,\, 2, \tag{2.72}$$

where B is the number of blocks, τ_{b1} and τ_{b2} are angles of the forces $p_{j,b1}$ and $p_{j,b2}$, applied at the corresponing boundary. Both forces and angles are randomly generated for each block $b = 1, \dots, B$, using a uniform distribution. The critical plane optimization is discrete, i.e. we choose maximal damage among $W = 18$ planes as in [27]. The damage curve $g(\sigma_g)$ has one slope:

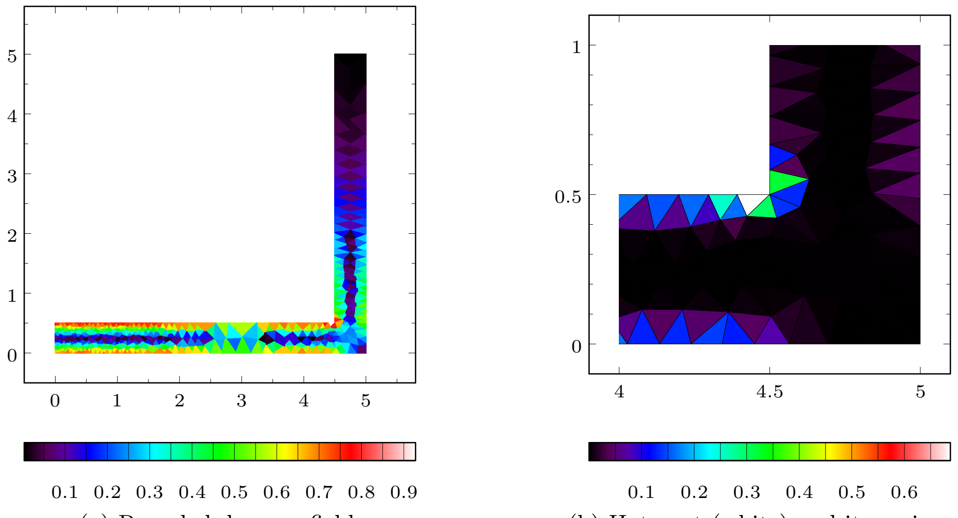
$$g(\sigma_f) = 10^{-6}\left(|\sigma_f|/75\right)^5 \tag{2.73}$$

and mean stress correction $h(\sigma_1, \sigma_2)$ is defined as follows:

$$h(\sigma_1, \sigma_2) = \begin{cases} 0.7\sigma_a & \text{for} \quad \sigma_a < -\sigma_m \\ \sigma_a + 0.3\sigma_m & \text{for} \quad \sigma_a \geq -\sigma_m \end{cases} \tag{2.74}$$

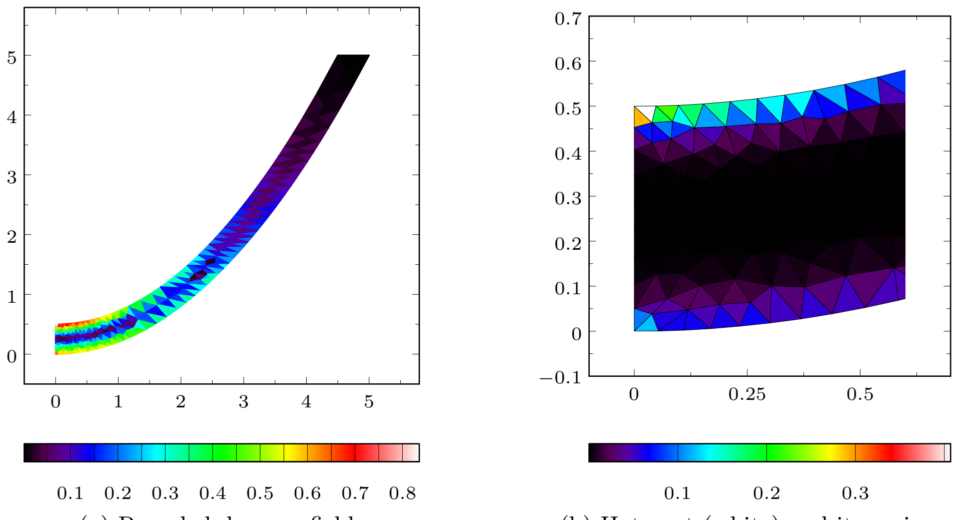
where $\sigma_a = |\sigma_1 - \sigma_2|/2$ and $\sigma_m = (\sigma_1 + \sigma_2)/2$ are the amplitude and mean of the stress-strain cycle respectively. Detailed description and physical meaning of the functions g and h can be found in Sections 2.1.2 and 2.1.3.

We define $p^{\max} = 12$ for both components and $B = 10^{12}$ for the angle bar and $B = 10^{10}$ for the countertop bar. Figs. 2.20a and 2.21a show the results of the reference damage generation.



(a) Rescaled damage field (b) Hot spot (white) and its region

Figure 2.20: Reference damage field $d^{\text{ref}}(\boldsymbol{x})$



(a) Rescaled damage field (b) Hot spot (white) and its region

Figure 2.21: Reference damage field $d^{\text{ref}}(\boldsymbol{x})$

Thus, the reference damage field for each polygon was computed. Each component has its spot with the maximal damage (see Figs. 2.20b and 2.21b). These spots are chosen as the hot spots for the optimization.

Results In this section we use previously calculated unit load stresses, reference damages and hot spots to solve the optimization problem (2.50)–(2.52). We apply the Theorem 2.4.8 to find the optimal applied force p_v^* for the discretized range of actuator's angles τ_v for $v = 1, \ldots, 180$. Then we choose the solution (p_v^*, τ_v) that minimizes the target functional $f(p, \tau)$.

Angle bar The minimum $f^* = 2.0$ was reached, using the single actuator, attached at the boundary Γ_1, i.e. the reference damage d^{ref} was successfully approximated (see Fig. 2.22b). At the same time, Fig. 2.22a shows that the component was fractured close to the fixation Γ_0.

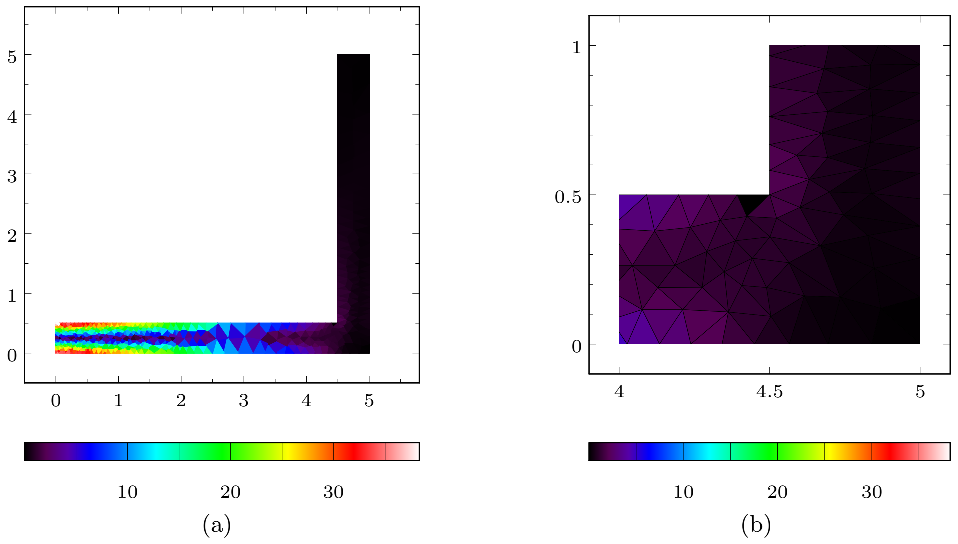
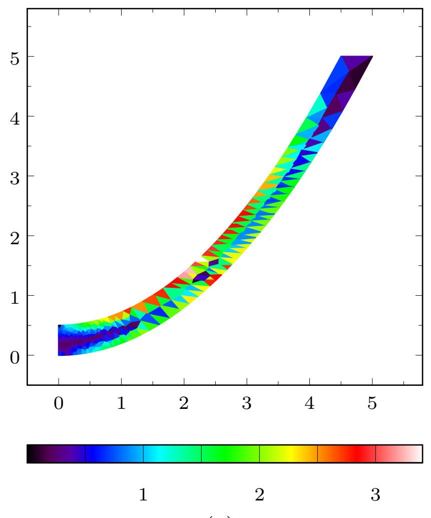
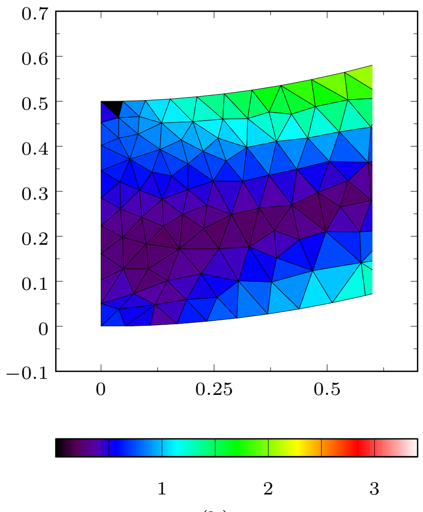


Figure 2.22: Achieved damage field $d(\boldsymbol{x})$

Countertop bar Similarly, the minimum $f^* = 2.0$ was reached, using the single actuator, attached at the boundary Γ_1 (see Fig. 2.23b), but now the component failed away from the fixation Γ_0, because the chosen hot spot was very close to it.



(a)



(b)

Figure 2.23: Achieved damage field $d(\boldsymbol{x})$

The results indicate that the single hot spot model with the constant fully reversed loading can provide acceptable damage approximation only in the neighborhood of the chosen hot spot even for simple geometries. Therefore, we should use more difficult models with several actuators, hot spots, blocks in the stress-strain signal and impose more constraints on the stress and damage values.

2.5 Conclusion

In this chapter we derived the test rig bilevel optimization problem that consists of the fatigue damage model for multiaxial block general loading signal, the method to compute stress vectors, corresponding to the turning points of the signal, and criterion, evaluating quality of the damage approximation at each hot spot.

We studied the properties of the model in the simplest case of a test rig with one hot spot and a single actuator, generating a constant fully reversed loading signal. As expected, the numerical investigation demonstrates realistic reference damage field, but poor approximation quality outside the hot spot neighborhood. That implies validity of the general model, although it also shows that we should avoid using simplified versions of the optimization model, since it does not provide suitable approximation quality even for ordinary geometries.

Chapter 3

Single Level Optimization

In the previous chapter we have described the test rig model and tested its validity. The derived optimization problem (2.28)–(2.33) belongs to the class of bilevel programs, since for each instance of the state vector we have to solve the inner optimization problem, finding a plane, where the fatigue damage is maximal. However, for some applications it is not necessary. If the hot spot is located at the seam-welded joints, then the angle of the critical plane is known in advance [26].

Assumption 3.0.1. *The critical plane angles are known at each hot spot $k = 1, \dots, K$, i.e. for every state vector $\boldsymbol{z}$ and the following inner maximization problem:*

$$d_k(\boldsymbol{z}) = \max_{\alpha} d_k(\boldsymbol{z}, \alpha)$$

a solution α_k^* is given and constant.

Global optimization problems are often convex with respect to a subset of the state variables, while the rest are fixed. The general test rig problem is not convex, but we can gain some useful properties, if we fix the alignment of each actuator. After that we can solve more regular problems for a discretized set of feasible alignment parameters.

Assumption 3.0.2. *The alignment variables $\boldsymbol{\tau}_i$ for every actuator $i = 1, \dots, A$ are given and immutable.*

This chapter contains the analysis of the test rig model under the Assumptions 3.0.1 and 3.0.2. Convexity of the damage function is proven. It allows to convert the problem into a DC (difference convex) program. Also we show that the test rig problem is a union of convex optimization problems on polyhedral cones, if it generates fully reversed constant loading. In order to detect such cones, a new algorithm is presented and proven to have worst case polynomial time complexity.

3.1 Model

Under the Assumptions 3.0.1 and 3.0.2 the general test rig problem (2.28)–(2.33) has predefined alignment parameters $\boldsymbol{\tau}_i$ for actuators $i = 1, \dots, A$ and critical plane angles α_k for hot spots $k = 1, \dots, K$. Thus, we need to optimize only the applied forces $\bar{\boldsymbol{p}} = [\boldsymbol{p}_{1,1}, \boldsymbol{p}_{2,1}, \dots, \boldsymbol{p}_{1,B}, \boldsymbol{p}_{2,B}] \in \mathbb{R}^{2AB}$. The problem is reduced to the following form:

$$\min_{\bar{\boldsymbol{p}}} \sum_{k=1}^{K} \frac{d_k(\bar{\boldsymbol{p}})}{d_k^{\text{ref}}} + \frac{d_k^{\text{ref}}}{d_k(\bar{\boldsymbol{p}})} \tag{3.1}$$

$$d_k(\bar{\boldsymbol{p}}) = \sum_{b=1}^{B} n_b g\left[h(\sigma_{1,kb}(\bar{\boldsymbol{p}}), \sigma_{2,kb}(\bar{\boldsymbol{p}}))\right], \tag{3.2}$$

$$\sigma_{j,kb}(\bar{\boldsymbol{p}}) = \boldsymbol{p}_{j,b}^T \boldsymbol{s}_k \quad \text{for} \quad j = 1,\, 2, \tag{3.3}$$

where $d_k^{\text{ref}} > 0$ denote the reference damage values, $n_b > 0$ are the numbers of stress-strain cycles per block and the vectors $\boldsymbol{s}_k \in \mathbb{R}^A$ already include unit load stresses, critical planes and alignment parameters, i.e. $s_{k,i} = \boldsymbol{c}(\alpha_k)^T \mathcal{B}_{ki}\, \boldsymbol{l}(\boldsymbol{\tau}_i)$. The function $g : \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+$ is the damage curve:

$$g(\sigma_f) := \frac{1}{N_s} \left\{ \begin{array}{ll} (\sigma_f/\sigma_s)^{k_1} & \text{for} \quad \sigma_f \le \sigma_s, \\ (\sigma_f/\sigma_s)^{k_2} & \text{for} \quad \sigma_f \ge \sigma_s, \end{array} \right. \tag{3.4}$$

where $k_1, k_2 > 2$ and $h : \mathbb{R}^2 \mapsto \mathbb{R}_0^+$ is the Goodman mean stress correction:

$$h(\sigma_1, \sigma_2) = \left\{ \begin{array}{ll} (1 - M)\, \sigma_a & \text{for} \quad \sigma_a < -\sigma_m \\ \sigma_a + M\sigma_m & \text{for} \quad \sigma_a \ge -\sigma_m \end{array} \right. \tag{3.5}$$

where $\sigma_a = |\sigma_1 - \sigma_2|/2$ and $\sigma_m = (\sigma_1 + \sigma_2)/2$ are the amplitude and mean of the stress-strain cycle respectively and the slope $M < 1$ is positive. Detailed description and physical meaning of these functions can be found in Sections 2.1.2 and 2.1.3.

3.1.1 Damage function convexity

In this section we discuss the properties of the damage function $d(\bar{\boldsymbol{p}})$ that we are using in the optimization problem (3.1)–(3.3):

$$d(\bar{\boldsymbol{p}}) = \sum_{b=1}^{B} n_b g\left[h(\sigma_{1,b}(\bar{\boldsymbol{p}}), \sigma_{2,b}(\bar{\boldsymbol{p}}))\right], \tag{3.6}$$

Finally, we prove that convexity and monotonicity of g imply convexity of the damage function. This important result can be used to convert the problem into DC program and, hence, apply more effective methods to solve it.

Since $d(\bar{\boldsymbol{p}})$ is a composition of several functions, we would like to recall some simple operations that preserve convexity, based on [7].

Lemma 3.1.1. *If f_1 and f_2 are convex functions then their pointwise maximum f, defined by*

$$f(\boldsymbol{x}) = \max\{f_1(\boldsymbol{x}), f_2(\boldsymbol{x})\}$$

with $\operatorname{dom} f = \operatorname{dom} f_1 \cap \operatorname{dom} f_2$*, is also convex.*

Lemma 3.1.2. *A piecewise-affine function $f(\boldsymbol{x})$ with L regions,*

$$f(\boldsymbol{x}) = \max\{\boldsymbol{a}_1^T\boldsymbol{x} + \boldsymbol{b}_1, \ldots, \boldsymbol{a}_L^T\boldsymbol{x} + \boldsymbol{b}_L\}$$

is convex since it is the pointwise maximum of affine functions.

Lemma 3.1.3. *Suppose $f : \mathbb{R}^n \mapsto \mathbb{R}$, $\mathcal{A} \in \mathbb{R}^{n\times m}$, and $\boldsymbol{b} \in \mathbb{R}^n$. Define $g : \mathbb{R}^m \mapsto \mathbb{R}$ by*

$$g(\boldsymbol{x}) = f(\mathcal{A}\boldsymbol{x} + \boldsymbol{b})$$

with $\operatorname{dom} g = \{\boldsymbol{x} \mid \mathcal{A}\boldsymbol{x} + \boldsymbol{b} \in \operatorname{dom} f\}$*. Then if f is convex, so is g.*

Lemma 3.1.4. *A nonnegative weighted sum of convex functions,*

$$f(\boldsymbol{x}) = \omega_1 f_1(\boldsymbol{x}) + \ldots + \omega_L f_L(\boldsymbol{x})$$

is convex.

Lemma 3.1.5. *Suppose $g : \mathbb{R}^k \mapsto \mathbb{R}$ and $h_i : \mathbb{R}^n \mapsto \mathbb{R}$ for $i = 1, \ldots, k$. Define $f : \mathbb{R}^n \mapsto \mathbb{R}$ by*

$$f(\boldsymbol{x}) = g(h(\boldsymbol{x})) = g(h_1(\boldsymbol{x}), \ldots, h_k(\boldsymbol{x}))$$

with $\operatorname{dom} h = \mathbb{R}^n$ *and* $\operatorname{dom} g = \mathbb{R}^k$*. Then f is convex if g is convex and non-decreasing in each argument, and h_i are convex for $i = 1, \ldots, k$.*

We prove convexity of the Goodman mean stress correction h with respect to the turning points σ_1 and σ_2. It is the only missing link to prove convexity of the damage function.

Lemma 3.1.6. *Goodman mean stress correction $h(\sigma_1, \sigma_2)$, defined by Eq. (3.5), is a pointwise maximum of linear functions*

$$\begin{aligned} h(\sigma_1, \sigma_2) = 0.5 \max\{&(1 - M)(\sigma_1 - \sigma_2),\ (1 + M)\sigma_1 - (1 - M)\sigma_2, \\ &(1 - M)(\sigma_2 - \sigma_1),\ (1 + M)\sigma_2 - (1 - M)\sigma_1\} \end{aligned} \tag{3.7}$$

and, thus, convex.

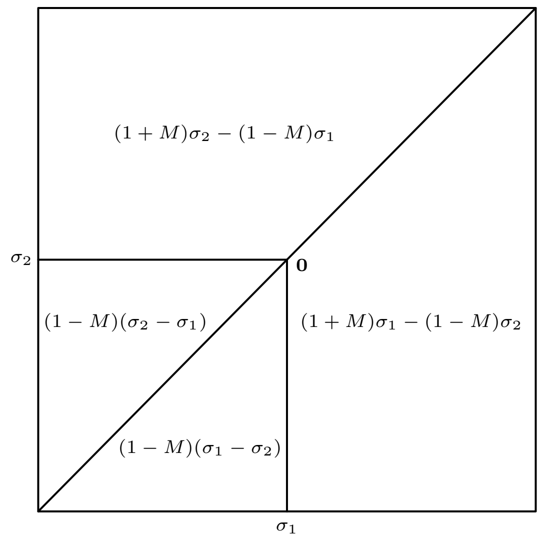
Proof. Consider the Formula (3.5):

$$h(\sigma_1, \sigma_2) = \frac{1}{2}\left\{ \begin{array}{lll} (1 - M)\,|\sigma_1 - \sigma_2| & \text{for} & |\sigma_1 - \sigma_2| < -(\sigma_1 + \sigma_2) \\ |\sigma_1 - \sigma_2| + M(\sigma_1 + \sigma_2) & \text{for} & |\sigma_1 - \sigma_2| \geq -(\sigma_1 + \sigma_2) \end{array} \right.$$

and expand the modulus $|\sigma_1 - \sigma_2|$:

$$h(\sigma_1,\sigma_2) = \frac{1}{2}\begin{cases} (1-M)\,(\sigma_1-\sigma_2) & \text{for} \quad \sigma_1 \geq \sigma_2 \quad \text{and} \quad \sigma_1 \leq 0 \\ (1-M)\,(\sigma_2-\sigma_1) & \text{for} \quad \sigma_1 \leq \sigma_2 \quad \text{and} \quad \sigma_2 \leq 0 \\ (1+M)\sigma_1 - (1-M)\sigma_2 & \text{for} \quad \sigma_1 \geq \sigma_2 \quad \text{and} \quad \sigma_1 \geq 0 \\ (1+M)\sigma_2 - (1-M)\sigma_1 & \text{for} \quad \sigma_1 \leq \sigma_2 \quad \text{and} \quad \sigma_2 \geq 0 \end{cases}$$

We get a piecewise linear function with four regions, depicted on Fig. 3.1.



$(1+M)\sigma_2 - (1-M)\sigma_1$
σ_2
$\mathbf{0}$
$(1-M)(\sigma_2-\sigma_1)$
$(1+M)\sigma_1 - (1-M)\sigma_2$
$(1-M)(\sigma_1-\sigma_2)$
σ_1

Figure 3.1: Domain of the Goodman mean stress correction $h(\sigma_1, \sigma_2)$

Introduce the function $f(\sigma_1, \sigma_2)$ as follows:

$$\begin{aligned} f(\sigma_1,\sigma_2) = 0.5 \max\{&(1-M)\,(\sigma_1-\sigma_2),\ (1+M)\,\sigma_1 - (1-M)\,\sigma_2, \\ &(1-M)\,(\sigma_2-\sigma_1),\ (1+M)\,\sigma_2 - (1-M)\,\sigma_1\} \end{aligned}$$

The function $f(\sigma_1, \sigma_2)$ is convex by Lemma 3.1.2, since it is a pointwise maximum of linear functions. It remains to prove that f and h are equal at each point (σ_1, σ_2).

Consider the case, when $\sigma_1 \geq \sigma_2$ and $\sigma_1 \leq 0$. Obviously, $h(\sigma_1, \sigma_2) = 0.5\,(1-M)(\sigma_1-\sigma_2)$. The following inequalities hold for the function $f(\sigma_1, \sigma_2)$:

$$\begin{aligned} &(1-M)\,(\sigma_1-\sigma_2) \geq (1-M)\,(\sigma_2-\sigma_1) \quad \text{for} \quad \sigma_1 \geq \sigma_2 \\ &(1+M)\,\sigma_1 - (1-M)\,\sigma_2 \geq (1+M)\,\sigma_2 - (1-M)\,\sigma_1 \quad \text{for} \quad \sigma_1 \geq \sigma_2 \\ &(1-M)\,(\sigma_1-\sigma_2) \geq (1+M)\,\sigma_1 - (1-M)\,\sigma_2 \quad \text{for} \quad \sigma_1 \leq 0 \end{aligned}$$

Therefore, the functions h and f are equal for $\sigma_1 \geq \sigma_2$ and $\sigma_1 \leq 0$. The other cases can be proven analogously. □

Theorem 3.1.7. *Damage model* $d(\boldsymbol{p}_{1,1},\ldots,\boldsymbol{p}_{2,B})$*:*

$$d(\boldsymbol{p}_{1,1},\ldots,\boldsymbol{p}_{2,B}) = \sum_{b=1}^{B} n_b g(h(\boldsymbol{p}_{1,b}^T\boldsymbol{s}, \boldsymbol{p}_{2,b}^T\boldsymbol{s})) \tag{3.8}$$

is convex, if the damage curve g *is convex and non-decreasing.*

Proof. Lemma 3.1.6 states the convexity of the Goodman mean stress correction $h(\sigma_1,\sigma_2)$. The composition of the convex function with the affine mapping is convex by Lemma 3.1.3, hence, the function $\bar{h}(\boldsymbol{p}_1,\boldsymbol{p}_2) := h(\boldsymbol{p}_1^T\boldsymbol{s},\boldsymbol{p}_2^T\boldsymbol{s})$ is also convex. A non-negative weighted sum of scalar compositions of non-decreasing and convex function g with convex $\bar{h}$ is convex, according to Lemma 3.1.4 and Lemma 3.1.5. □

The damage curve g, defined by Eq. (3.4), is a continuous piecewise monomial function with positive coefficients and non-negative domain, so it is non-decreasing. But is it really convex like we assumed in the theorem above? The answer is yes when $k_1 \le k_2$ and no otherwise. In the latter case, we can convexify it in the neighborhood of σ_s.

Lemma 3.1.8. *The continuous piecewise monomial function* $f(t)$ *with parameters* $k_1 > k_2 > 0$*:*

$$f(t) = \begin{cases} t^{k_1} & \text{for} \quad t \le 1, \\ t^{k_2} & \text{for} \quad t \ge 1, \end{cases} \tag{3.9}$$

can be convexified by the following continuously differentiable function:

$$\bar{f}(x) = \begin{cases} t^{k_1} & \text{for} \quad t \le l, \\ a_1(t-l)+a_0 & \text{for} \quad l \le t \le r, \\ t^{k_2} & \text{for} \quad t \ge r, \end{cases} \tag{3.10}$$

where the parameters l*,* r*,* a_0 *and* a_1 *are defined by*

$$\begin{aligned} a_0 &= l^{k_1} \\ a_1 &= k_1 l^{k_1-1} \\ l &= \left(\frac{k_1}{k_2}\right)^{\frac{k_2}{k_2-k_1}} \left(\frac{k_1-1}{k_2-1}\right)^{\frac{1-k_2}{k_2-k_1}} \\ r &= \left(\frac{k_1}{k_2}\right)^{\frac{k_1}{k_2-k_1}} \left(\frac{k_1-1}{k_2-1}\right)^{\frac{1-k_1}{k_2-k_1}} \end{aligned}$$

Proof. We have to solve the system of 4 equations that include conditions of the continuity of $f(t)$ and $f'(t)$ at the points l and r to find the coefficients:

$$a_0 = l^{k_1} \tag{3.11}$$

$$a_0 + a_1(r-l) = r^{k_2} \tag{3.12}$$

$$a_1 = k_1 l^{k_1-1} \tag{3.13}$$

$$a_1 = k_2 r^{k_2-1} \tag{3.14}$$

Substitute Eqs. (3.11), (3.13) and (3.14) in (3.12) and get the following system:

$$a_0 = l^{k_1} \tag{3.15}$$

$$a_1 = k_1 l^{k_1-1} \tag{3.16}$$

$$k_1 l^{k_1-1} = k_2 r^{k_2-1} \tag{3.17}$$

$$l^{k_1}(k_1-1) = r^{k_2}(k_2-1) \tag{3.18}$$

Solution of the Eqs. (3.17) and (3.18) with respect to l and r yields the desired result. The function $\bar{f}(t)$ is continuously differentiable and convex by construction. □

Corollary 3.1.9. *The function $\bar{f}$ in Lemma 3.1.8 has a convex reciprocal $1/\bar{f}$.*

Proof. The reciprocal of $\bar{f}$ is a piecewise function:

$$1/\bar{f}(x) = \begin{cases} t^{-k_1} & \text{for} \quad t \le l, \\ 1/(a_1(t-l)+a_0) & \text{for} \quad l \le t \le r, \\ t^{-k_2} & \text{for} \quad t \ge r, \end{cases} \tag{3.19}$$

or equivalently, since $k_1 > k_2$:

$$1/\bar{f}(x) = \max\left\{t^{-k_1},\, t^{-k_2},\, 1/(a_1(t-l)+a_0)\right\} \tag{3.20}$$

which is convex by Lemma 3.1.1. □

Corollary 3.1.10. *The functions f and $\bar{f}$ in Lemma 3.1.8 have the following distance:*

$$\|f - \bar{f}\|_\infty = 1 - k_1 l^{k_1-1} - l^{k_1}(1-k_1), \tag{3.21}$$

where $\|\cdot\|_\infty$ is the supremum norm.

Proof. The difference $\delta(t) := f(t) - \bar{f}(t)$ is non-negative, continuous and equal to zero for $t \in [\,0, l\,] \cup [\,r, \infty\,)$. Therefore, we consider only the intervals $[l, 1]$ and $[1, r]$. On the interval $t \in [l, 1]$ the difference $\delta(t)$ is increasing, since

$$\begin{aligned} \delta(t) = f(t) - \bar{f}(t) &= t^{k_1} - a_1(t-l) - a_0 \\ &= t^{k_1} - t k_1 l^{k_1-1} - l^{k_1}(1-k_1). \end{aligned}$$

Hence, the maximum is attained at the boundary:

$$\max_{t\in[l,1]} \delta(t) = \delta(1) = 1 - k_1 l^{k_1-1} - l^{k_1}(1-k_1), \tag{3.22}$$

Analogously, the difference $\delta(t)$ is decreasing for $t \in [1, r]$. Consequently, the distance $\|f - \bar{f}\|_\infty = \delta(1)$ and that proves the statement. □

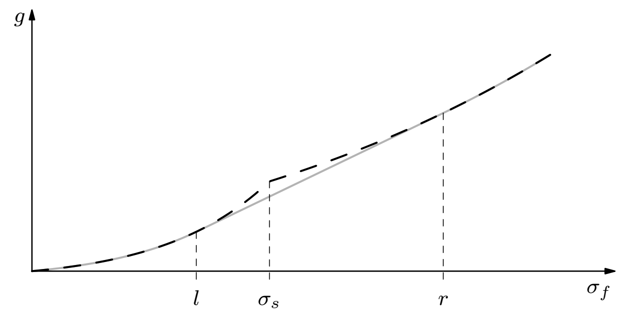


Figure 3.2: Original $f(t)$ (dashed) and convexified $\bar{f}(t)$ (gray)

Fig. 3.2 shows the original function $f(t)$ and its approximated version $\bar{f}(t)$. Now we can apply the results above to convexify the damage curve $d(\sigma_f)$.

Theorem 3.1.11. *The damage curve $g(\sigma_f)$ with $k_1 > k_2$ can be convexified by the following continuously differentiable function:*

$$\bar{g}(\sigma_f) := \frac{1}{N_s}\bar{f}(\sigma_f/\sigma_s), \tag{3.23}$$

where $\bar{f}$ is defined by Eq. (3.10) and $\|g - \bar{g}\|_\infty = N_s^{-1}\|f - \bar{f}\|_\infty$.

Proof. The result follows immediately from the Lemma 3.1.8 and Corollary 3.1.10, if we notice that the damage curve $g(\sigma_f)$ is nothing else but a rescaled function $f(t)$, so that $g(\sigma_f) = {N_s}^{-1}f(\sigma_f/\sigma_s)$. $\square$

We proved convexity of the damage function, reformulating the Goodman mean stress correction as a pointwise maximum of linear functions and convexifying the damage curve. Next, we are going to apply this important result to construct a DC decomposition of the model (3.1)–(3.3), thus, simplifying the global optimum search.

3.1.2 DC decomposition

Most of the global optimization problems are NP-hard, so for all methods there will always be some instances, which cannot be solved effectively. This in turn implies that we have to analyze properties of real-world problems and invent or select tailor-made algorithms to minimize the computational time. Sometimes it is possible to show that the problem belongs to an existing class of programs, whose solution methods are well-known.

DC (difference convex) programming constitutes an important part of non-convex optimization, since it can be not only theoretically studied, but more importantly widely applied. We define DC functions formally, based on [18], and then convert the model (3.1)–(3.3) to the DC program.

Definition 3.1.12. Let C be a convex subset of $\mathbb{R}^n$. A real-valued function $f : C \mapsto \mathbb{R}^n$ is called DC on C, if there exist two convex functions ψ, $\phi : C \mapsto \mathbb{R}^n$ such that f can be expressed in the form

$$f(\boldsymbol{x}) = \psi(\boldsymbol{x}) - \phi(\boldsymbol{x}). \tag{3.24}$$

Each representation of the form (3.24) is called a DC decomposition of f. An optimization problem with DC target functional and DC constraints is called a DC program.

Lemma 3.1.13. *Suppose $d_k(\boldsymbol{x})$ are convex and non-negative for $k = 1, \dots, K$ and the parameters r_k are positive, then the functional F, defined by*

$$F = \sum_{k=1}^{K} (d_k(\boldsymbol{x}) - r_k)^2 \tag{3.25}$$

has the following DC decomposition:

$$F = \psi(\boldsymbol{x}) - \phi(\boldsymbol{x}) \tag{3.26}$$

$$\psi(\boldsymbol{x}) = \sum_{k=1}^{K} \left(d_k^2(\boldsymbol{x}) - r_k^2\right) \tag{3.27}$$

$$\phi(\boldsymbol{x}) = 2\sum_{k=1}^{K} d_k(\boldsymbol{x}) r_k \tag{3.28}$$

where the functions ψ and ϕ are convex.

Proof. Obviously, $F(\boldsymbol{x}) \equiv \psi(\boldsymbol{x}) - \phi(\boldsymbol{x})$. Functions $d_k^2(\boldsymbol{x})$ are convex by Lemma 3.1.5, since they are compositions of the convex and non-decreasing function y^2 for $y \geq 0$ and convex function $d_k(\boldsymbol{x})$. Hence, $\psi(\boldsymbol{x})$ and $\phi(\boldsymbol{x})$ are nonnegative sums of convex functions. Therefore, they are convex by Lemma 3.1.4. □

Corollary 3.1.14. *If the damage curve g is convex and non-decreasing, then the test rig problem (3.1)–(3.3) can be converted to a DC program, choosing a squared distance as a quality criterion:*

$$\min_{\bar{\boldsymbol{p}}} \sum_{k=1}^{K} \left(d_k(\bar{\boldsymbol{p}}) - d_k^{ref}\right)^2 \tag{3.29}$$

$$d_k(\bar{\boldsymbol{p}}) = \sum_{b=1}^{B} n_b g\left[h(\sigma_{1,kb}(\bar{\boldsymbol{p}}), \sigma_{2,kb}(\bar{\boldsymbol{p}}))\right], \tag{3.30}$$

$$\sigma_{j,kb}(\bar{\boldsymbol{p}}) = \boldsymbol{p}_{j,b}^T \boldsymbol{s}_k \quad \textit{for} \quad j = 1,\, 2, \tag{3.31}$$

Proof. According to Theorem 3.1.7, the damage functions $d_k(\bar{\boldsymbol{p}})$ are convex for $k = 1, \dots, K$. Therefore, the target functional in Eq. (3.29) has a DC decomposition by Lemma 3.1.13. □

Corollary 3.1.14 allows the use of DC global optimality conditions and faster algorithms [25, 33] for the test rig problem with the discretized set of feasible alignment parameters.

3.1.3 Fully reversed constant loading

In this section we study a special case of the test rig problem (3.1)–(3.3), when the loading is constant and fully reversed, i.e. the number of blocks in the signal $B = 1$ and $\boldsymbol{p}_1 = -\boldsymbol{p}_2$ and, hence, $h(\sigma_{1,k}, \sigma_{2,k}) = h(-\boldsymbol{p}_2^T \boldsymbol{s}_k, \boldsymbol{p}_2^T \boldsymbol{s}_k) = |\boldsymbol{p}_2^T \boldsymbol{s}_k|$. Therefore, the original problem reduces to:

$$\min_{\boldsymbol{p}} \sum_{k=1}^{K} \left(\frac{ng(|\boldsymbol{p}^T \boldsymbol{s}_k|)}{d_k^{\text{ref}}} + \frac{d_k^{\text{ref}}}{ng(|\boldsymbol{p}^T \boldsymbol{s}_k|)} \right) \tag{3.32}$$

Next, we consider convexity properties of the simplified test rig problem (3.32) and represent it as a union of convex optimization programs.

Lemma 3.1.15. *If the damage curve $g(x)$ and its reciprocal $1/g(x)$ are convex for $x > 0$, then the problem (3.32) is locally convex everywhere except the hyperplanes $\boldsymbol{x}^T \boldsymbol{s}_k = 0$, i.e. when $\boldsymbol{p} \in \{\boldsymbol{p} \in \mathbb{R}^n \,|\, \boldsymbol{p}^T \boldsymbol{s}_k \neq 0,\ k = 1, \dots, K\}$.*

Proof. Convexity of the damage curve and its reciprocal yields convexity of their nonnegative weighted sum $f_k(x)$ by Lemma 3.1.4:

$$f_k(x) = \frac{ng(x)}{d_k^{\text{ref}}} + \frac{d_k^{\text{ref}}}{ng(x)}, \tag{3.33}$$

since the constants n and d_k^{ref} are strictly positive. The composition of $f_k(x)$ with the linear function $\boldsymbol{p}^T \boldsymbol{s}_k$ or $-\boldsymbol{p}^T \boldsymbol{s}_k$ is locally convex for $\boldsymbol{p}$ such that $\boldsymbol{p}^T \boldsymbol{s}_k \neq 0$, according to Lemma 3.1.3. The target functional in Eq. (3.32) is the sum of locally convex functions $f_k(|\boldsymbol{p}^T \boldsymbol{s}_k|)$ and that proves the statement. □

Theorem 3.1.16. *If we use the convexified damage curve $\bar{g}$, defined in Theorem 3.1.11, then the problem (3.32) is locally convex everywhere except the hyperplanes $\boldsymbol{x}^T \boldsymbol{s}_k = 0$, i.e. when $\boldsymbol{p} \in \{\boldsymbol{p} \in \mathbb{R}^n \,|\, \boldsymbol{p}^T \boldsymbol{s}_k \neq 0,\ k = 1, \dots, K\}$.*

Proof. Convexity of $\bar{g}(\sigma_f)$, combined with the convexity of its reciprocal $1/\bar{g}(\sigma_f)$ by Lemma 3.1.9, yields the local convexity of the target function by Lemma 3.1.15. □

Corollary 3.1.17. *If we use the convexified damage curve $\bar{g}$, defined in Theorem 3.1.11, then the problem (3.32) is a union of convex optimization programs on polyhedral cones, generated by the intersection of K hyperplanes $\boldsymbol{p}^T\boldsymbol{s}_k = 0$, passing through the origin.*

Proof. The target functional tends to infinity, when $\boldsymbol{p}^T\boldsymbol{s}_k \to 0$ for any $k \in \{1,\dots,K\}$, since $\bar{g}(0) = \infty$. Hence, the domain of the problem is split by the hyperplanes $\boldsymbol{p}^T\boldsymbol{s}_k = 0$ into conic subdomains, where the target functional is convex, according to the Theorem 3.1.16. Thus, the global optimization problem can be considered as a union of local problems. □

We showed that the global optimization problem (3.32) is a union of local problems on cones. Actual numerical solution of the problem needs polyhedral description of each cone, i.e. linear constraints that support it, and a good starting point inside. The next section is devoted to the derivation of the algorithm, that can detect all possible cones and starting points

3.2 Cone Generation

There is a problem that arises in global optimization problems when the target functional is finite and convex in all regions except hyperplanes sharing a common point. This implies that the target function is convex in all polyhedral cones defined by the intersection of half-spaces. Best practices include converting a non-convex problem into a union of convex ones, which can be easily solved by almost any local optimization method if a starting point and respective constraints are provided.

This section presents a fast algorithm that determines all proper polyhedral cones in $\mathbb{R}^N$ generated by intersections of M hyperplanes, passing through the origin. In addition, the respective interior points are determined.

Definition 3.2.1. Let $a_i \in \mathbb{R}^N$ for $i = 1,\dots,M$. A collection $\mathcal{M}(A)$ generated by matrix $A = [\boldsymbol{a}_1,\dots,\boldsymbol{a}_M]^T$ is defined as

$$\mathcal{M}(A) = \left\{B \mid B = \operatorname{diag}(\boldsymbol{s})A,\ \boldsymbol{s} = (\pm 1,\dots,\pm 1) \in \mathbb{R}^M\right\},$$

where the function $\operatorname{diag}(\boldsymbol{s}) : \mathbb{R}^M \mapsto \mathbb{R}^{M\times M}$ maps the vector $\boldsymbol{s}$ to the corresponding diagonal matrix so that $[\operatorname{diag}(\boldsymbol{s})]_{ii} = s_i$.

Definition 3.2.2. A polyhedral cone generated by matrix $A \in \mathbb{R}^{M\times N}$ is given by

$$\operatorname{cone}(A) = \{\boldsymbol{x} \in \mathbb{R}^N \mid A\boldsymbol{x} \geq \boldsymbol{0}\}$$

Definition 3.2.3. cone(A) is called proper, if $\operatorname{int}\operatorname{cone}(A) \neq \emptyset$.

Remark 3.2.4. Obviously, $|\mathcal{M}(A)| = 2^M$, but not all elements of $\mathcal{M}(A)$ generate proper cones. That motivates to define the following collection.

Definition 3.2.5. A collection $\mathcal{M}_p(A)$ generated by matrix $A \in \mathbb{R}^{M\times N}$ is defined as

$$\mathcal{M}_p(A) = \{B \mid B \in \mathcal{M}(A), \text{ s.t. } \operatorname{cone}(B) \text{ is proper}\}$$

Definition 3.2.6. A cone-point pair collection $\Upsilon(A)$ generated by matrix $A \in \mathbb{R}^{M\times N}$ is defined as follows:

$$\Upsilon(A) = \{(A_k, \boldsymbol{x}_k) \mid A_k \in \mathcal{M}_p(A),\ \boldsymbol{x}_k \in \operatorname{int}\operatorname{cone}(A_k),\ A_{k_1} \neq A_{k_2} \text{ for } k_1 \neq k_2\}$$

Remark 3.2.7. A cone-point pair collection $\Upsilon(A)$ is symmetric, i.e. if $(B, \boldsymbol{x}) \in \Upsilon(A)$, then $(-B, -\boldsymbol{x}) \in \Upsilon(A)$.

The objective is to develop an algorithm that locates a cone-point pair collection $\Upsilon(A)$ generated by the given matrix A. This problem is equivalent to finding all proper cones, generated by intersections of planes $\boldsymbol{a}_i^T\boldsymbol{x} = 0$ for $i = 1, \ldots, M$.

Assumption 3.2.8. *Matrix A has full rank. Vectors $\boldsymbol{a}_j$ define different planes, i.e.:*

$$\frac{\boldsymbol{a}_{j_1}}{\|\boldsymbol{a}_{j_1}\|} \neq \frac{\pm\boldsymbol{a}_{j_2}}{\|\boldsymbol{a}_{j_2}\|} \quad \textit{for} \quad j_1 \neq j_2$$

We shall consider three different cases, that might appear in this problem.

3.2.1 Case $M = N$

Theorem 3.2.9. *If $A \in \mathbb{R}^{N\times N}$, then under Assumption 3.2.8 a cone-point pair collection $\Upsilon(A)$ is defined by the following formula:*

$$\Upsilon(A) = \left\{(\operatorname{diag}(\boldsymbol{s})A,\ A^{-1}\boldsymbol{s}) \mid \boldsymbol{s} = (\pm1, \ldots, \pm1) \in \mathbb{R}^N\right\}$$

Proof. Assumption 3.2.8 guarantees, that in this case $\mathcal{M}_p(A) = \mathcal{M}(A)$. Let us consider a matrix $B(\boldsymbol{s}) = \operatorname{diag}(\boldsymbol{s})A$ and a point $\boldsymbol{x}(\boldsymbol{s}) = A^{-1}\boldsymbol{s}$ with $s_i = \pm1$ for $i = 1, \ldots, M$, then the following is true:

$$B(\boldsymbol{s})\boldsymbol{x}(\boldsymbol{s}) = \operatorname{diag}(\boldsymbol{s})AA^{-1}\boldsymbol{s} = \mathbf{1} > 0 \tag{3.34}$$

The inequality (3.34) implies that $\operatorname{cone}(B(\boldsymbol{s}))$ is proper, since $x(\boldsymbol{s})$ is an interior point of the cone. Thus, the cone-point pair $(B(\boldsymbol{s}), \boldsymbol{x}(\boldsymbol{s})) \in \Upsilon(A)$ and that proves the statement. □

Remark 3.2.10. The proof of Theorem 3.2.9 can also be explained geometrically. If we take polyhedral cone $C(\boldsymbol{s})$, generated by extreme rays $s_i\boldsymbol{e}_i$ for $i = 1, \ldots, N$:

$$C(\boldsymbol{s}) = \left\{\sum_{i=1}^{N} \lambda_i s_i \boldsymbol{e}_i \mid \lambda_i > 0 \text{ for } i = 1, \ldots, N\right\}$$

where the unit vectors $\boldsymbol{e}_1, \dots, \boldsymbol{e}_N$ form an orthonormal basis of $\mathbb{R}^N$ and $s_i = \pm 1$, then vector $\boldsymbol{s} \in \operatorname{int} C(\boldsymbol{s})$. The interior points $\boldsymbol{s} = (\pm 1, \dots, \pm 1)^T \in \mathbb{R}^N$ are vertices of a hypercube. If we perform an affine transformation of this cube, according to extreme rays $\boldsymbol{w}_i$ for $i = 1, \dots, N$, we obtain interior points of cones, generated by extreme rays $s_i \boldsymbol{w}_i$. Fig. 3.3 shows example of such transformation in $\mathbb{R}^2$.

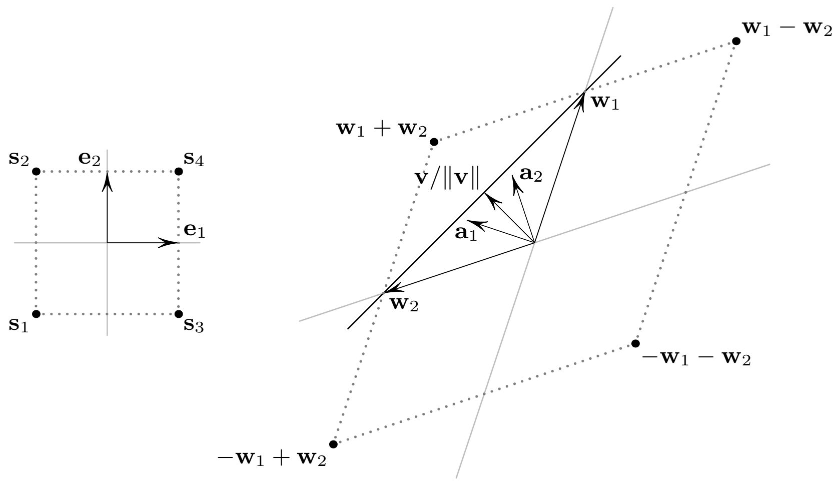

$\mathbf{s}_2$
$\mathbf{e}_2$
$\mathbf{s}_4$
$\mathbf{e}_1$
$\mathbf{s}_1$
$\mathbf{s}_3$
$\mathbf{w}_1 - \mathbf{w}_2$
$\mathbf{w}_1$
$\mathbf{w}_1 + \mathbf{w}_2$
$\mathbf{v}/\|\mathbf{v}\|$
$\mathbf{a}_2$
$\mathbf{a}_1$
$\mathbf{w}_2$
$-\mathbf{w}_1 - \mathbf{w}_2$
$-\mathbf{w}_1 + \mathbf{w}_2$

Figure 3.3: Illustration of Theorem 3.2.9 in $\mathbb{R}^2$

Let us consider a matrix $B = [\boldsymbol{b}_1, \dots, \boldsymbol{b}_M]^T \in \mathcal{M}(A)$, s.t. $B = \operatorname{diag}(\boldsymbol{s})A$. The extreme rays $\boldsymbol{w}_i$ are found by calculating the intersections of planes $\boldsymbol{b}_i^T \boldsymbol{x} = 0$ for $i = 1, \dots, N$ with a plane $\boldsymbol{v}^T \boldsymbol{x} = 1$, where $\boldsymbol{v}^T \boldsymbol{b}_i \neq 0$. So we have to solve the following linear equation:

$$Q_i \boldsymbol{w}_i = \boldsymbol{e}_i, \tag{3.35}$$

where $Q_i = [\boldsymbol{b_1}, \dots, \boldsymbol{b_{i-1}}, \boldsymbol{v}, \boldsymbol{b_{i+1}}, \dots, \boldsymbol{b_N}]^T$. For example define a vector $\boldsymbol{v} \in \mathbb{R}^N$ as follows:

$$\boldsymbol{v} := \sum_{i=1}^{N} \boldsymbol{b}_i = \sum_{i=1}^{N} s_i \boldsymbol{a}_i \tag{3.36}$$

Matrix Q_i in Eq. (3.35) can be decomposed to $Q_i = P_i A$, where the matrix P_i is defined as:

$$P_i = \begin{bmatrix} s_1 & & & & \\ & \ddots & & & \\ s_1 & \cdots & s_i & \cdots & s_N \\ & & & \ddots & \\ & & & & s_N \end{bmatrix} \tag{3.37}$$

Substitute (3.37) in (3.35):

$$\boldsymbol{w}_i = Q_i^{-1} \boldsymbol{e}_i = (P_i A)^{-1} \boldsymbol{e_i} = A^{-1} s_i \, \boldsymbol{e_i} \tag{3.38}$$

The internal point $\boldsymbol{x}$ can be calculated as the sum of extreme rays $\boldsymbol{w}_i$ using the expression (3.38):

$$\boldsymbol{x} = \sum_{i=1}^{N} \boldsymbol{w_i} = A^{-1} \sum_{i=1}^{N} s_i \boldsymbol{e_i} = A^{-1}\boldsymbol{s} \tag{3.39}$$

Hence, we arrived to the same cone-point pair $(B, \boldsymbol{x})$ as in Theorem 3.2.9.

Algorithm and complexity

Since matrix inversion requires at most $O(N^3)$ time and each of 2^{N-1} executions of the **for** loop has two matrix-vector multiplications $O(2N^2)$, in total the Algorithm 1 will require $O(2^N N^2 + N^3)$ time [14].

Algorithm 1: Y = GenerateConePointPairsNN(A)

```
**Input**: $A \in \mathbb{R}^{N\times N}$ under Assumption 3.2.8
**Output**: $Y$, s.t. $Y \cup -Y = \Upsilon(A)$
$A_{\text{inv}} = A^{-1}$
$Y = \emptyset$
**for** $i = 1$ **to** $2^{N-1}$ **do**
    construct new vector $\boldsymbol{s} = (\pm1, \dots, \pm1, 1)$
    append $(\operatorname{diag}(\boldsymbol{s})A, A_{\text{inv}}\,\boldsymbol{s})$ to $Y$
```

Remark 3.2.11. Note, that the Algorithm 1 generates only half of the the elements of the cone-point pair collection $\Upsilon(A)$. According to Corollary 3.2.7 the second half of the elements are point reflections of the cone-point pairs we generated.

3.2.2 Case $M < N$

Theorem 3.2.12. *If $A \in \mathbb{R}^{M\times N}$ and $M < N$, then under Assumption 3.2.8 a cone-point pair collection $\Upsilon(A)$ is given by*

$$\Upsilon(A) = \left\{(\operatorname{diag}(\boldsymbol{s})A,\ U(AU)^{-1}\boldsymbol{s}) \mid \boldsymbol{s} = (\pm1, \dots, \pm1) \in \mathbb{R}^M\right\},$$

where the columns of matrix $U \in \mathbb{R}^{N\times M}$ form an orthonormal basis of the row space of A.

Proof. Let $\boldsymbol{u}_1, \dots, \boldsymbol{u}_M$ be an orthonormal basis of the M-dimensional subspace, spanned by vectors $\boldsymbol{a}_1, \dots, \boldsymbol{a}_M$.

Define a matrix $U = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_M] \in R^{N\times M}$ and a matrix $D = AU$. Rows of D are coordinates of the respective rows of A in the basis U. Based on Theorem 3.2.9 we get a cone-point pair collection for the matrix D:

$$\Upsilon(D) = \left\{(\operatorname{diag}(\boldsymbol{s})D,\ D^{-1}\boldsymbol{s}) \mid \boldsymbol{s} = (\pm1, \dots, \pm1) \in \mathbb{R}^M\right\} \tag{3.40}$$

In the original basis the cone-point pair collection (3.40) has the following form:

$$\Upsilon(A) = \left\{ (\mathrm{diag}(\boldsymbol{s})A,\ UD^{-1}\boldsymbol{s}) \mid \boldsymbol{s} = (\pm 1, \ldots, \pm 1) \in \mathbb{R}^M \right\} \tag{3.41}$$

Eq. (3.41) shows, how to calculate all cone-point pairs, generated by matrix $A \in \mathbb{R}^{M \times N}$ with $M < N$, using its row space. □

Algorithm and complexity

In order to find a basis of the row space of matrix A, one can use Gram-Schmidt orthogonalization process, but it has poor numerical properties, which was illustrated in [14]. In our implementation we use Householder QR-decomposition, which requires $O(N^2M - NM^2 + M^3/3)$ flops [14]. We have to multiply an $N \times M$ matrix by the inverse of an $M \times M$ matrix, calculating matrix T in the Algorithm 2, which takes $O(M^3 + M^2N)$ operations. Since every execution of the **for** loop has two matrix-vector multiplications $O(2MN)$, in total the Algorithm 1 requires $O(2^M MN + MN^2 + 4M^3/3)$ operations.

Algorithm 2: Y = GenerateConePointPairsSmallM(A)

```
Input: $A \in \mathbb{R}^{M \times N}$, where $M < N$, under Assumption 3.2.8
Output: $Y$, s.t. $Y \cup -Y = \Upsilon(A)$
// orthonormal matrix $Q = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_N] \in R^{N \times N}$
$Q$ = HouseholderQRDecomposition($A^T$)            // $Q \in \mathbb{R}^{N \times N}$
// columns of $U$ form basis of the row space of $A$
$U = [\boldsymbol{q}_1, \ldots, \boldsymbol{q}_M]$                         // $U \in \mathbb{R}^{N \times M}$
$T = U(AU)^{-1}$                                // $T \in \mathbb{R}^{N \times M}$
$Y = \emptyset$
for $i = 1$ to $2^{M-1}$ do
    construct new vector $\boldsymbol{s} = (\pm 1, \ldots, \pm 1, 1)$
    append $(\mathrm{diag}(\boldsymbol{s})A, T\,\boldsymbol{s})$ to $Y$
```

3.2.3 Case $M > N$

Even in $\mathbb{R}^3$ the explicit formula for the number of cones, not to mention a formula for the interior points, is not that obvious. For example we have three orthogonal planes in Fig. 3.4 and different outcomes after adding one more plane. We get either 12 cones as in Fig. 3.4a or 14 cones as in Fig. 3.4b.

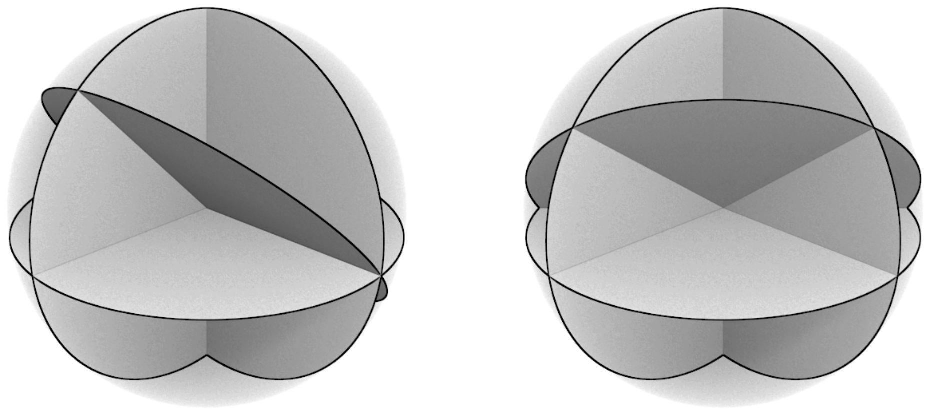

(a) 12 cones (b) 14 cones

Figure 3.4: Cones, generated by intersection of 4 planes

Ho and Zimmerman [17] studied the number of regions, cut by a set of hyperplanes in general position, which gives us an upper bound for the number of proper cones and could be used to analyze the worst-case complexity.

Definition 3.2.13. A set $\mathcal{H}$ of M hyperplanes in $\mathbb{R}^N$ is said to be in general position if for each k, $0 \leq k \leq N$, no $k+1$ members of $\mathcal{H}$ contain a common $(N-k)$-dimensional affine subset of $\mathbb{R}^N$.

If the hyperplanes are in general position, then the space is split into as many nonempty regions as possible.

Theorem 3.2.14 (Ho, Zimmerman)**.** *The number of regions $C_N(M)$, cut by M hyperplanes in $\mathbb{R}^N$ in general position all contain a common point, equals to:*

$$C_N(M) = 2\sum_{k=0}^{N-1}\binom{M-1}{k} \tag{3.42}$$

Remark 3.2.15. A set of hyperplanes $\boldsymbol{a}_1^T\boldsymbol{x} = 0, \ldots, \boldsymbol{a}_M^T\boldsymbol{x} = 0$ does not have to be in general position, which implies $|\Upsilon(A)| \leq C_N(M)$.

We are interested not only in the number of proper cones, but in their polyhedral description and interior points. In order to generate all cone-point pairs $\Upsilon(A)$ we can solve 2^{M-1} linear feasibility problems of type:

$$\text{find } \boldsymbol{x}, \quad \text{s.t.} \quad [\pm\boldsymbol{a}_1, \ldots, \pm\boldsymbol{a}_{M-1}, \boldsymbol{a}_M]^T\boldsymbol{x} > 0, \tag{3.43}$$

but this approach has exponential running time complexity with respect to the number of hyperplanes M. Also, most of the problems, defined in (3.43) are infeasible according to Theorem 3.2.14. We call this approach naive throughout the rest of this paper. A better approach would use Theorem 3.2.9 to pregenerate first 2^{N-1} cone-point pairs and then take each hyperplane $\boldsymbol{a}_i^T\boldsymbol{x} = 0$ for $i = N+1, \dots, M$ and check, if it splits any of pregenerated cones. We describe this algorithm and study its complexity theoretically and experimentally in the following sections.

Algorithm

The idea of the algorithm is to utilize Theorem 3.2.9 and Algorithm 1 to generate first 2^{N-1} cone-point pairs $\Upsilon([\boldsymbol{a}_1, \dots, \boldsymbol{a}_N]^T)$ and generate pairs inductively for each new *arriving* plane $\boldsymbol{a}_i$ inductively for $i = N+1, \dots, M$. Every pair $(B, \boldsymbol{x}_0) \in Y$ serves as a root of a binary tree. For a plane normal $\boldsymbol{p}$ such that $\boldsymbol{p}^T\boldsymbol{x}_0 \geq 0$, we check, whether cone$([B; -p^T])$ is proper by solving a corresponding linear feasibility problem, looking for a vector $\boldsymbol{x}_{\text{new}} \in \operatorname{int}\operatorname{cone}(B)$ such that $\boldsymbol{p}^T\boldsymbol{x}_{\text{new}} < 0$. If it is proper and the node is a leave, then we add two children $([B; \boldsymbol{p}^T], \boldsymbol{x}_0)$ and $([B; -\boldsymbol{p}^T], \boldsymbol{x}_{\text{new}})$. If it is proper, but the node has children, then we try to split each of them.

Algorithm 3: Y = GenerateConePointPairsBigM(A)

```
Input: $A \in \mathbb{R}^{M \times N}$, where $M > N$, under Assumption 3.2.8
Output: $Y$, s.t. $Y \cup -Y = \Upsilon(A)$
// define binary tree node
NODE $\equiv$ {cone-point pair $(B, \boldsymbol{x})$, NODE left, NODE right}
// generate $2^{N-1}$ first cones
$Y$ = GenerateConePointPairsNN($[\boldsymbol{a}_1, \dots, \boldsymbol{a}_N]^T$)
// matrix to store inequality description of cones
$P = \emptyset$
foreach $(B, \boldsymbol{x})$ in $Y$ do
    // initialize the tree
    T = NODE($(B, \boldsymbol{x})$, NULL, NULL)
    remove $(B, \boldsymbol{x})$ from $Y$
    $P = \emptyset$
    // split the cone $B\boldsymbol{x} \geq 0$ with planes $\boldsymbol{a}_{N+1}, \dots, a_M$
    for $i = N+1$ to $M$ do
        SplitCone($T$, $\boldsymbol{a}_i$, $P$)          // see Algorithm 4
    append generated pairs of $T$ to $Y$
```

Remark 3.2.16. To reduce storage complexity in Algorithm 3 we store matrix B only in the root node, all other nodes store only added vectors. To store inequality representation of the cones we use matrix P.

Algorithm 4: SplitCone(NODE, $\boldsymbol{p}$, P)

```
Input: NODE - node of a binary tree, which stores pair $(B, \boldsymbol{x})$
       $\boldsymbol{p} \in \mathbb{R}^N$ - plane candidate to split cone($B$)
       $P$ - matrix of the current cone
// define $\boldsymbol{p}$ and $\boldsymbol{p}_{\mathtt{new}}$ so, that $\boldsymbol{p}^T\boldsymbol{x} \geq 0$ and $\boldsymbol{p}_{\mathtt{new}}^T\boldsymbol{x} \leq 0$
if $\boldsymbol{p}^T\boldsymbol{x} \geq 0$ then
    $\boldsymbol{p}_{\text{new}} = -\boldsymbol{p}$
else
    $\boldsymbol{p}_{\text{new}} = \boldsymbol{p}$
    $\boldsymbol{p} = -\boldsymbol{p}$
// check if the cone($[P; B; p_{\mathtt{new}}^T]$) is proper
is_proper, $\boldsymbol{x}_{\text{new}}$ = SolveLinearFeasibility($[P$; NODE.$B$; $p_{\text{new}}^T]$)
if $is\_proper$ then
    if $NODE.left == NULL$ then
        NODE.left = NODE(($\boldsymbol{p}^T, \boldsymbol{x}$), NULL, NULL)
        NODE.right = NODE(($\boldsymbol{p}_{\text{new}}^T, \boldsymbol{x}_{\text{new}}$), NULL, NULL)
    else
        SplitCone(NODE.left, $\boldsymbol{p}$, $[P; B]$)
        SplitCone(NODE.right, $\boldsymbol{p}$, $[P; B]$)
```

Example 3.2.17. We would like to illustrate the algorithm with a two-dimensional example. Assume, four plane normals $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_4 \in \mathbb{R}^2$ are given. The goal is to find every proper cone, supported by the planes.

Step 1. Consider first two planes $\boldsymbol{a}_1^T\boldsymbol{x} = 0$ and $\boldsymbol{a}_2^T\boldsymbol{x} = 0$, depicted in Fig. 3.5. If $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are linearly independent, we can use the Algorithm 1 to generate first two cone-point pairs $([\boldsymbol{a}_1, \boldsymbol{a}_2], \boldsymbol{x}_1)$ and $([\boldsymbol{a}_1, -\boldsymbol{a}_2], \boldsymbol{x}_2)$. The roots of two binary trees are initialized with the pairs.

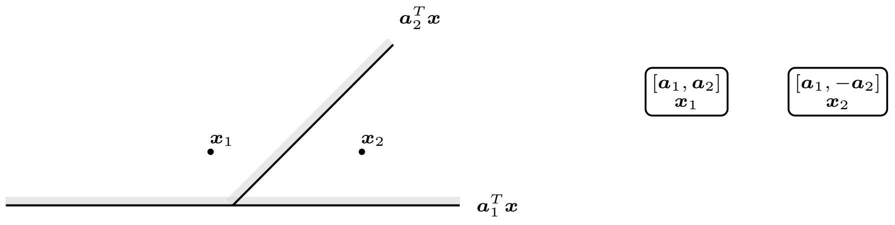

$\boldsymbol{a}_2^T\boldsymbol{x}$
$\boldsymbol{x}_1$
$\boldsymbol{x}_2$
$\boldsymbol{a}_1^T\boldsymbol{x}$
$[\boldsymbol{a}_1, \boldsymbol{a}_2]$ $\boldsymbol{x}_1$
$[\boldsymbol{a}_1, -\boldsymbol{a}_2]$ $\boldsymbol{x}_2$

Figure 3.5: Cone-point pairs, generated by $\boldsymbol{a}_1, \boldsymbol{a}_2$. Gray stripes indicate positive half-spaces.

Step 2. Fig. 3.6 shows the next plane $\boldsymbol{a}_3^T\boldsymbol{x} = 0$ and we want to test, whether it splits the cone($[\boldsymbol{a}_1, \boldsymbol{a}_2]^T$). Thus, we compute the product $\boldsymbol{a}_3^T\boldsymbol{x}_1$. Since it is positive, the cone is split if and only if the linear feasibility problem $[\boldsymbol{a}_1, \boldsymbol{a}_2, -\boldsymbol{a}_3]^T\boldsymbol{x} > 0$ has a solution. If it was negative, we would search a vector $\hat{\boldsymbol{x}}$, such that $[\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3]^T\hat{\boldsymbol{x}} > 0$.

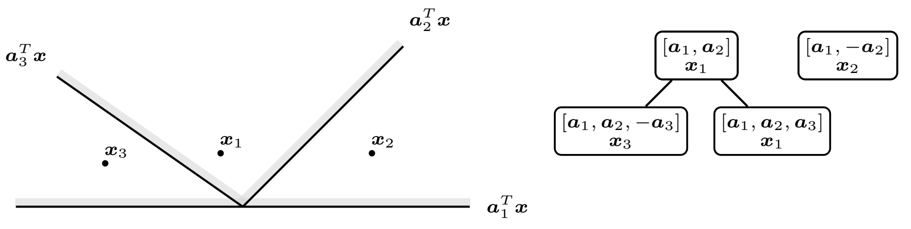


Figure 3.6: Cone-point pairs, generated by $\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3$. Gray stripes indicate positive half-spaces.

The linear feasibility solver returns $\boldsymbol{x}_3 \in \operatorname{int} \operatorname{cone}([\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3]^T)$. Since the root is a leave, i.e. does not have children, we append two nodes with the cone-point pairs $([\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3], \boldsymbol{x}_1)$ and $([\boldsymbol{a}_1, \boldsymbol{a}_2, -a_3], \boldsymbol{x}_3)$ to the first tree. Analogously, consider the second tree. The sign of $\boldsymbol{a}_3 \boldsymbol{x}_2$ is also positive, but the solver returns the certificate of infeasibility for the problem $[\boldsymbol{a}_1, -\boldsymbol{a}_2, -\boldsymbol{a}_3]\, \boldsymbol{x} > 0$, so the cone is not split by the plane $\boldsymbol{a}_3^T \boldsymbol{x} = 0$. We proceed to the next plane.

Step 3. Consider the plane $\boldsymbol{a}_4 \boldsymbol{x} = 0$, shown in Fig. 3.7. Following the same procedure as in Step 2, we determine that it splits the cone, generated by $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$. Recursively, we repeat the procedure for cones in the left and right node till we get to the leaves.

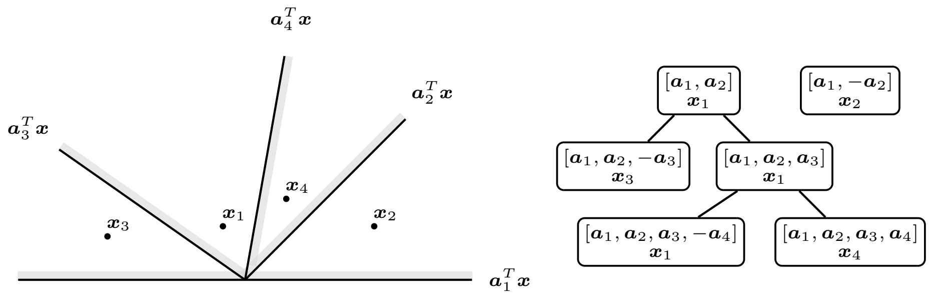


Figure 3.7: Cone-point pairs, generated by $\boldsymbol{a}_1, \boldsymbol{a}_2, \boldsymbol{a}_3, \boldsymbol{a}_4$. Gray stripes indicate positive half-spaces.

As one can observe in Fig. 3.7, the union of the leaves of both trees contains the cone-point pairs, generated by the plane normals $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_4$.

3.2.4 Linear Feasibility Problem

The main ingredient of the procedure SplitCone, listed in Algorithm 4, is the function SolveLinearFeasibility(A), that computes interior point $\boldsymbol{x}_{\text{new}}$ of cone(A) or proves, that cone(A) is not proper.

There are several well-known approaches to find interior points of cone(A). The simplex method of Dantzig has exponential worst-case complexity [24], relaxation methods of Agmon [1], Motzkin and Shoenberg [30] do not detect infeasibility, ellipsoid method of Khachiyan [23] has polynomial time complexity, but it is too slow and difficult to implement compared to the interior point method of Karmarkar [22].

Recently Chubanov [8] presented a promising projection algorithm that considers linear system with non-negativity constraints:

$$D\boldsymbol{y} = \mathbf{0}, \quad y \geq \mathbf{0}, \tag{3.44}$$

where the matrix $D \in \mathbb{R}^{m\times n}$. The algorithm finds a positive feasible solution $\boldsymbol{y} > 0$ or proves that there are no such solutions. Chubanov proved that running time of the algorithm is bounded by $O(n^4 + n^3 L_{\min})$, where the value $L_{\min}$ is bounded by a polynomial in the size of A.

Roos [37] proposed an improvement of Chubanov's method. In this paper we will use the Chubanov-Roos method, but any polynomial time method that detects infeasibility can be plugged in our algorithm.

Problem 3.2.18. The problem of finding an interior point of a cone(A) can be rewritten as a system (3.44) to fit into Chubanov's method to find a strictly positive solution of the system, such that

$$[A, -A, -I]\,\boldsymbol{y} = \mathbf{0}, \tag{3.45}$$

where $\boldsymbol{y} := [\boldsymbol{x}^+, \boldsymbol{x}^-, \boldsymbol{z}] \in (0,\infty)^{2N+M}$. Then vector $\boldsymbol{x}^+ - \boldsymbol{x}^-$ lies in the interior of cone(A).

3.2.5 Complexity

Previously we have fully described the algorithm to generate cone-point pairs inductively. In this section we will provide a polynomial bound on the number of calls of the function SolveLinearFeasibility(A), which is the most time-consuming part of the Algorithm 3.

Lemma 3.2.19. *Let set of $\mathcal{H}$ of $M-1$ hyperplanes in $\mathbb{R}^N$ be in general position all contain $\mathbf{0}$. Then the maximal number of regions $A_N(M)$ added by one more hyperplane, containing zero, equals to:*

$$A_N(M) = 2\sum_{k=0}^{N-2}\binom{M-2}{k} \tag{3.46}$$

Proof. The statement of the Lemma 3.2.19 follows directly from the Theorem 3.2.14:

$$\begin{aligned} A_N(M) = C_N(M) - C_N(M-1) &= 2\sum_{k=0}^{N-1}\binom{M-1}{k} - \binom{M-2}{k} \\ &= 2\sum_{k=0}^{N-2}\binom{M-2}{k} \end{aligned}$$

□

Theorem 3.2.20. *The number of calls to the linear feasibility problems solver in Algorithm 3 for an input $A \in \mathbb{R}^{M\times N}$ s.t. $M > N$ is bounded by $O((M-N)2^{N-1} + M^N)$.*

Proof. The Algorithm 3 generates 2^{N-1} cones for vectors $\boldsymbol{a}_1, \dots, \boldsymbol{a}_N$ as described in Section 3.2. Then it iteratively considers each row $\boldsymbol{a}_i$ for $i = N+1, \dots, M$ of the matrix A. Let $L = M - N$ denote the number of planes to consider and let K_i for $i = 1, \dots, L$ be the number of calls to the function SolveLinearFeasibility for the added vector $\boldsymbol{a}_{N+i}$ in the worst-case:

$$\begin{aligned}
K_1 &= 2^{N-1} \\
K_2 &= 2^{N-1} + A_N(N+1)/2 \\
&\dots \\
K_L &= 2^{N-1} + A_N(N+1)/2 + \dots + A_N(N+L-1)/2
\end{aligned}$$

Total number of calls K equals to:

$$K = \sum_{i=1}^{L} K_i = 2^{N-1}L + \frac{1}{2}\sum_{k=1}^{L-1} k\, A_N(N+L-k) \tag{3.47}$$

Let's consider the second term in equation (3.47):

$$\begin{aligned}
\frac{1}{2}\sum_{k=1}^{L-1} k\, A_N(N+L-k) &= \sum_{i=0}^{N-2}\sum_{k=1}^{L-1} k \binom{N+L-k-2}{i} \\
&= \sum_{i=0}^{N-2} \binom{N+L-1}{i+2} \\
&- \frac{N+2L-2+i(L-1)}{(i+2)} \binom{N-1}{i+1} \\
&\le \sum_{i=0}^{N-2} \binom{N+L-1}{i+2} \le (N+L)^N
\end{aligned} \tag{3.48}$$

We use the inequality (3.48) in the expression (3.47) to obtain the following bound:

$$K \le 2^{N-1}L + (N+L+2)^N = 2^{N-1}(M-N) + (M+2)^N \tag{3.49}$$

That proves the statement of the theorem. □

Remark 3.2.21. Theorem 3.2.20 implies that the number of solutions of linear feasibility problems 3.2.18 in the Algorithm 3 is polynomially bounded with respect to the number of planes M for fixed dimension of the space N. This is a good result, compared to the naive approach which has to solve 2^{M-1} linear feasibility problems to achieve the goal and hence grows exponentially when M increases.

3.2.6 Numerical Results

We have implemented Chubanov's projection algorithm [8] with Roos' improvement [37] in C++ for solving homogeneous linear feasibility problems. This solver is used by both the inductive Algorithm 3 and the naive approach, described in the Section 3.2.3. We launch each case for randomly generated matrices $A \in \mathbb{R}^{M \times N}$ where $M > N$. The elements of each matrix were uniformly randomly chosen in the interval $[-10, 10]$.

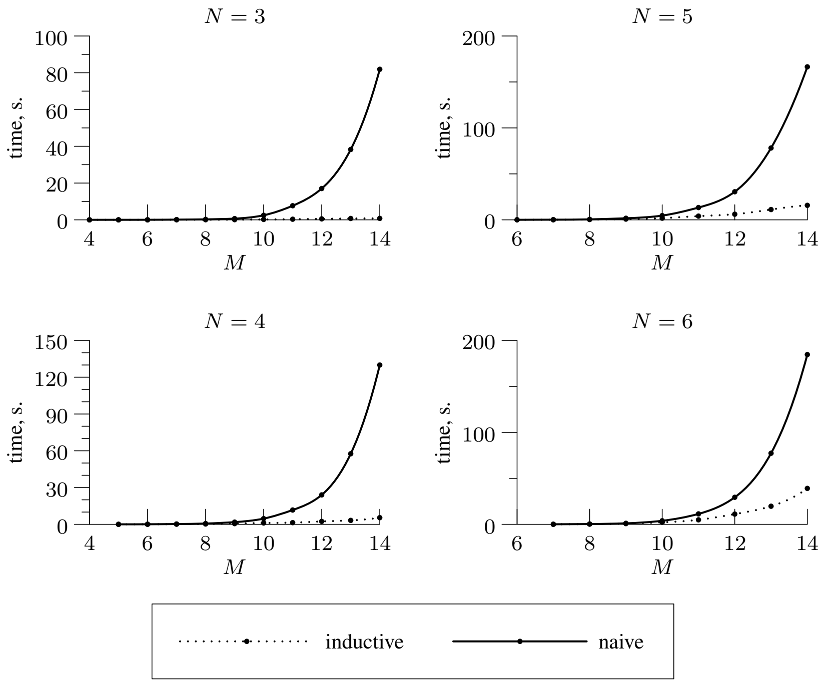


Figure 3.8: Performance comparison

In Figure 3.8 we show the average running time of the Algorithm 3 and the naive algorithm over 100 launches for randomly generated matrices. In Table 3.1 we collect running time of our algorithm and the factor which shows how much faster the presented algorithm is than the naive approach.

One can see that the inductive Algorithm 3 is substantially faster when the number of N-dimensional planes M is big enough which fully coincides with the results of the Theorem 3.2.20.

Table 3.1: Performance of Algorithm 3

	$N = 3$		$N = 4$		$N = 4$		$N = 6$	
M	time, s.	factor	time, s.	factor	time, s.	factor	time, s.	factor
4	0.0025	1.1	—	—	—	—	—	—
5	0.0089	1.4	0.0113	1.1	—	—	—	—
6	0.0121	2.5	0.0496	1.2	0.0234	1.2	—	—
7	0.0338	3.5	0.1311	1.9	0.1017	1.5	0.0456	1.3
8	0.0560	4.8	0.2853	2.2	0.3460	1.5	0.2425	1.3
9	0.0899	7.4	0.5647	3.2	0.8127	2.1	0.7896	1.4
10	0.2155	11.5	1.0894	4.3	1.8012	2.6	2.1884	1.8
11	0.3391	22.6	1.3633	8.6	4.0579	3.3	4.8817	2.3
12	0.5273	32.3	2.3754	10.0	6.1179	5.0	11.1282	2.6
13	0.7374	51.9	3.2025	18.0	11.1997	7.0	19.6711	3.9
14	0.7783	105.2	5.4289	23.0	15.7726	10.6	39.0665	4.7

We have derived explicit formulas for cone-point pairs when the number of planes M is less than or equal to the dimension of the space N. We used the Formula 3.2.9 for the $M = N$ case to generate first the 2^{N-1} cones when $M > N$. We, then, constructed an algorithm that splits these cones inductively. We showed that inductive generation of cone-point pairs is much more effective than the naive approach.

3.3 Conclusion

In this chapter we considered a special case of the test rig problem, when all of the critical plane angles are known, so that there is no need to optimize them. We proved that the damage function is convex, if we use the damage curve with $k_1 = k_2$ or the convexified the damage curve for $k_1 > k_2$. This result was used to find the DC decomposition of the target function.

Also, we proved that the problem with fully reversed constant loading is locally convex, if the damage curve $g(\sigma_f)$ and its reciprocal $1/g(\sigma_f)$ are convex. It was shown that in this case the non-convex test rig problem can be converted into a union of convex optimization problems on polyhedral cones, generated by hyperplanes, passing through origin. A method for defining representations and interior points of such cones was constructed. It includes analytical formulas for the cases, when the number of planes M is less or equal than the dimension N. For $M > N$, a recursive algorithm using a homogeneous linear feasibility solver was designed and proven to have a polynomial worst case complexity. Also, it had impressive running time, compared with the existing approach. This method provides starting points and respective constraints for all convexity regions of the problem. Hence, we can use fast local optimization solvers e.g. [41] to solve the global problem.

Chapter 4

Bilevel Optimization

In the previous chapter we studied a special case of the test rig problem (2.29)–(2.33), assuming that the critical planes are known and the alignment parameters are fixed. Although this can happen in practice, it is not valid for all applications. Thus, the properties of the general test rig problem have to be studied.

The problem mentioned above belongs to the class of bilevel programs, since every instance of the state vector, which includes alignment parameters and applied forces, defines a one-dimensional inner maximization problem at each hot spot.

Hence, the discussion is divided into two parts. At first the inner problem with and without fully reversed loading assumption is considered. In both cases the sum of functions $d(\alpha) = d_1(\alpha) + \ldots + d_B(\alpha)$ has to be maximized. The global maximum of the sum d cannot be found so easily, although the local maximum points can be computed analytically for every $d_i(\alpha)$. We propose a new branch-and-bound technique, which exploits the knowledge of the extrema of each summand d_i to find the global maximum effectively. Convergence of the algorithm is proven and its running time is tested and compared with the running time of DIRECT algorithm.

Next, the regularity of the outer program is studied. It is not convex in general, but we can discretize the set of alignment parameters of each actuator to gain convexity of the damage function, therefore, converting the problem to a DC program. Hence, the following assumption is made.

Assumption 4.0.1. *The alignment variables $\boldsymbol{\tau}_i$ are given and immutable for every actuator $i = 1, \ldots, A$.*

Also differentiability of the problem is shown and the gradient with respect to all state variables, including alignment parameters, is found. We arrive at a conclusion that we can solve the problem by derivative-based global optimization methods.

4.1 Model

The general test rig problem (2.28)–(2.33) approximates the reference damage values $d_k^{\mathrm{ref}} > 0$, generating a loading signal with B blocks. The state vector $\boldsymbol{z} = [\bar{\boldsymbol{p}}; \bar{\boldsymbol{\tau}}]$ includes applied forces $\bar{\boldsymbol{p}} = [\boldsymbol{p}_{1,1}, \boldsymbol{p}_{2,1}, \ldots, \boldsymbol{p}_{1,B}, \boldsymbol{p}_{2,B}] \in \mathbb{R}^{2AB}$ and alignment parameters $\bar{\boldsymbol{\tau}} = [\boldsymbol{\tau}_1, \ldots, \tau_A] \in \mathbb{R}^{AT}$ for actuators $i = 1, \ldots, A$.

$$\min_{\boldsymbol{z}} \sum_{k=1}^{K} \frac{d_k(\boldsymbol{z})}{d_k^{\mathrm{ref}}} + \frac{d_k^{\mathrm{ref}}}{d_k(\boldsymbol{z})} \tag{4.1}$$

$$d_k(\boldsymbol{z}) = \max_{\alpha_k} \sum_{b=1}^{B} n_b g\left[h(\sigma_{1,kb}(\alpha_k, \boldsymbol{z}), \sigma_{2,kb}(\alpha_k, \boldsymbol{z}))\right], \tag{4.2}$$

$$\sigma_{j,kb}(\alpha_k, \boldsymbol{z}) = \boldsymbol{c}(\alpha_k)^T \boldsymbol{\sigma}_{j,kb}(\bar{\boldsymbol{p}}, \bar{\boldsymbol{\tau}}), \tag{4.3}$$

$$\boldsymbol{\sigma}_{j,kb}(\bar{\boldsymbol{p}}, \bar{\boldsymbol{\tau}}) = \sum_{i=1}^{A} p_{j,bi}\, \mathcal{B}_{ki}\, \boldsymbol{l}(\boldsymbol{\tau}_i), \quad \text{for} \quad j = 1,\, 2, \tag{4.4}$$

where $n_b > 0$ are number of stress-strain cycles per block, $\mathcal{B}_{ki} \in \mathbb{R}^{3\times 6}$ contain stresses, generated by both actuators load and the reaction load of the mounted attachment, $\boldsymbol{l} : \mathbb{R}^T \mapsto \mathbb{R}^6$ is the alignment function, $\boldsymbol{c} : \mathbb{R} \mapsto \mathbb{R}^3$ denotes the critical plane normal (see Def. 2.1.15):

$$\boldsymbol{c}(\alpha) = [1 + \cos\alpha, 1 - \cos\alpha, 2\sin\alpha]/2 \tag{4.5}$$

The function $g : \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+$ is the damage curve:

$$g(\sigma_f) := \frac{1}{N_s} \begin{cases} (\sigma_f/\sigma_s)^{k_1} & \text{for} \quad \sigma_f \leq \sigma_s, \\ (\sigma_f/\sigma_s)^{k_2} & \text{for} \quad \sigma_f \geq \sigma_s, \end{cases} \tag{4.6}$$

and $h : \mathbb{R}^2 \mapsto \mathbb{R}_0^+$ is the Goodman mean stress correction:

$$h(\sigma_1, \sigma_2) = \begin{cases} (1 - M)\, \sigma_a & \text{for} \quad \sigma_a < -\sigma_m \\ \sigma_a + M\sigma_m & \text{for} \quad \sigma_a \geq -\sigma_m \end{cases} \tag{4.7}$$

where $\sigma_a = |\sigma_1 - \sigma_2|/2$ and $\sigma_m = (\sigma_1 + \sigma_2)/2$ are the amplitude and mean of the stress-strain cycle respectively and the slope $M < 1$ is positive. Detailed description and physical meaning of these functions can be found in Sections 2.1.2 and 2.1.3.

Now, when the model is fixed and the functions it uses are defined, we can analyze the problem (4.1)–(4.4). Since it has two nested levels, we study them separately, starting with the inner one.

4.2 Inner Problem

For every instance of vector $\boldsymbol{z}$ we have the unconstrained global critical plane optimization problem (4.2)–(4.4) at every hot spot k. Let us rewrite the target functional to shorten notation:

$$\max_{\alpha} d(\alpha) \tag{4.8}$$

$$d(\alpha) = \sum_{b=1}^{B} d_b(\alpha) \tag{4.9}$$

$$d_b(\alpha) = n_b g\left[h(\boldsymbol{c}(\alpha)^T \boldsymbol{\sigma}_{1,b}, \boldsymbol{c}(\alpha)^T \boldsymbol{\sigma}_{2,b})\right], \tag{4.10}$$

We can restrict our optimization to interval $\alpha \in [0, 2\pi]$, because the plane $\boldsymbol{c}(\alpha)$ is periodic and so is the target $d(\alpha)$.

Remark 4.2.1 (Fully reversed loading)**.** If the test rig generates fully reversed loading, i.e. $\boldsymbol{\sigma}_{1,b} = -\boldsymbol{\sigma}_{2,b} =: \boldsymbol{\sigma}_b$, then Eq. (4.10) has the following form:

$$\max_{\alpha} d(\alpha) \tag{4.11}$$

$$d(\alpha) = \sum_{b=1}^{B} d_b(\alpha) \tag{4.12}$$

$$d_b(\alpha) = n_b g(|\boldsymbol{c}(\alpha)^T \boldsymbol{\sigma}_b|), \tag{4.13}$$

since $h(\boldsymbol{c}(\alpha)^T \boldsymbol{\sigma}_b, -\boldsymbol{c}(\alpha)^T \boldsymbol{\sigma}_b) = |\boldsymbol{c}(\alpha)^T \boldsymbol{\sigma}_b|$.

Next, several methods to solve the general critical plane optimization problem (4.8)–(4.10) and its fully reversed variation (4.11)–(4.13) will be discussed.

4.2.1 Simple Branch and Bound

In the critical plane optimization problem (4.8)–(4.10) the sum of the functions $d_b(\alpha)$ has to be maximized. Even though the local maxima of every summand can be found analytically, this is not true for the sum. Therefore, global optimization methods e.g. branch and bound (BnB) have to be used. BnB algorithms are global optimization methods that refine lower and upper bounds for the optimum, splitting the domain. Unfortunately, most of the procedures, calculating these bounds, do not utilize knowledge of the problem's underlying structure effectively. In this section we propose a BnB technique that uses local maxima of every d_b to divide the domain into several subdomains and then bounds the maximum of d, applying the following lemma.

Definition 4.2.2. A $\operatorname{Locmax}_{x\in X} f(x)$ is the set of all local maximum points of the function $f(x)$ in the domain X.

Lemma 4.2.3. *If $d_b(\alpha) : [l, r] \mapsto \mathbb{R}$ for $b = 1, \dots, B$ are continuous functions in the closed and bounded interval $[l, r]$, then the global maximum of the sum $d(\alpha) = d_1(\alpha) + \ldots + d_B(\alpha)$ has the following bounds:*

$$\max\{d(l), d(r)\} \le \max_{\alpha\in[l,r]} d(\alpha) \le \sum_{b=1}^{B} \max\{d_b(l), d_b(r)\} \qquad (4.14)$$

if $\operatorname*{Locmax}_{\alpha\in[l,r]} d_b(\alpha) = \operatorname*{Locmax}_{\alpha\in\{l,r\}} d_b(\alpha)$ *for* $b = 1, \dots, B$.

Proof. Obviously, the maximum of a function in the closed domain X is greater or equal than the maximum on the boundary ∂X, since the feasible set is extended. Therefore, the first part of the inequality (4.14) is valid:

$$\max\{d(l), d(r)\} \le \max_{\alpha\in[l,r]} d(\alpha)$$

The functions $d_b(\alpha)$ attain their maximum values on the boundary, hence:

$$\max_{\alpha\in[l,r]} d(\alpha) \le \sum_{b=1}^{B} \max_{\alpha\in[l,r]} d_b(\alpha) \le \sum_{b=1}^{B} \max\{d_b(l), d_b(r)\},$$

which proves the lemma. □

Algorithm

In this section we provide the pseudocode for the Simple BnB algorithm that finds global maximum of the function $d = d_1 + \ldots + d_B$ on the interval $[l, r]$ with the tolerance ε. An object that represents intervals has four attributes: $left$ – left end of a segment, *right* – right end of a segment, *lower* – lower bound of $d(\alpha)$ on the interval and *upper* – upper bound of $d(\alpha)$ on the interval.

The intervals are contained in the max-priority queue I, keyed on the *upper* attribute. It supports two operations:

- Insert(I, z) – inserts the element z into the set I
- Extract-Max(I) – removes and returns the element of I with the largest key

For more details about priority queues, their complexity and possible implementations see [9].

Algorithm 5: $d^* = \text{SimpleBnB}(l_{\text{init}}, r_{\text{init}}, \Phi_{LB}, \Phi_{UB}, \varepsilon)$

```
Input: $[l_{\text{init}}, r_{\text{init}}]$ – initial interval
$\Phi_{\text{lb}}(l,r) = \max\{d(l), d(r)\}$ – lower bound function
$\Phi_{\text{ub}}(l,r) = \sum_{b=1,\dots,B} \max\{d_b(l), d_b(r)\}$ – upper bound function
$\varepsilon$ – tolerance
Output: $d^* = \max_{\alpha\in[l,r]} d_1(\alpha) + \ldots + d_B(\alpha)$
X = {l, r} // initial grid
// find local maxima of each summand $d_b(\alpha)$
for $b = 1$ to $B$ do $X = X \cup \text{Locmax}_{\alpha\in[l,r]} d_b(\alpha)$
$X = \text{Sort}(X)$
$L = -\infty$ // lower bound for the global maximum
$U = -\infty$ // upper bound for the global maximum
// define interval node
NODE $\equiv \{left, right, lower, upper\}$
// add intervals and bounds to the priority queue $I$
$I = \emptyset$
for $i = 1$ to $|X| - 1$ do
    $z = \text{NODE}(X_i, X_{i+1}, \Phi_{\text{lb}}(X_i, X_{i+1}), \Phi_{\text{ub}}(X_i, X_{i+1}))$
    if $z.lower \geq L$ then $L = z.lower$
    if $z.upper \geq U$ then $U = z.upper$
    if $z.upper - z.lower \geq \varepsilon$ then
        Insert($I$, $z$)
// split intervals with maximal upper bound
while $U - L \geq \varepsilon$ do
    $z = \text{Extract-Max}(I)$
    if $z.upper > L$ then
        $mid = (z.left + z.right)/2$
         // interval midpoint
        // consider left subinterval
        $y = \text{NODE}(z.left, mid, \Phi_{\text{lb}}(z.left, mid), \Phi_{\text{ub}}(z.left, mid))$
        if $y.lower \geq L$ then $L = y.lower$
        if $y.upper \geq U$ then $U = y.upper$
        if $y.upper - y.lower \geq \varepsilon$ AND $y.upper > L$ then
            Insert($I$, $y$)
        // consider right subinterval
        $y = \text{NODE}(mid, z.right, \Phi_{\text{lb}}(mid, z.right), \Phi_{\text{ub}}(mid, z.right))$
        if $y.lower \geq L$ then $L = y.lower$
        if $y.upper \geq U$ then $U = y.upper$
        if $y.upper - y.lower \geq \varepsilon$ AND $y.upper > L$ then
            Insert($I$, $y$)
    return $L$
```

We would like to illustrate the algorithm with a simple example. Fig. 4.1 shows the target function, decomposed into two summands.

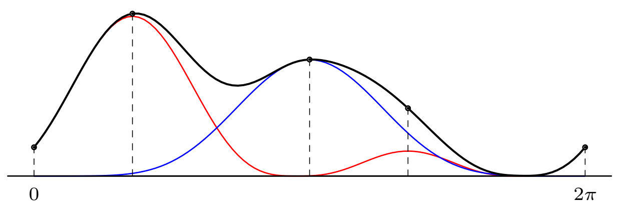


Figure 4.1: Sum (black) of a function with two local maxima (red) and a function with one local maximum (blue)

Algorithm 5 begins its search, computing the local maxima of the summands and using them to split the original interval into four subintervals (Fig. 4.2).

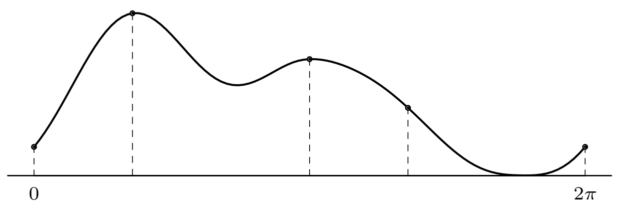


Figure 4.2: The interval $[0, 2\pi]$ is split into four parts

The lower and upper bounds of the target function for every interval are calculated using Lemma 4.2.3. Fig. 4.3 depicts them as sides of rectangles. Global bounds L and U are maxima of the lower and upper bounds for every interval.

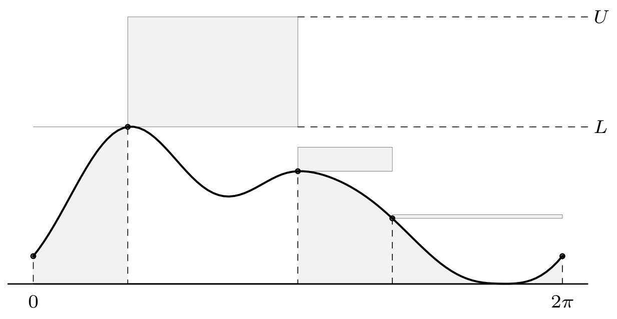


Figure 4.3: Lower and upper bounds of the target function for the intervals

If an interval has an upper bound smaller than L or the difference of its bounds is less than the given tolerance, it is pruned. Fig. 4.3 shows removed regions for the initial grid. The remaining interval is, then, divided into halves and the bounds are updated (Fig. 4.4).

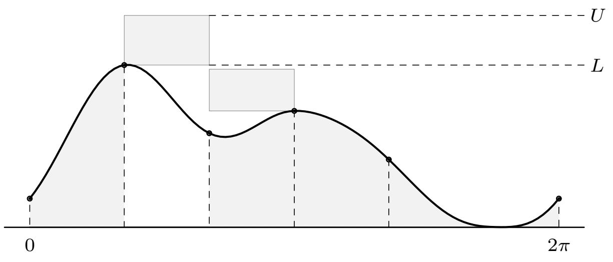


Figure 4.4: Bounds after the first refinement and pruned regions (gray)

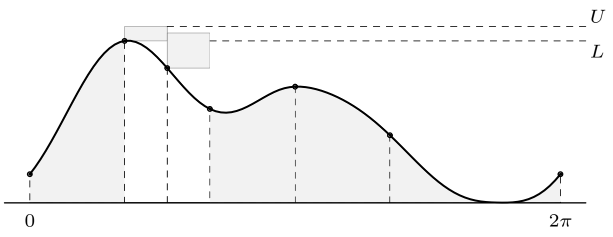


Figure 4.5: Bounds after the second refinement and pruned regions (gray)

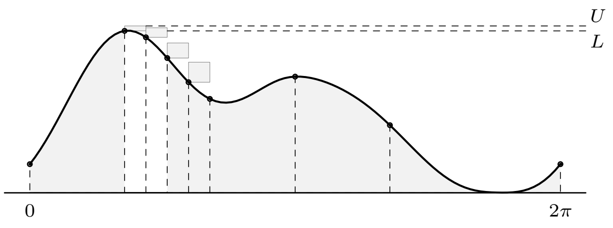


Figure 4.6: Bounds after the third refinement and pruned regions (gray)

Figs. 4.4–4.6 show, which subintervals are pruned and how the bounds are updated after several iterations of the algorithm. Observe that $|U - L|$ monotonically decreases. In the next section we analyze convergence of the algorithm.

Convergence

Lemma 4.2.3 states that the functions $\Phi_{\text{lb}}(l,r)$ and $\Phi_{\text{ub}}(l,r)$ defined by

$$\Phi_{\text{lb}}(l,r) = \max\left\{d(l), d(r)\right\}, \tag{4.15}$$

$$\Phi_{\text{ub}}(l,r) = \sum_{b=1}^{B} \max\left\{d_b(l), d_b(r)\right\}, \tag{4.16}$$

are bound functions of the global maximum of the sum $d(\alpha) = d_1(\alpha) + \ldots + d_B(\alpha)$ over the interval $[l,r]$, if there are no local maxima of a summand d_b for $b = 1, \ldots, B$ in the interior. In order for the BnB algorithm to converge in a finite number of steps, the functions $\Phi_{\text{lb}}(l,r)$ and $\Phi_{\text{ub}}(l,r)$ should satisfy conditions of the following theorem [3].

Theorem 4.2.4. *The BnB algorithm that finds the global maximum of a function $d(\alpha) : \mathbb{R} \mapsto \mathbb{R}$ over the interval $[l_{init}, r_{init}]$, converges in a finite number of steps, if the difference between the upper and lower bounds uniformly converges to zero as the length of any subinterval $[l,r]$ goes to zero:*

$$\begin{gathered}\forall\, \varepsilon > 0 \quad \exists\, \delta(\varepsilon) > 0 \quad \textit{such that} \\ \forall\, [l,r] \subset [l_{init}, r_{init}], \quad r - l < \delta \Rightarrow \Phi_{ub}(l,r) - \Phi_{lb}(l,r) \le \varepsilon\end{gathered}$$

where $\Phi_{lb}(l,r)$ and $\Phi_{ub}(l,r)$ are bound functions:

$$\Phi_{lb}(l,r) \le \max_{\alpha \in [l,r]} d(\alpha) \le \Phi_{ub}(l,r) \tag{4.17}$$

for any interval $[l,r]$.

Algorithm 5 uses local maxima of each d_b to split the initial interval $[l_{\text{init}}, r_{\text{init}}]$ into disjoint subsets, where the bound functions (4.15)–(4.16) are valid. Hence, we have to show only the uniform convergence in Theorem 4.2.4 to prove that Simple BnB comes to the end in a finite number of steps.

Lemma 4.2.5. *If the functions $d_b(\alpha)$ for $b = 1, \ldots, B$ are continuous on the closed interval $[l_{init}, r_{init}]$, then the difference between bound functions functions $\Phi_{lb}(l,r)$ and $\Phi_{ub}(l,r)$, defined by Eq. (4.15)–(4.16), is uniformly continuous on $[l_{init}, r_{init}]$.*

Proof. Continuity of the functions $d_b(\alpha)$ on a compact set $[l_{\text{init}}, r_{\text{init}}]$ for $b = 1, \ldots, B$ implies their uniform continuity, i.e.,

$$\begin{gathered}\forall\, \varepsilon > 0 \quad \exists\, \delta_b = \delta_b(\varepsilon) > 0 \quad \text{such that} \\ \forall\, x_1, x_2 \in [l_{\text{init}}, r_{\text{init}}], \quad |x_1 - x_2| < \delta_b \Rightarrow |d_b(x_1) - d_b(x_2)| \le \varepsilon\end{gathered}$$

The function $d(\alpha) = d_1(\alpha) + \ldots d_B(\alpha)$ is the sum of continuous functions, hence, it is also continuous and even uniformly continuous on $[l_{\text{init}}, r_{\text{init}}]$ with $\delta_{\text{sum}} = \delta_{\text{sum}}(\varepsilon) > 0$.

Let us fix $\varepsilon > 0$ and choose $\delta = \min\{\delta_1(\varepsilon), \dots, \delta_B(\varepsilon), \delta_{\text{sum}}(\varepsilon)\}$. Suppose $x_1, x_2 \in [l_{\text{init}}, r_{\text{init}}]$ and $|x_1 - x_2| < \delta$, then the absolute value between the bound functions $\Phi_{\text{lb}}(l,r)$ and $\Phi_{\text{ub}}(l,r)$ can be bounded from above:

$$\begin{aligned}
|\Phi_{\text{ub}}(l,r) - \Phi_{\text{lb}}(l,r)| &= \sum_{b=1}^{B} \max\{d_b(x_1), d_b(x_2)\} - \max\{d(x_1), d(x_2)\}, \\
&= \frac{1}{2}\Big| \sum_{b=1}^{B} (d_b(x_1) + d_b(x_2) + |d_b(x_1) - d_b(x_2)|) \\
&- d(x_1) - d(x_2) - |d(x_1) - d(x_2)|)\Big| \\
&= \frac{1}{2}\left| \sum_{b=1}^{B} |d_b(x_1) - d_b(x_2)| - |d(x_1) - d(x_2)| \right| \\
&\le \frac{1}{2}\sum_{b=1}^{B} |d_b(x_1) - d_b(x_2)| + \frac{1}{2}\left| \sum_{b=1}^{B} d_b(x_1) - \sum_{b=1}^{B} d_b(x_2) \right| \\
&\le \varepsilon B
\end{aligned}$$

which proves the lemma. □

Lemma 4.2.3 and 4.2.5 provide all necessary conditions for Theorem 4.2.4. Hence, Simple BnB (Algorithm 5) converges to the global maximum in a finite number of steps.

Local maxima

Simple BnB algorithm starts with splitting the initial interval into disjoint subintervals, having no local maxima of $d_b(\alpha)$ for $b = 1, \dots, B$ in their interior. Therefore, before Algorithm 5 can be applied to solve the critical plane optimization problems (4.8)–(4.10) and its fully reversed variation (4.11)–(4.13), all local maximum points of every summand have to be found analytically. In this section we provide necessary formulas for all cases of the damage function $d_b(\alpha)$.

Lemma 4.2.6. *The following holds:*

$$a_1 \cos x + a_2 \sin x = b_1 \sin(x + b_2) \tag{4.18}$$

where $b_1 = \sqrt{a_1^2 + a_2^2}$ *and* $b_2 = \arccos(a_2/b_1)\operatorname{sgn} a_1$.

Proof. Using the sine of sum identity, Eq.(4.18) can be transformed into

$$\begin{aligned} b_1 \sin(x+b_2) &= b_1 \left(\sin b_2 \cos x + \cos b_2 \sin x\right) \\ &= b_1 \sin\left(\arccos\left(a_2/b_1\right)\right) \cos x + a_2 \sin x \\ &= \sqrt{a_1^2+a_2^2}\ \operatorname{sgn} a_1 \sqrt{1-\frac{a_2^2}{a_1^2+a_2^2}}\ \cos x + a_2 \sin x \\ &= \operatorname{sgn} a_1 |a_1| \cos x + a_2 \sin x \end{aligned}$$

Substitute $\operatorname{sgn} a_1 |a_1| = a_1$ to prove the statement. □

Lemma 4.2.7. *Scalar stress $\sigma(\alpha) = \boldsymbol{c}(\alpha)^T\boldsymbol{\sigma}$, where $\boldsymbol{c}(\alpha)$ is the critical plane normal, defined by Eq. (4.5), can be written in the following form:*

$$\sigma(\alpha) = \boldsymbol{c}(\alpha)^T\boldsymbol{\sigma} = q_1(\boldsymbol{\sigma}) \sin(\alpha + q_2(\boldsymbol{\sigma})) + q_3(\boldsymbol{\sigma}) \tag{4.19}$$

where $q_i(\boldsymbol{\sigma})$ are components of the function $\boldsymbol{q} : \mathbb{R}^3 \mapsto \mathbb{R}^3$, defined by

$$\boldsymbol{q}(\boldsymbol{\sigma}) = \begin{bmatrix} q_1(\boldsymbol{\sigma}) \\ q_2(\boldsymbol{\sigma}) \\ q_2(\boldsymbol{\sigma}) \end{bmatrix} = \begin{bmatrix} \sqrt{(\sigma_1-\sigma_2)^2/4+\sigma_3^2} \\ \arccos\left(\sigma_3/A\right) \operatorname{sgn}(\sigma_1-\sigma_2) \\ (\sigma_1+\sigma_2)/2 \end{bmatrix} \tag{4.20}$$

Proof. Substitute the critical plane normal Eq. (4.5) into the scalar stress $\sigma(\alpha) = \boldsymbol{c}(\alpha)^T\boldsymbol{\sigma}$ to get the following:

$$\sigma(\alpha) = \boldsymbol{c}(\alpha)^T\boldsymbol{\sigma}(\alpha) = \frac{\sigma_1-\sigma_2}{2}\cos\alpha + \sigma_3 \sin\alpha + \frac{\sigma_1+\sigma_2}{2} \tag{4.21}$$

Identity (4.18), applied to Eq. (4.21), proves the statement. □

Theorem 4.2.8. *If $g : \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+$ is monotonically non-decreasing and n is positive, then the function $d(\alpha) = ng(|\sigma(\alpha)|)$, where $\sigma(\alpha) = \boldsymbol{c}(\alpha)^T\boldsymbol{\sigma}$, has the following local maximum points:*

$$\operatorname*{Locmax}_{\alpha\in[0,2\pi]} d(\alpha) = \begin{cases} \left\{\frac{\pi}{2}-q_2\right\}, & \textit{if} \quad q_3-q_1 \geq 0, \\ \left\{\frac{3\pi}{2}-q_2\right\}, & \textit{if} \quad q_3+q_1 \leq 0, \\ \left\{\frac{\pi}{2}-q_2, \frac{3\pi}{2}-q_2\right\}, & \textit{else}, \end{cases} \tag{4.22}$$

where parameters $\boldsymbol{q} = \boldsymbol{q}(\sigma)$ are defined by Eq. (4.20).

Remark 4.2.9. Local maximum points in Eq. (4.22) do not always belong to the interval $[0, 2\pi]$, but we can always translate them if necessary, since the function $d(\alpha)$ is periodic with the period equal to 2π. Here and below, it is assumed, that the local maximum points have already been translated.

Proof. The function g is monotonically non-decreasing on $\mathbb{R}_0^+$. Hence, the composition $ng(|\sigma(\alpha)|)$ has the same set of local maximum points as $|\sigma(\alpha)|$. First-order optimality condition, using Formula (4.19), reads as follows:

$$|\sigma(\alpha)|' = |q_1 \sin(\alpha + q_2) + q_3|' = q_1 \cos(\alpha + q_2) \operatorname{sgn} \sigma(\alpha) = 0 \tag{4.23}$$

Eq. (4.23) has two solutions:

$$\alpha_1 = \frac{\pi}{2} - q_2 \tag{4.24}$$

$$\alpha_2 = \frac{3\pi}{2} - q_2 \tag{4.25}$$

The parameter q_1 is always positive by construction. If $q_3 - q_1 \geq 0$, then $\sigma(\alpha) \geq 0$ for all α. Hence, in this case α_1 is the only maximum point of $|\sigma(\alpha)|$. Analogously, when $q_3 + q_1 \leq 0$, then $\sigma(\alpha) \leq 0$ everywhere and α_2 is the only maximum point. If the function $\sigma(\alpha)$ changes its sign, then both α_1 and α_2 belong to $\operatorname*{Locmax}_{\alpha\in[0,2\pi]} |\sigma(\alpha)|$. □

Corollary 4.2.10. *If $g : \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+$ is monotonically non-decreasing and n is positive, then the function $d(\alpha) = ng(|\sigma(\alpha)|)$, where $\sigma(\alpha) = \boldsymbol{c}(\alpha)^T\boldsymbol{\sigma}$, has the following global maxima:*

- *if $q_3 \geq 0$, then $d^* = ng(q_1 + q_3)$ and $\alpha^* = \frac{\pi}{2} - q_2$*
- *if $q_3 \leq 0$, then $d^* = ng(q_1 - q_3)$ and $\alpha^* = \frac{3\pi}{2} - q_2$*

where parameters $\boldsymbol{q} = \boldsymbol{q}(\sigma)$ are defined by Eq. (4.20).

Proof. The corollary can be proved by computing $d(\alpha)$ for local maximum points $\alpha^* \in \operatorname*{Locmax}_{\alpha\in[0,2\pi]} |\sigma(\alpha)|$ in Theorem 4.2.8. □

Theorem 4.2.11. *If $g : \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+$ is monotonically non-decreasing and n is positive, $h : \mathbb{R}^2 \mapsto \mathbb{R}_0^+$ is Goodman mean stress correction, defined by Eq. (4.7), then the function $d(\alpha) = ng[h(\sigma(\alpha), \xi(\alpha))]$, where $\sigma(\alpha) = \boldsymbol{c}(\alpha)^T\boldsymbol{\sigma}$ and $\xi(\alpha) = \boldsymbol{c}(\alpha)^T\boldsymbol{\xi}$, has the following local maximum points:*

1. *$\alpha_1^* = \frac{\pi}{2} - q_2^{diff}$, if $q_1^{diff} \geq -q_3^{diff}$ and $\sigma(\alpha_1^*) \leq 0$*
2. *$\alpha_2^* = \frac{3\pi}{2} - q_2^{diff}$, if $q_1^{diff} \geq q_3^{diff}$ and $\xi(\alpha_2^*) \leq 0$*
3. *$\alpha_3^* = \frac{\pi}{2} - q_2^{left}$, if $q_1^{diff} \cos(q_2^{diff} - q_2^{left}) \geq -q_3^{diff}$ and $\sigma(\alpha_3^*) \geq 0$*
4. *$\alpha_4^* = \frac{\pi}{2} - q_2^{right}$, if $q_1^{diff} \cos(q_2^{diff} - q_2^{right}) \leq -q_3^{diff}$ and $\xi(\alpha_4^*) \geq 0$*

where the parameters $\boldsymbol{q}^{\mathit{diff}}$*,* $\boldsymbol{q}^{\mathit{left}}$ *and* $\boldsymbol{q}^{\mathit{diff}}$ *are defined by Eq. (4.20):*

$$\begin{aligned} \boldsymbol{q}^{\mathit{diff}} &= \boldsymbol{q}[\,(\boldsymbol{\sigma}-\boldsymbol{\xi})/2\,] &(4.26)\\ \boldsymbol{q}^{\mathit{left}} &= \boldsymbol{q}[\,(\boldsymbol{\sigma}(1+M)-\boldsymbol{\xi}(1-M))/2\,] &(4.27)\\ \boldsymbol{q}^{\mathit{right}} &= \boldsymbol{q}[\,(\boldsymbol{\xi}(1+M)-\boldsymbol{\sigma}(1-M))/2\,] &(4.28)\end{aligned}$$

Proof. The function g is monotonically non-decreasing on $\mathbb{R}_0^+$. Hence, the composition $ng[h(\sigma(\alpha),\xi(\alpha))]$ has the same set of local maximum points as $h(\sigma(\alpha),\xi(\alpha))$.

Consider Goodman mean stress correction, defined by Eq. (4.7):

$$h(\sigma(\alpha),\xi(\alpha)) = \frac{1}{2}\begin{cases} (1-M)\,|\sigma-\xi| & \text{for} \quad |\sigma-\xi| < -(\sigma+\xi)\\ |\sigma-\xi| + M(\sigma+\xi) & \text{for} \quad |\sigma-\xi| \geq -(\sigma+\xi)\end{cases}$$

and expand the modulus $|\sigma-\xi|$:

$$h(\sigma(\alpha),\xi(\alpha)) = \frac{1}{2}\begin{cases} (1-M)\,(\sigma(\alpha)-\xi(\alpha)) \\ \quad \text{for} \quad \sigma(\alpha)-\xi(\alpha)\geq 0 \quad \text{and} \quad \sigma(\alpha)\leq 0\\ (1-M)\,(\xi(\alpha)-\sigma(\alpha)) \\ \quad \text{for} \quad \sigma(\alpha)-\xi(\alpha)\leq 0 \quad \text{and} \quad \xi(\alpha)\leq 0\\ (1+M)\sigma(\alpha)-(1-M)\xi(\alpha) \\ \quad \text{for} \quad \sigma(\alpha)-\xi(\alpha)\geq 0 \quad \text{and} \quad \sigma(\alpha)\geq 0\\ (1+M)\xi(\alpha)-(1-M)\sigma(\alpha) \\ \quad \text{for} \quad \sigma(\alpha)-\xi(\alpha)\leq 0 \quad \text{and} \quad \xi(\alpha)\geq 0 \end{cases} \tag{4.29}$$

The difference $(\sigma(\alpha)-\xi(\alpha))/2$ between the turning points $\sigma(\alpha)$ and $\xi(\alpha)$ can be decomposed, using Eq. (4.19):

$$(\sigma(\alpha)-\xi(\alpha))/2 = \boldsymbol{c}(\alpha)^T\,(\boldsymbol{\sigma}-\boldsymbol{\xi})/2 = q_1^{\text{diff}}\sin(\alpha+q_2^{\text{diff}})+q_3^{\text{diff}}, \tag{4.30}$$

where $\boldsymbol{q}^{\text{diff}} = \boldsymbol{q}[\,(\boldsymbol{\sigma}-\boldsymbol{\xi})/2\,]$. Analogously, decompose the remaining parts of the piecewise function h:

$$\begin{aligned} (1+M)\sigma(\alpha)-(1-M)\xi(\alpha) &= q_1^{\text{left}}\sin(\alpha+q_2^{\text{left}})+q_3^{\text{left}}, &(4.31)\\ (1+M)\xi(\alpha)-(1-M)\sigma(\alpha) &= q_1^{\text{right}}\sin(\alpha+q_2^{\text{right}})+q_3^{\text{right}}, &(4.32)\end{aligned}$$

where $\boldsymbol{q}^{\text{left}} = \boldsymbol{q}[(1+M)\sigma-(1-M)\xi]$. and $\boldsymbol{q}^{\text{right}} = \boldsymbol{q}[(1+M)\xi-(1-M)\sigma]$.

Substitute Eqs. (4.30)–(4.32) in Eq. (4.29) and differentiate $h(\sigma(\alpha),\xi(\alpha))$

with respect to α:

$$h'(\alpha) = \begin{cases} (1-M)\, q_1^{\text{diff}} \cos(\alpha + q_2^{\text{diff}}) \\ \quad \text{for} \quad q_1^{\text{diff}} \sin(\alpha + q_2^{\text{diff}}) \geq -q_3^{\text{diff}} \quad \text{and} \quad \sigma(\alpha) \leq 0 \\ -(1-M)\, q_1^{\text{diff}} \cos(\alpha + q_2^{\text{diff}}) \\ \quad \text{for} \quad q_1^{\text{diff}} \sin(\alpha + q_2^{\text{diff}}) \geq -q_3^{\text{diff}} \quad \text{and} \quad \xi(\alpha) \leq 0 \\ q_1^{\text{left}} \cos(\alpha + q_2^{\text{left}}) \\ \quad \text{for} \quad q_1^{\text{diff}} \sin(\alpha + q_2^{\text{diff}}) \geq -q_3^{\text{diff}} \quad \text{and} \quad \sigma(\alpha) \geq 0 \\ q_1^{\text{right}} \cos(\alpha + q_2^{\text{right}}) \\ \quad \text{for} \quad q_1^{\text{diff}} \sin(\alpha + q_2^{\text{diff}}) \leq -q_3^{\text{diff}} \quad \text{and} \quad \xi(\alpha) \geq 0 \end{cases} \tag{4.33}$$

By solving the first-order optimality conditions $h'(\alpha) = 0$, we get local maxima for each of four regions and, hence, prove the statement. □

Theorems 4.2.8 and 4.2.11 provide formula to compute local maxima of each summand $d_b(\alpha)$ for $b = 1, \ldots, B$ of the target functionals in the critical plane optimization problem (4.8)–(4.10) and its fully reversed variation (4.11)–(4.13). This is necessary to ensure convergence of Simple BnB (Algorithm 5) to the global optimum in a finite number of steps, according to Theorem 4.2.4 and Lemmas 4.2.3 and 4.2.5. The question is whether Simple BnB is fast enough to compete with the existing methods.

4.2.2 DIRECT Algorithm

DIRECT is a derivative-free global optimization algorithm, modification of the Lipschitzian approach, e.g. Shubert algorithm [38]. In this section we describe the key ideas of the method, based on [21].

It is assumed that the function $d(\alpha)$ is Lipschitz continuous in the interval $[l_{\text{init}}, r_{\text{init}}]$, i.e. there exists a constant K such that

$$|d(\alpha_1) - d(\alpha_2)| \leq K|\alpha_1 - \alpha_2|, \tag{4.34}$$

for any α_1, $\alpha_2 \in [l_{\text{init}}, r_{\text{init}}]$. Hence, if the Lipschitz function $d(\alpha)$ is evaluated at the midpoint $m = (l + r)/2$ of any closed interval $[l, r] \subset [l_{\text{init}}, r_{\text{init}}]$, then $d(\alpha)$ satisfies the inequalities:

$$d(\alpha) \leq d(m) - K(\alpha - m) \quad \text{for} \quad \alpha \leq m \tag{4.35}$$
$$d(\alpha) \leq d(m) + K(\alpha - m) \quad \text{for} \quad \alpha \geq m \tag{4.36}$$

Fig. 4.7 shows linear bounds, defined by Eqs. (4.35)–(4.36).

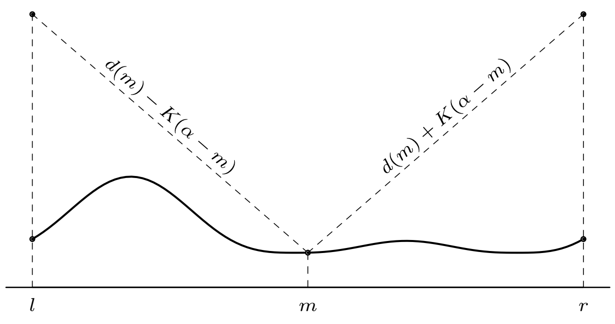


Figure 4.7: Upper bound with center-point sampling

The linear bounds can attain their highest value only at the endpoints l and r, hence, the upper bound is given by

$$\max_{\alpha \in [l,r]} d(\alpha) \leq d(m) + K(r - l)/2 \tag{4.37}$$

Center-point sampling can be maintained, if in the branching step the interval $[l, r]$ is divided into three equal subintervals $[l, l + (r - l)/3]$, $[l + (r - l)/3, l + 2(r - l)/3]$ and $[l + 2(r - l)/3, r]$, since in this case the point m becomes the midpoint of the second subinterval and only two function values have to be computed. The branching step partitions the interval into many. The order in which subintervals are chosen and considered, has to be specified. The basic idea of DIRECT is to select a subset of *potentially optimal* intervals and sample each of them.

Definition 4.2.12. Suppose that the interval $[l_{\text{init}}, r_{\text{init}}]$ is partitioned into intervals $[l_i, r_i]$ with midpoints m_i, for $i = 1, \dots, I$. Let $\varepsilon > 0$ be a positive constant, and the $d_{\max}$ be the current best function value. Interval j is said to be potentially optimal if there exists some rate-of-change constant $\tilde{K} > 0$ such that

$$\begin{aligned} &d(m_j) + \tilde{K}[(r_j - l_j)/2] \geq d(m_i) + \tilde{K}[(r_i - l_i)/2] \quad \text{for} \quad i = 1, \dots, I \\ &d(m_j) + \tilde{K}[(r_j - l_j)/2] \geq d_{\max} + \varepsilon |d_{\max}| \end{aligned}$$

The first condition requires the interval to be on the upper right of the convex hull of the dots (Fig. 4.8). The second condition forces the upper bound for the interval to exceed the current best solution.

Remark 4.2.13. The set of potentially optimal intervals can be identified in $O(m')$ time, where m' is the number of distinct interval lengths (abscissas in Fig. 4.8).

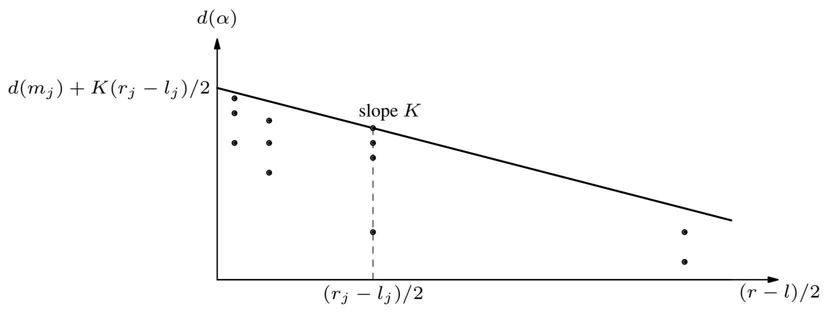


Figure 4.8: Representation of subintervals and corresponding function values by dots

In a standard branch-and-bound algorithm the most promising intervals with the largest upper bound would be selected, partitioned and pruned, if their upper bound is lower then the current best function value $d_{\max}$. BnB stops, when the difference between the upper and lower bounds for the maximum are below the given tolerance. If the pruning is skipped, the stopping criterion is changed to a limit on the number of iterations and at every step we select only potentially optimal intervals, then the Lipschitz constant K does not have to be known anymore.

Remark 4.2.14. DIRECT is guaranteed [21] to converge to the globally optimal function value if the target function is continuous in the neighborhood of a global optimum.

Two promising deterministic methods to solve the critical optimization problem (4.8)–(4.10) were described above. A comparison of the methods in terms of CPU time continues the chapter.

4.2.3 Numerical Results

For this study, Simple BnB algorithm was written in C++. The implementation of DIRECT method was provided by an open-source nonlinear optimization library NLopt [20]. In order to benchmark the performance of both algorithms, 100000 randomly generated instances of each problem were solved and the average CPU time is compared.

The critical plane optimization problem (4.8)–(4.10) takes as an input two stress vectors $\boldsymbol{\sigma}_{1,b} \in \mathbb{R}^3$ and $\boldsymbol{\sigma}_{2,b} \in \mathbb{R}^3$ for every block $b = 1, \dots, B$. Its fully reversed variation (4.11)–(4.13) needs only one vector $\boldsymbol{\sigma}_b \in \mathbb{R}^3$ per block. The elements of the input were uniformly randomly chosen in the interval $[-100, 100]$ and the number of stress-strain cycles n_b is fixed at 30000. The damage curve $g(\sigma_f)$, defined by Eq. (4.6), is initialized with the parameters $N_s = 10^6$, $\sigma_s = 80$ and $k_1 = k_2 = 5$. The Goodman mean stress correction (Eq. (4.7)) has the slope $M = 0.3$.

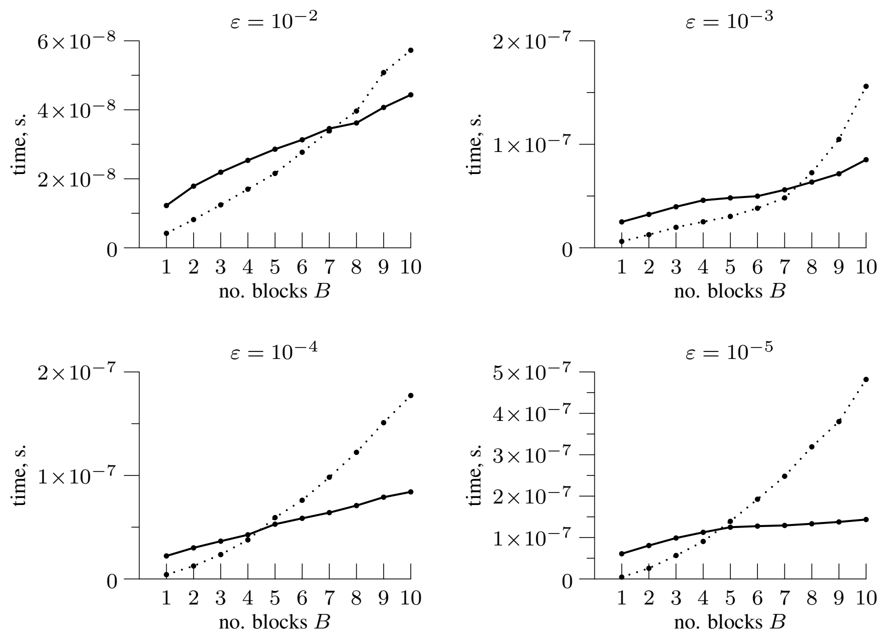


Figure 4.9: Performance comparison of Simple BnB (dotted) and DIRECT (solid) for the critical plane optimization problem (4.8)–(4.10)

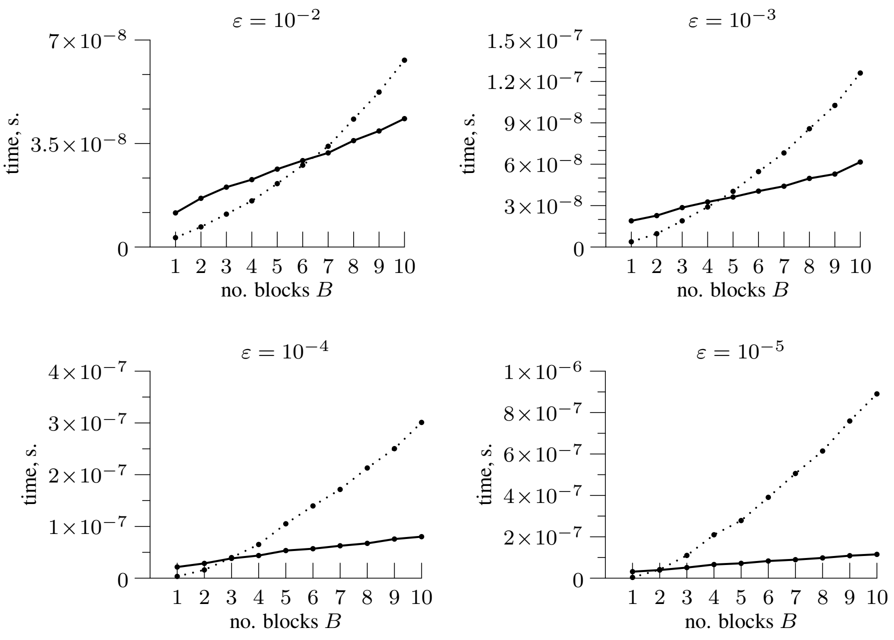


Figure 4.10: Performance comparison of Simple BnB (dotted) and DIRECT (solid) for the fully reversed loading variation (4.11)–(4.13)

Figs. 4.9, 4.10 show the average CPU time of Simple BnB and DIRECT for different tolerance values ε and number of blocks B in the loading signal. As can be seen in Fig. 4.9, the critical plane optimization problem (4.8)–(4.10) benefits from using Simple BnB method only for low tolerance $\varepsilon \in [10^{-3}, 10^{-2}]$, when $B \leq 7$, and for high tolerance $\varepsilon \in [10^{-5}, 10^{-4}]$, when $B \leq 4$. Fig. 4.10 shows that Simple BnB algorithm becomes less effective for the fully reversed variation (4.11)–(4.13). At the same time, the running time of DIRECT stays approximately the same, since the algorithm has a limit on the number of iterations. Therefore, it is likely that sometimes the method terminates before the global solution was found.

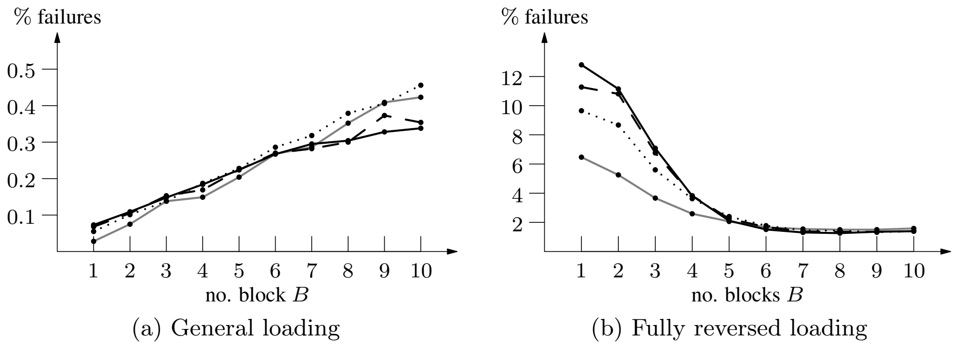


Figure 4.11: Failure rate of the algorithms for $\varepsilon = 10^{-2}$ (gray), $\varepsilon = 10^{-3}$ (dotted), $\varepsilon = 10^{-4}$ (dashed), $\varepsilon = 10^{-4}$ (black)

As one can observe in Fig. 4.11a, the failure rate of DIRECT algorithm for the general loading is relatively low and slowly increases with the number of blocks B. It is not sensitive to the parameter ε. However, in case of fully reversed signal (see Fig. 4.11b) percentage of failures is immense for low number of blocks. This can be explained by the fact, that the summands $d_b(\alpha)$ have only one or two peak-like maxima, according to Theorem 4.2.8. If the center-point sampling misses the neighborhood of the needed maximum, then selection of potentially optimal intervals in DIRECT algorithm can lead to the exploration of wrong part of the domain. The method would have eventually converged and found the global solution, if the limit on the number of iterations were higher. Observe that the failure rate rapidly decreases in the first part of Fig. 4.11b, since there becomes more summands and, therefore, the maxima of the sum are likely to stop resembling peaks.

The results suggest to use Simple BnB to find global maximum of the critical plane optimization problem and its fully reversed variation for $B > 1$, if the solution has to be found in 100%. Otherwise, one can apply DIRECT algorithm for large number of blocks B to speed up the process. The trivial case of the constant loading with a single block $B = 1$ can be solved analytically, using the Theorems 4.2.8 and 4.2.11.

4.2.4 Equidistant grid

In engineering applications the critical optimization problem $\max_{\alpha\in[0,2\pi]} d(\alpha)$ is usually discretized, so that the function d is computed for every point of an equidistant grid [27] and then the maximum is selected. In this section we study deviation of such approximate solution from the exact maximum in case of constant fully reversed loading (Eqs. (4.11)–(4.13)) with one-slope damage curve $g(\sigma_f) = (\sigma_f/\sigma_s)^k/N_s$:

$$\max_{\alpha\in[0,2\pi]} d(\alpha) \tag{4.38}$$

$$d(\alpha) = \nu|\boldsymbol{c}(\alpha)^T\boldsymbol{\sigma}|^k, \tag{4.39}$$

where $\nu = n/\left(N_s\,\sigma_s^k\right)$ is a positive constant. Let us rewrite the damage $d(\alpha)$ in Eq. (4.38) using Lemma 4.2.7:

$$d(\alpha) = \nu|q_1(\boldsymbol{\sigma})\sin(\alpha + q_2(\boldsymbol{\sigma})) + q_3(\boldsymbol{\sigma})|^k, \tag{4.40}$$

where $q_i(\boldsymbol{\sigma})$ are components of the function $\boldsymbol{q} : \mathbb{R}^3 \mapsto \mathbb{R}^3$ is defined by

$$\boldsymbol{q}(\boldsymbol{\sigma}) = \begin{bmatrix} q_1(\boldsymbol{\sigma}) \\ q_2(\boldsymbol{\sigma}) \\ q_2(\boldsymbol{\sigma}) \end{bmatrix} = \begin{bmatrix} \sqrt{(\sigma_1-\sigma_2)^2/4+\sigma_3^2} \\ \arccos\left(\sigma_3/A\right)\operatorname{sgn}(\sigma_1-\sigma_2) \\ (\sigma_1+\sigma_2)/2 \end{bmatrix} \tag{4.41}$$

We study this special case to provide a lower bound on the maximum deviation for more general cases.

Lemma 4.2.15. *The exact solution of the problem (4.38)–(4.40) has the following maximum deviation from the discrete solution:*

$$\begin{aligned} E &= \max_{\alpha\in[0,2\pi]} d(\alpha) - \max_{i=0,\dots,N} d(\alpha_i), \\ &= \nu\left((q_1+|q_3|)^k - (q_1\cos\frac{\delta}{2}+|q_3|)^k\right), \end{aligned} \tag{4.42}$$

where the finite sequence $\alpha_i = i\delta$ for $i = 0,\dots,N$ generates an equidistant grid with the step $\delta = 2\pi/N$.

Proof. According to Corollary 4.2.10, the problem (4.38)–(4.39) has the global maximum point at $\alpha^* = \pi/2 - q_2$ if $q_3 \geq 0$ or at $\alpha^* = 3\pi/2 - q_2$ if $q_3 \leq 0$ with the following target function value:

$$\max_{\alpha\in[0,2\pi]} d(\alpha) = \nu(q_1+|q_3|)^k \tag{4.43}$$

Suppose the grid is fine enough, so that the discrete solution is in the δ-neighborhood of the global maximum point α^*:

$$\max_{i=0,\dots,N} d(\alpha_i) = \max\{d(\alpha_{i^*}), d(\alpha_{i^*+1})\} \tag{4.44}$$

where $\alpha_{i^*} = \lfloor \alpha^*/\delta \rfloor \delta$. Reflectional symmetry of the function $d(\alpha)$ at α^* implies that the deviation is maximized, when α^* is in the middle of the interval $[\alpha_{i^*}, \alpha_{i^*+1}]$, i.e. $\alpha_{i^*} = \alpha^* - \delta/2$. Therefore, the deviation E takes the following form:

$$\begin{aligned}
E &= \max_{\alpha \in [0,2\pi]} d(\alpha) - \max_{i=0,\ldots,N} d(\alpha_i), \\
&= \nu(q_1 + |q_3|)^k - \max\{d(\alpha_{i^*}), d(\alpha_{i^*+1})\} \\
&= \nu(q_1 + |q_3|)^k - d(\alpha^* + \delta/2) \\
&= \nu(q_1 + |q_3|)^k - \nu\,|q_1 \sin(\alpha^* + \delta/2 + q_2) + q_3|^k \\
&= \nu(q_1 + |q_3|)^k - \nu\,(q_1 \sin(\alpha^* + \delta/2 + q_2) + |q_3|)^k
\end{aligned}$$

Substitute the global maximum point $\alpha^* = \pi/2 - q_2$ to prove the lemma. □

The deviation E in Lemma 4.2.15 depends on the stress vector $\boldsymbol{\sigma}$. Next, the instance of $\boldsymbol{\sigma}$ that satisfies the box constraints (2.35) and maximizes $E(\boldsymbol{\sigma})$ is derived. Hence, we have to solve the following problem:

$$\max_{\sigma} E(\boldsymbol{\sigma}) \tag{4.45}$$

$$E(\boldsymbol{\sigma}) = g(q_1(\boldsymbol{\sigma}) + |q_3(\boldsymbol{\sigma})|) - g(q_1(\boldsymbol{\sigma}) \cos\frac{\delta}{2} + |q_3(\boldsymbol{\sigma})|)), \tag{4.46}$$

$$g(x) = \nu x^k, \tag{4.47}$$

$$q_1(\boldsymbol{\sigma}) = \sqrt{(\sigma_1 - \sigma_2)^2/4 + \sigma_3^2} \tag{4.48}$$

$$q_3(\boldsymbol{\sigma}) = (\sigma_1 + \sigma_2)/2 \tag{4.49}$$

$$-\boldsymbol{e}\sigma^{\max} \le \boldsymbol{\sigma} \le \boldsymbol{e}\sigma^{\max} \tag{4.50}$$

The solution leads to the formula to compute maximum possible deviation of the exact solution of the problem (4.38)–(4.40) from the discrete one, given the maximal stress $\sigma^{\max}$.

Lemma 4.2.16. *The function $\tilde{E}(x,y)$*

$$\tilde{E}(x,y) = g(x+y) - g(\epsilon x + y), \tag{4.51}$$

where g is defined by Eq. (4.47) and $\epsilon < 1$, is non-decreasing in each argument and convex for $x, y \in \mathbb{R}_0^+$.

Proof. First, we prove that the function $\tilde{E}(x,y)$ is non-decreasing in each argument $x, y \in \mathbb{R}_0^+$ by looking at its gradient:

$$\nabla\tilde{E} = \begin{bmatrix} g'(x+y) - \epsilon g'(\epsilon x + y) \\ g'(x+y) - g'(\epsilon x + y) \end{bmatrix},$$

Since $x+y \geq \epsilon x+y$ and g' is monotonically increasing, the gradient $\nabla \tilde{E} \geq 0$ and, therefore, the function $\tilde{E}$ is non-decreasing in x and y. Convexity of $\tilde{E}$ with respect to $[x;y]$ is proved by checking the positive definiteness of the Hessian matrix $\nabla^2 \tilde{E}$:

$$\nabla^2 \tilde{E} = \begin{bmatrix} g''(x+y) - \epsilon^2 g''(\epsilon x + y) & g''(x+y) - \epsilon g''(\epsilon x + y) \\ g''(x+y) - \epsilon g''(\epsilon x + y) & g''(x+y) - g''(\epsilon x + y) \end{bmatrix}$$

The statement follows from Sylvester's criterion and monotonicity of g''. □

Lemma 4.2.17. *The functions $q_1(\boldsymbol{\sigma})$ and $|q_3(\boldsymbol{\sigma})|$, defined by Eq. (4.49)–(4.50), are convex.*

Proof. The function $|q_3(\boldsymbol{\sigma})| = |(\sigma_1 + \sigma_2)/2|$ is an absolute value of a plane and, hence, convex by Lemma 3.1.3. The Hessian matrix of $q_1(\boldsymbol{\sigma})$ has the following form:

$$\nabla^2 q_1(\boldsymbol{\sigma}) = \frac{1}{4q_1^3(\boldsymbol{\sigma})} \begin{bmatrix} \sigma_3^2 & -\sigma_3^2 & (\sigma_2 - \sigma_1)\sigma_3 \\ -\sigma_3^2 & \sigma_3^2 & (\sigma_1 - \sigma_2)\sigma_3 \\ (\sigma_2 - \sigma_1)\sigma_3 & (\sigma_1 - \sigma_2)\sigma_3 & (\sigma_1 - \sigma_2)^2 \end{bmatrix} \tag{4.52}$$

The eigenvalues of the Hessian $\left\{0, 0, (\sigma_1 - \sigma_2)^2 + 2\sigma_3^2\right\}$ are non-negative, which means it is positive semidefinite and $q_1(\boldsymbol{\sigma})$ is convex. □

Theorem 4.2.18. *The function $E(\boldsymbol{\sigma})$, defined by Eq. (4.46), is convex.*

Proof. The function $E(\boldsymbol{\sigma})$ is a composition of $\tilde{E}(x, y)$, which is convex non-decreasing in each argument according to Lemma 4.2.16, and convex functions $q_1(\boldsymbol{\sigma})$ and $|q_3(\boldsymbol{\sigma})|$ by Lemma 4.2.17. Therefore, Lemma 3.1.5 proves the statement. □

Corollary 4.2.19. *The target functional $E(\boldsymbol{\sigma})$ of the problem (4.45)–(4.50) attains its maximum at $\boldsymbol{\sigma}^* = [\sigma^{\max}, \sigma^{\max}, -\sigma^{\max}]$:*

$$E(\boldsymbol{\sigma}^*) = \nu \, [\sigma^{\max}]^k \left(2^k - (1 + \cos \delta/2))^k\right). \tag{4.53}$$

Proof. The problem (4.45)–(4.50) is a maximization problem of the convex function $E(\boldsymbol{\sigma})$ over the convex set. Hence, the maximum is attained at the boundary. Since the feasible set is the cube $-\boldsymbol{e}\sigma^{\max} \leq \boldsymbol{\sigma} \leq \boldsymbol{e}\sigma^{\max}$, the search can be restricted to its vertices. Hence, there are 8 candidates. The symmetry of the functions $E(\boldsymbol{\sigma})$, $q_1(\boldsymbol{\sigma})$ and $|q_3(\boldsymbol{\sigma})|$ reduces the number to only 2 vertices. Therefore, comparing the target functional E at the points $\boldsymbol{\sigma}_1 = [\sigma^{\max}, \sigma^{\max}, \sigma^{\max}]$ and $\boldsymbol{\sigma}_2 = [\sigma^{\max}, -\sigma^{\max}, \sigma^{\max}]$, we prove the corollary. □

Remark 4.2.20. For the stress vector $\boldsymbol{\sigma}^*$ the maximum of the critical plane optimization problem $d(\alpha^*) = \nu(2\sigma^{\max})^k$ and, hence, the relative deviation $E(\boldsymbol{\sigma}^*)/d(\alpha^*)$ does not depend on $\sigma^{\max}$. Fig. 4.12 shows the relative error for different values of the damage curve parameter k and number of nodes in the equidistant grid N.

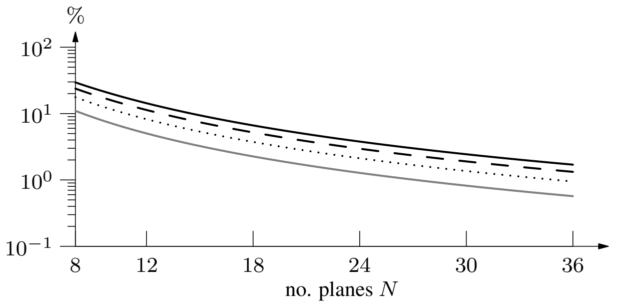


Figure 4.12: Relative error for $k = 3$ (gray), $k = 5$ (dotted), $k = 7$ (dashed) and $k = 9$ (solid)

The number of planes N, used in industry [27, 40], usually equals 8, 12, 18, 24, 30, 36. As one can observe in Fig. 4.12, the relative error for such grid can be large, therefore, it has to be refined or global optimization methods should be used.

In this section the solution methods for the inner problem of the bilevel test rig problem (4.1)–(4.4) were discussed. Simple BnB algorithm was presented and compared with DIRECT method. Both of them converge to the global optimum. Simple BnB always terminates its search in the ε-neighborhood of the exact solution. Although DIRECT algorithm can be faster for some instances of the problem, it does not provide any certificate for optimality, so the limit on the number of iterations should be adjusted. Also the usual approach to maximize the damage, considering a finite number of planes, i.e. discretizing the problem, was studied. We showed that the number of planes has to be dramatically increased to get close to the optimum for all possible input vectors. Therefore, applications, demanding high precision, have to obsolete this method.

4.3 Outer problem

In this section we discuss regularity of the outer problem and prove that it is continuous and differentiable under certain conditions. Let us recapitulate the outer part of the general test rig problem (4.1)–(4.4) for a single criterion:

$$\min_{\boldsymbol{z}} F(\boldsymbol{z}), \tag{4.54}$$

$$F(\boldsymbol{z}) = \frac{d(\boldsymbol{z})}{d^{\text{ref}}} + \frac{d^{\text{ref}}}{d(\boldsymbol{z})}, \tag{4.55}$$

$$d(\boldsymbol{z}) = \max_{\alpha \in [0,2\pi]} d(\alpha, \boldsymbol{z}), \tag{4.56}$$

$$d(\alpha, \boldsymbol{z}) = \sum_{b=1}^{B} n_b g\left[h(\sigma_{1,b}(\alpha, \boldsymbol{z}), \sigma_{2,b}(\alpha, \boldsymbol{z}))\right], \tag{4.57}$$

where the reference damage values $d(\boldsymbol{z})$ are positive and the state vector $\boldsymbol{z} = [\bar{\boldsymbol{p}}; \bar{\boldsymbol{\tau}}]$ includes both applied forces $\bar{\boldsymbol{p}} = [\boldsymbol{p}_{1,1}, \boldsymbol{p}_{2,1}, \dots, \boldsymbol{p}_{1,B}, \boldsymbol{p}_{2,B}] \in \mathbb{R}^{2AB}$ and alignment parameters $\bar{\boldsymbol{\tau}} = [\boldsymbol{\tau}_1, \dots, \tau_A] \in \mathbb{R}^{AT}$ for actuators $i = 1, \dots, A$.

First, convexity of the damage function $d(\boldsymbol{z}) = d([\bar{\boldsymbol{p}}; \bar{\boldsymbol{\tau}}])$ with respect to applied forces $\bar{\boldsymbol{p}}$ is proven. This property can be used to reformulate the problem as a DC program for every fixed alignment $\bar{\boldsymbol{\tau}}$, using the same decomposition as in Section 3.1.2.

Theorem 4.3.1. *Damage function $d(\boldsymbol{z}) = \max_{\alpha \in [0,2\pi]} d(\alpha, [\bar{\boldsymbol{p}}; \bar{\boldsymbol{\tau}}])$, defined by Eqs. (4.56)–(4.57), is convex with respect to $\bar{\boldsymbol{p}}$, if the damage curve g is convex and non-decreasing.*

Proof. The function $d(\alpha, \boldsymbol{z})$ is convex for every fixed angle α by Theorem 3.1.7. Therefore, $d(\boldsymbol{z})$ is a pointwise maximum of convex functions $d(\alpha, \boldsymbol{z})$ and, hence, convex according to Lemma 3.1.1. □

The following theorems provide conditions for continuity and differentiability of the target functional $F(\boldsymbol{z})$.

Theorem 4.3.2 (Continuity)**.** *Let $\mathbb{A}$ be compact and function $f(\alpha, \boldsymbol{x})$ be continuous with respect to $\alpha \in \mathbb{A} \subset \mathbb{R}$ and $\boldsymbol{x} \in \mathbb{X} \subseteq \mathbb{R}^N$ in the neighborhood of $\boldsymbol{x}_0$ and $\alpha \in \mathbb{A}$. Then function*

$$\tilde{f}(\boldsymbol{x}) = \max_{\alpha \in \mathbb{A}} f(\alpha, \boldsymbol{x}) \tag{4.58}$$

is also continuous at the point $\boldsymbol{x}_0$.

Proof. See [36] for details. □

Theorem 4.3.3 (Differentiability)**.** *Let $f(\alpha, \boldsymbol{x})$ and $\partial_\alpha f(\alpha, \boldsymbol{x})$ be continuously differentiable with respect to $\alpha \in \mathbb{A} \subset \mathbb{R}$ and $\boldsymbol{x} \in \mathbb{X} \subseteq \mathbb{R}^N$. Also suppose, that there exists a neighborhood $\mathbb{V}$ of $\boldsymbol{x}_0$, where the argument of the maximum*

$$\alpha(\boldsymbol{x}) := \arg\max_{\alpha \in \mathbb{A}} f(\alpha, \boldsymbol{x}) \in \operatorname{int}(\mathbb{A})$$

is unique for each $\boldsymbol{x} \in \mathbb{V}$ and $\partial^2_\alpha f(\alpha(\boldsymbol{x}_0), \boldsymbol{x}_0) \neq 0$. Then the function $\tilde{f}(\boldsymbol{x})$, defined in Theorem 4.3.2 is differentiable at $\boldsymbol{x}_0$.

Proof. One way to prove differentiability of $\tilde{f}(\boldsymbol{x})$ is to prove existence and continuity of its partial derivatives at $\boldsymbol{x}_0$:

$$\begin{aligned}\frac{\partial \tilde{f}}{\partial x_i}(\boldsymbol{x_0}) &= \lim_{h\to 0} \frac{\tilde{f}(\boldsymbol{x}_0) - \tilde{f}(\boldsymbol{x}_0 + h\boldsymbol{e}_i)}{h} \\ &= \lim_{h\to 0} \frac{f(\alpha_0, \boldsymbol{x}_0) - f(\alpha_h, \boldsymbol{x}_0 + h\boldsymbol{e}_i)}{h} \\ &= \lim_{h\to 0} \frac{f(\alpha_0, \boldsymbol{x}_0) - f(\alpha_0, \boldsymbol{x}_0 + h\boldsymbol{e}_i)}{h} \\ &+ \lim_{h\to 0} \frac{f(\alpha_0, \boldsymbol{x}_0 + h\boldsymbol{e}_i) - f(\alpha_h, \boldsymbol{x}_0 + h\boldsymbol{e}_i)}{h},\end{aligned} \tag{4.59}$$

where $\alpha_h := \arg\max_{\alpha\in\mathbb{A}} f(\alpha, \boldsymbol{x}_0 + h\boldsymbol{e}_i)$. Now we consider the first term in Eq. (4.59). From the differentiablility of $f(\alpha, \boldsymbol{x})$ with respect to $\boldsymbol{x}$ it follows, that

$$\lim_{h\to 0} \frac{f(\alpha_0, \boldsymbol{x}_0) - f(\alpha_0, \boldsymbol{x}_0 + h\boldsymbol{e}_i)}{h} = \frac{\partial f}{\partial x_i}(\alpha_0, \boldsymbol{x}_0) \tag{4.60}$$

The second term can be rewritten, using the Taylor expansion

$$\lim_{h\to 0} \frac{f(\alpha_0, \boldsymbol{x}_0 + h\boldsymbol{e}_i) - f(\alpha_h, \boldsymbol{x}_0 + h\boldsymbol{e}_i)}{h} = \lim_{h\to 0} \frac{\alpha_0 - \alpha_h}{h}\, \partial_\alpha f(\alpha_0, \boldsymbol{x}_0 + h\boldsymbol{e}_i) \tag{4.61}$$

Since $\partial_\alpha f(\alpha, \boldsymbol{x})$ is continuous w.r.t. $\boldsymbol{x}$ and, moreover $\alpha_0 \in \operatorname{int}(\mathbb{A})$, then

$$\lim_{h\to 0} \partial_\alpha f(\alpha_0, \boldsymbol{x}_0 + h\boldsymbol{e}_i) = \partial_\alpha f(\alpha_0, \boldsymbol{x}_0) = 0$$

The function $\partial_\alpha f(\alpha, \boldsymbol{x})$ is continuously differentiable with respect to $\boldsymbol{x}$ and α and $\partial^2_\alpha f(\alpha_0, \boldsymbol{x_0}) \neq 0$, so we can apply the Implicit function theorem. Hence, there exists an open set $\mathbb{U}_{\boldsymbol{x}} \ni \boldsymbol{x}_0$ and an open set $\mathbb{U}_\alpha \ni \alpha_0$, and a unique continuously differentiable function $a(\boldsymbol{x}) : \mathbb{U}_{\boldsymbol{x}} \mapsto \mathbb{U}_\alpha$, such that $\alpha_h = a(\boldsymbol{x}_0 + h\boldsymbol{e}_i)$ for small enough h, when $\boldsymbol{x}_0 + h\boldsymbol{e}_i \in \mathbb{U}_\alpha$. That gives us the boundedness of $\lim_{h\to 0}(\alpha_0 - \alpha_h)/h$ in Eq. (4.59) and hence the expression Eq. (4.61) has the following form:

$$\lim_{h\to 0} \frac{\alpha_0 - \alpha_h}{h}\, \partial_\alpha f(\alpha_0, \mathbf{x}_0 + h\mathbf{e}_i) = a'(\mathbf{x_0})\, \partial_\alpha f(\alpha_0, \mathbf{x}_0) = 0 \tag{4.62}$$

So, combining Eq. (4.60) and Eq. (4.62), we get the derivative of $g(\boldsymbol{x})$ with respect to x_i:

$$\frac{\partial g}{\partial x_i}(\boldsymbol{x_0}) = \frac{\partial f}{\partial x_i}(\alpha_0, \boldsymbol{x}_0), \tag{4.63}$$

which is continuously differentiable. Therefore, $\tilde{f}(\boldsymbol{x})$ is also differentiable. □

Theorem 4.3.4. *The damage function $d(\boldsymbol{z})$, defined by Eqs. (4.56)–(4.57), has the following gradient at $\boldsymbol{z}_0$:*

$$\nabla d(\boldsymbol{z}_0) = \nabla_{\boldsymbol{z}} d(\alpha^*, \boldsymbol{z}_0), \tag{4.64}$$

if the function $d(\alpha, \boldsymbol{z})$ satisfies conditions of Theorem 4.3.3 at $\boldsymbol{z}_0$ and α^* is a solution of the inner problem $\max_{\alpha\in[0,2\pi]} d(\alpha, \boldsymbol{z}_0)$.

Proof. The gradient of $d(\boldsymbol{z})$ at $\boldsymbol{z}_0$ has the following form:

$$\nabla d(\boldsymbol{z}_0) = \nabla_{\boldsymbol{z}} d(\alpha^*, \boldsymbol{z}_0) + \partial_\alpha d(\alpha^*, \boldsymbol{z}_0)\nabla\alpha(\boldsymbol{z}_0), \tag{4.65}$$

where the partial derivative $\partial_\alpha d(\alpha^*, \boldsymbol{z}_0)$ equals to zero, because $d(\alpha, \boldsymbol{z}_0)$ is periodic with respect to α and the maximum point α^* of a differentiable function on an open set is a stationary point according to Fermat's theorem. The gradient $\nabla\alpha(\boldsymbol{z}_0)$ is finite by Theorem 4.3.3. Therefore, the second term in Eq. (4.64) can be neglected. □

Theorem 4.3.4 makes it possible to use derivative-based methods to find extrema of the outer part of the test rig problem (4.54)–(4.57). These local solutions can be exploited by a global optimization algorithm, e.g. multi level single linkage [6], to ensure convergence.

4.4 Conclusion

In this chapter bilevel formulation of the test rig problem was considered. In the inner part the sum of the functions has to be maximized. We derived formulas for local maximum points of every summand and used this information to develop a new branch-and-bound algorithm. Later it was proven to converge and showed good computational time, compared to DIRECT algorithm. Also we found that the worst case relative error for the discretized solution, commonly applied in industry, could achieve 10% and more, depending on the damage curve parameters. Consequently, this method should be used only with a very fine grid or replaced by an approach that guarantees ε-optimal solution, if an application demands high precision.

For the outer part we proved convexity of the damage function with respect to applied forces, if the alignment parameters of each actuator are fixed. That implies, that the test rig problem can be decomposed into a union of DC programs. Also we showed, that the target functional is continuous and differentiable and provided the formula for its gradient. Therefore, it can be solved by derivative-based global optimization methods.

Chapter 5

Numerical Investigation

We intend to answer several important questions. Can the model (2.28)–(2.33) find a good approximation of reference damages at the hot spots? Is optimization necessary with respect to all state variables or can we vary only applied forces $\boldsymbol{p}_{1,b}$ and $\boldsymbol{p}_{2,b}$ in each block $b = 1, \ldots, B$ and fix the alignment parameters $\boldsymbol{\tau}_i$ of the actuator $i = 1, \ldots, A$? Does a fully reversed loading signal provide appropriate results? How many blocks should the loading signal have? Do the extender moments play an important role in the overall result? The algorithms mentioned in the chapter 4 help us to address these questions numerically.

5.1 Problem

5.1.1 Model

The general test rig problem, defined by Eqs. (2.28)–(2.33), approximates the reference damage values $d_k^{\mathrm{ref}} > 0$ at K hot spots, generating a loading signal with B blocks. The state vector $\boldsymbol{z} = [\bar{\boldsymbol{p}}; \bar{\boldsymbol{\tau}}]$ includes applied forces $\bar{\boldsymbol{p}} = [\boldsymbol{p}_{1,1}, \boldsymbol{p}_{2,1}, \ldots, \boldsymbol{p}_{1,B}, \boldsymbol{p}_{2,B}] \in \mathbb{R}^{2AB}$ and alignment parameters $\bar{\boldsymbol{\tau}} = [\boldsymbol{\tau}_1, \ldots, \tau_A] \in \mathbb{R}^{AT}$ for actuators $i = 1, \ldots, A$.

$$\min_{\boldsymbol{z}} \frac{1}{K} \sum_{k=1}^{K} \frac{d_k(\boldsymbol{z})}{d_k^{\mathrm{ref}}} + \frac{d_k^{\mathrm{ref}}}{d_k(\boldsymbol{z})} - 2 \tag{5.1}$$

$$d_k(\boldsymbol{z}) = \max_{\alpha_k} \sum_{b=1}^{B} n_b g \left[h(\sigma_{1,kb}(\alpha_k, \boldsymbol{z}), \sigma_{2,kb}(\alpha_k, \boldsymbol{z})) \right], \tag{5.2}$$

$$\sigma_{j,kb}(\alpha_k, \boldsymbol{z}) = \boldsymbol{c}(\alpha_k)^T \boldsymbol{\sigma}_{j,kb}(\bar{\boldsymbol{p}}, \bar{\boldsymbol{\tau}}), \tag{5.3}$$

$$\boldsymbol{\sigma}_{j,kb}(\bar{\boldsymbol{p}}, \bar{\boldsymbol{\tau}}) = \sum_{i=1}^{A} p_{j,bi} \, \mathcal{B}_{ki} \, \boldsymbol{l}(\boldsymbol{\tau}_i), \quad \text{for} \quad j = 1,\, 2, \tag{5.4}$$

where $n_b > 0$ are number of stress-strain cycles per block, $\mathcal{B}_{ki} \in \mathbb{R}^{3\times 6}$ contain stresses, generated by both actuators load and the reaction load of the mounted attachment, $\boldsymbol{l} : \mathbb{R}^T \mapsto \mathbb{R}^6$ is the alignment function, defined by Eq. (2.24), $\boldsymbol{c} : \mathbb{R} \mapsto \mathbb{R}^3$ denotes the critical plane normal (see Def. 2.1.15).

The function $g : \mathbb{R}_0^+ \mapsto \mathbb{R}_0^+$ is the damage curve:

$$g(\sigma_f) := \frac{1}{N_s} \begin{cases} (\sigma_f/\sigma_s)^{k_1} & \text{for} \quad \sigma_f \leq \sigma_s, \\ (\sigma_f/\sigma_s)^{k_2} & \text{for} \quad \sigma_f \geq \sigma_s, \end{cases} \tag{5.5}$$

and $h : \mathbb{R}^2 \mapsto \mathbb{R}_0^+$ is the Goodman mean stress correction:

$$h(\sigma_1, \sigma_2) = \begin{cases} (1-M)\,\sigma_a & \text{for} \quad \sigma_a < -\sigma_m \\ \sigma_a + M\sigma_m & \text{for} \quad \sigma_a \geq -\sigma_m \end{cases} \tag{5.6}$$

where $\sigma_a = |\sigma_1 - \sigma_2|/2$ and $\sigma_m = (\sigma_1 + \sigma_2)/2$ are the amplitude and mean of the stress-strain cycle respectively and the slope $M < 1$ is positive. Detailed description and physical meaning of these functions can be found in Sections 2.1.2 and 2.1.3. The state $\boldsymbol{z}$ and the components of the stress vectors $\boldsymbol{\sigma}_{j,kb}$ are bounded as in Section 2.3.4.

5.1.2 Number of cycles

The number of stress-strain cycles per block n_b for $b = 1, \ldots, B$ are integer parameters of the test rig problem (5.1)–(5.4). They can be relaxed and used as the state variables in order to extend the feasible set. After the optimization procedure they have to be rounded and, of course, the goal must be recomputed.

5.1.3 Methods

Derivative information makes it possible to use the multi level single linkage method [6] for global optimization with the interior-point method [41] for local searches. The inner critical plane problem, defined by Eq. (5.2), is solved by Simple BnB algorithm, discussed in Section 4.2.1. The solvers were implemented in C++ using an external library IPOPT [41] for local optimization routine.

5.1.4 Critical spots

A solution $\boldsymbol{z}^*$ of the problem (5.1)–(5.4) allows us to find an optimal loading signal, producing the best feasible approximation for damage values d_k^{ref} at K hot spots, but this does not imply that the component cannot have extremely high damage, exceeding a given threshold, at some other node. We call such node a *critical spot* and its occurrence indicates that the component will fail there. Let us discuss several ways to deal with the critical spots:

1. Damage approximation at all spots of the component. All nodes can be considered as hot spots, making optimization computationally expensive, but removing possible critical spots.

2. Adaptive optimization extending damage approximation. After the optimal signal for the set of K hot spots is found, we use it to compute the damage at every other node of the mesh. If there are critical spots, we add the one with maximal damage to the criteria of the original problem and optimize for the set of $K + 1$ hot spots. This approach will converge because there is a finite number of nodes and in the worst case it will use all of them.

3. Adaptive optimization imposing constraints. After optimization and detection of the critical spot with maximal damage as in approach 2 is finished, we add a constraint on the stresses and damage at this spot.

Approach 1 guarantees the absence of the critical spots, but target value and gradient are extremely expensive to calculate, hence optimization process is very slow and Pareto front approximation and navigation are impossible due to the great number of criteria. Approach 2 is faster, but it changes the number of hot spots thus making multi-criteria optimization impossible again. Approach 3 is the fastest and the most flexible, since it keeps the number of criteria constant and increases the number of considered spots only if necessary. Thus multi-criteria optimization described is possible.

5.2 Component A

The component in Fig. 5.1 has four attachments. It is fixed in the central attachment and three actuators are installed in the remaining ones. Artificial loading signals, generated by the three actuators with randomly varying alignment, are used to compute reference damage field as in Section 2.1.7. The mesh contains 17256 quadrangles eight of which are selected as hot spots.

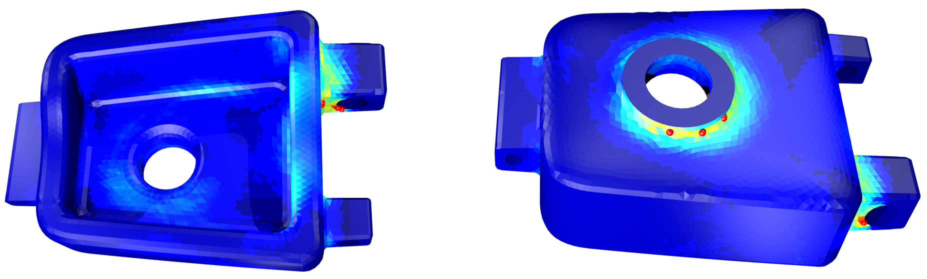


Figure 5.1: Damage field and hot spots (red)

The computation of the reference damages and optimal damages has to employ the same model and parameters. In the current study we use the one-slope damage curve $g(\sigma)$ with paramaters $k_1 = k_2 = 5$, $\sigma_s = 80$ and $N_s = 10^6$ in Eq. (5.5) and the Goodman correction $h(\sigma_1, \sigma_2)$, defined by Eq. (5.6), with $M = 0.3$.

5.2.1 Powers and cycles

The more optimization parameters are introduced, the larger the feasible set is, the better solution we should expect, although the use of all available variables requires longer computational time, not to mention the more complicated implementation. The importance of particular state variables is checked numerically. Fig. 5.2 shows that when the actuator alignment parameters $\boldsymbol{\tau}_i$ are fixed, there is no significant difference, whether we optimize only with respect to applied forces $\boldsymbol{p}_{1,b}$ and $\boldsymbol{p}_{2,b}$ with number of cycles in each block $n_b = n^{\max}$ or whether we also vary n_b. As can be seen in Fig. 5.2, optimal values become noticeably lower when we include alignment parameters into consideration.

The target function monotonically decreases with respect to the number of blocks B in the loading signal. A great number of blocks makes both the inner (5.2) and the outer (5.1)–(5.4) optimization problems more computationally expensive. Therefore, we want to keep B as small as possible.

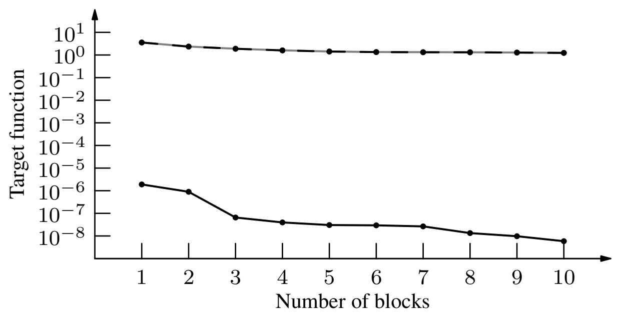


Figure 5.2: Target function values for different blocks: optimized applied forces (gray), applied forces and cycles (dashed), applied forces, cycles and alignment (dashed).

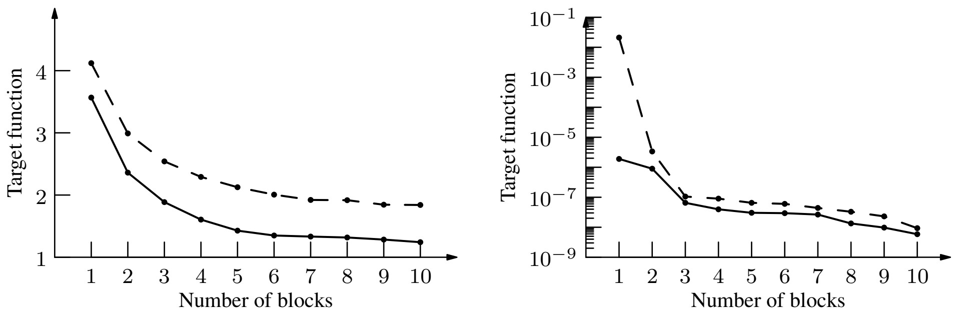
The target function does not change significantly when $B > 3$. Although this behavior might depend on reference signal and damage distribution and component geometry, we recommend using three blocks in the load signal as a starting point of the test signal design.

5.2.2 Mean stress correction

The damage model uses stress signals with arbitrary amplitude and mean in each block, which extends our search space, but involves correction function and doubles the number of applied force variables. We want to check whether it is possible to achieve good approximation with fully reversed signal, i.e. when $\boldsymbol{p}_{1,b} = -\boldsymbol{p}_{2,b}$. In this case properties of the mean stress correction function imply that $h(\sigma_1, -\sigma_1) = |\sigma_1|$ and hence damage computation Eq. (5.2) becomes:

$$d_k(\boldsymbol{z}) = \max_{\alpha_k} \sum_{b=1}^{B} n_b g(|\boldsymbol{c}(\alpha_k)^T \sum_{i=1}^{A} p_{bi}\, \mathcal{B}_{ki}\, \boldsymbol{l}(\boldsymbol{\tau}_i)|),$$

Consequently, the inner optimization problem becomes easier to solve, since it has less local maxima (see Section 4.2.1 for details).



(a) Optimized applied forces with mean stress correction (solid) and without (dashed).

(b) Optimized applied forces, cycles and alignment with mean stress correction (solid) and without (dashed).

Figure 5.3: Target function values with and without mean stress correction.

From Fig. 5.3 it can be seen that the target values increased by 16%–49% in case of fixed alignment of the actuators (see Fig. 5.3a) and by 59%–276% in general case when $B > 1$ (see Fig. 5.3b), compared to arbitrary mean stress loading. This clearly indicates the necessity of using the mean stress correction methods. The number of cycles in a block does not have much influence on the results.

5.2.3 Forces and moments

Here we want to check numerically if the moments applied at the attachments affect the quality of the damage approximation. Therefore, we optimize two alignment models. One has alignment parameters $\boldsymbol{\tau} = [r, \phi, \theta, \phi', \theta']$ and varies both moments and forces as in Example 2.2.9. Another varies only forces, i.e. $r = 1$, $\phi = \phi'$ and $\theta = \theta'$.

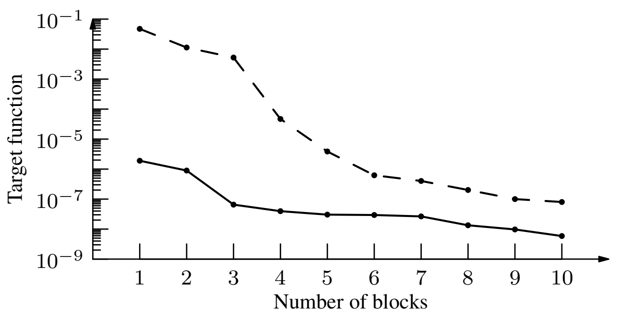


Figure 5.4: Target function values: optimized forces and moments (solid), only forces (dashed).

Examining the results of the optimization (Fig. 5.4) it can be seen that including moments in the model makes the target values at least 10 times smaller and hence the damage approximation at the hot spots becomes much better.

5.2.4 Relative error

Up to now, we have studied only values of the target function, but we are more interested in deviation of achieved damage d_k from the reference d_k^{ref} at every hot spot k.

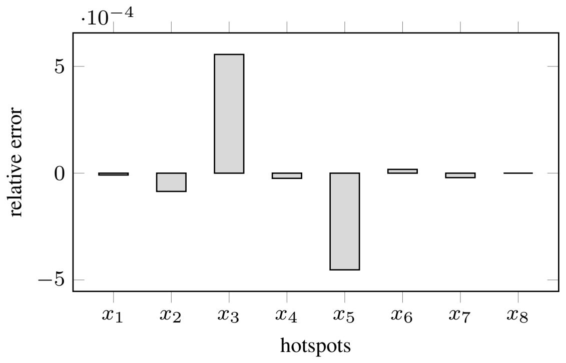
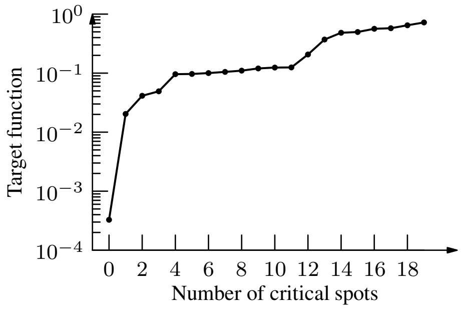


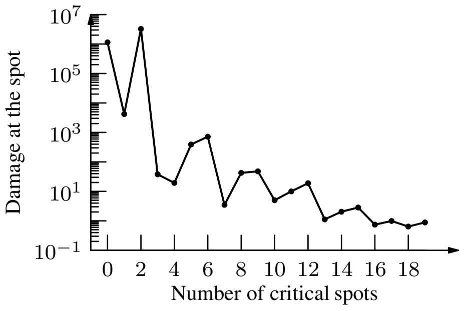
Figure 5.5: Relative damage error $d/d^{\mathrm{ref}} - 1$ at the hot spots

Let us have a close look at the relative damage error $(d - d^{\mathrm{ref}})/d^{\mathrm{ref}}$. Solution of the optimization problem (5.1)–(5.4) with $B = 3$ and obtained minimal value $6.5 \cdot 10^{-08}$ is shown in Fig. 5.5. Relative error at the hot spots does not exceed 0.06%, which can be considered a very good approximation. Hence, the proposed model proved to find an appropriate test rig design.

The reference damage values were approximated at only 8 hot spots. Meanwhile, the component can fail at one of the 17458 spots that were not considered in the test rig problem. The critical spot with the damage, exceeding 10^6, was found after the entire damage field was calculated. This indicates that the adaptive optimization, discussed in Section 5.1.4, has to be applied.



(a) Target function



(b) Damage at critical spots

Figure 5.6: Adaptive optimization imposing constraints

Fig. 5.6a shows that the target function monotonically increases with the number of critical spots. Such behavior can be explained by the fact that each new critical spot adds a new constraint and, hence, decreases the feasible set. From Fig. 5.6b it can be seen that 19 constraints were sufficient to make the damage at all 17256 elements of the mesh below the specified threshold. This happens due to the continuity of the stress vector field of the component. Convergence of the maximal damage in Fig. 5.6b is not monotonic, since the test rig parameters can be adjusted in such a way that the damage decreases at the already added critical spots, but increases at some other spot. The geometry of the component affects such convergence.

The final result of the adaptive optimization is shown in Fig. 5.7. The minimal value equals 0.883658. This is considerably higher, compared to the result in Fig. 5.5, but it has a physical meaning, since the damage was approximated at the hot spots and the component did not fail anywhere else.

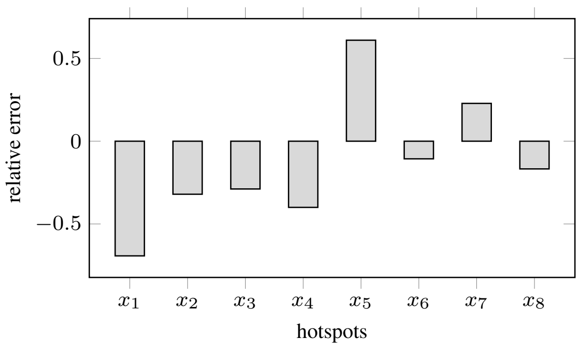


Figure 5.7: Relative damage error $d/d^{\mathrm{ref}} - 1$ at the hot spots

5.3 Component B

The component in Fig. 5.8 has seven attachments. It is fixed in the central attachment and three actuators are installed. The mesh contains 21105 polygons ten of which are selected as hot spots. In this section we use exactly the same methodology to consider Component B as was used for Component A.

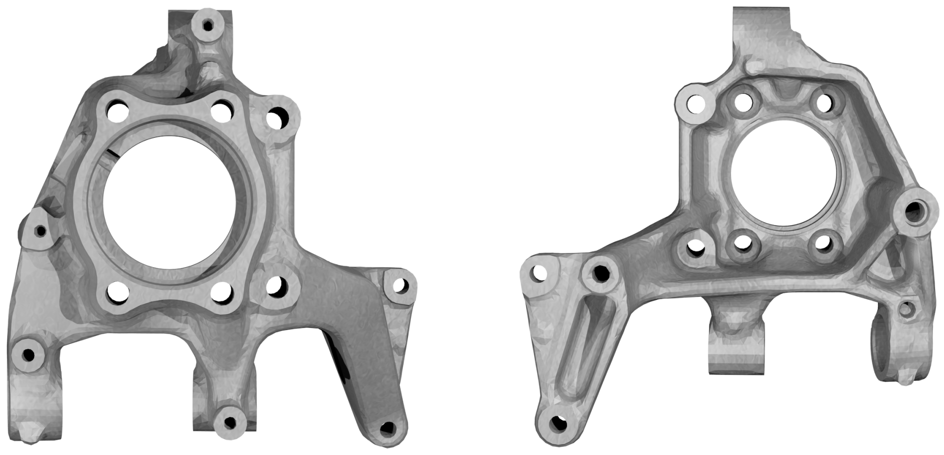


Figure 5.8: Geometry of the component

5.3.1 Powers and cycles

The importance of particular state variables is checked numerically. Fig. 5.9 shows that optimal values become noticeably lower when we include alignment parameters into consideration. Also, there is no significant improvement, when the number of cycles in each block are optimized.

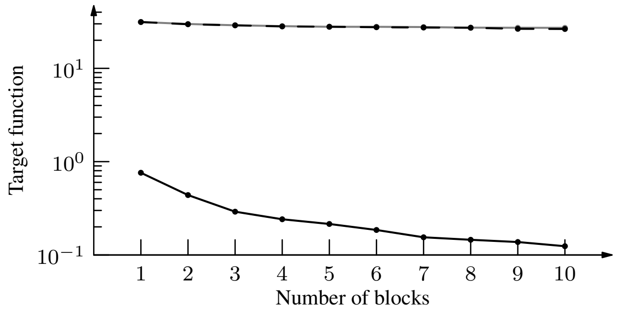
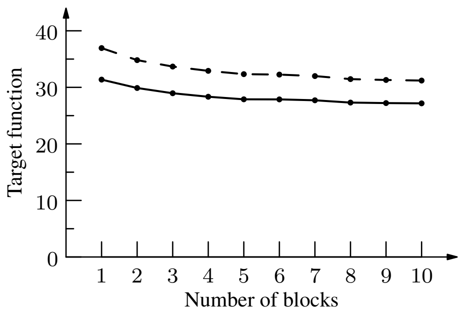


Figure 5.9: Target function values for different blocks: optimized applied forces (gray), applied forces and cycles (dashed), applied forces, cycles and alignment (dashed).

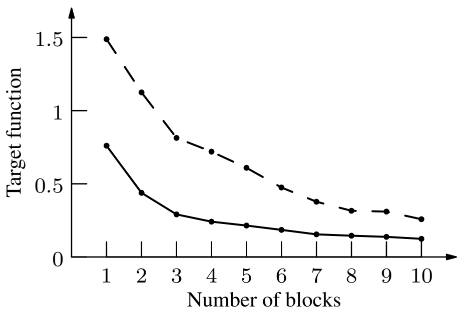
The target function monotonically decreases with respect to the number of blocks B in the loading signal, but it does not change significantly when $B > 3$. Therefore, the behavior of the minimum for Component B in Fig. 5.9 fully coincides in a qualitative sense with the one of Component A in Fig. 5.2.

5.3.2 Mean stress correction

From Fig. 5.10 it can be seen that the target values increased by 14.8%–17.8% in case of fixed alignment of the actuators (see Fig. 5.10a) and by 95.7%–198.3% in general case when $B > 1$ (see Fig. 5.10b), compared to arbitrary mean stress loading.



(a) Optimized applied forces with mean stress correction (solid) and without (dashed).



(b) Optimized applied forces, cycles and alignment with mean stress correction (solid) and without (dashed).

Figure 5.10: Target function values with and without mean stress correction.

As for Component A, it can be concluded that the use of the mean stress correction methods and, hence, general loading is necessary to achieve good approximation. The number of cycles in a block does not have much influence on the results.

5.3.3 Forces and moments

The dependency of the damage approximation quality, when the moments are applied at the attachments, is checked in this section. Fig. 5.11 shows that including moments in the model makes the target values at least 2 times smaller and hence the damage approximation at the hot spots becomes much better.

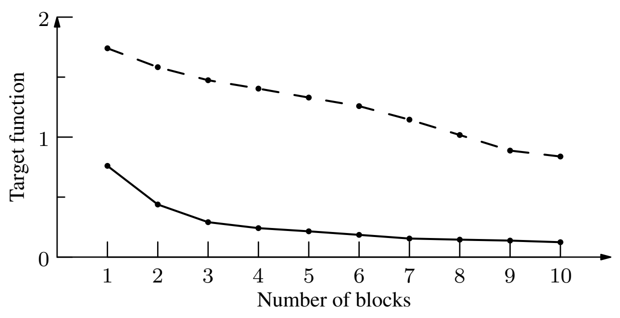


Figure 5.11: Target function values: optimized forces and moments (solid), only forces (dashed).

5.3.4 Relative error

Solution of the optimization problem (5.1)–(5.4) with $B = 3$ and obtained minimal value 0.16 is shown in Fig. 5.12. Relative error at $\boldsymbol{x}_7$ exceeds 130%. Since the damage was overestimated, it is acceptable for the application.

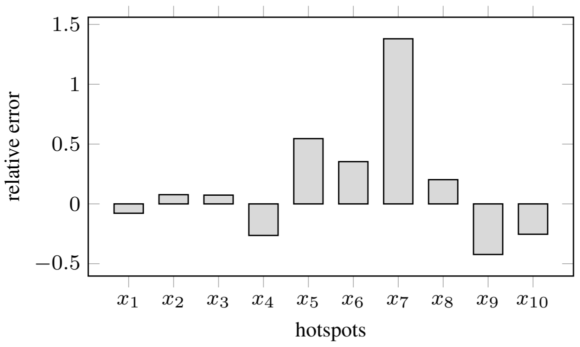


Figure 5.12: Relative damage error $d/d^{\text{ref}} - 1$ at the hot spots

The reference damage values were approximated at only 10 hot spots, but the component can fail at one of the other 21095 spots. The calculation of the entire damage field shows that there are no critical spots with the damage exceeding the specified threshold. This indicates that the adaptive optimization, discussed in Section 5.1.4, is not needed. From Fig. 5.12 it can be seen that the approximation quality for Component B is comparable to the final result for Component A in Fig. 5.7.

Chapter 6

Conclusion

This thesis substantially contributes to designing good test rigs for fatigue life. In this study, we have presented an optimization problem that allows to find the best design of the test rig, capable of approximating fatigue damages at the hot spots generated by the block load signals. Mathematically this test rig design is a multi-criteria global bilevel continuous minimization problem, that can be solved using derivative-based methods such as multi level single linkage. The main contributions of this thesis concern:

- overview, classification of fatigue damage calculation and derivation of the mathematical program that allows to find an optimal design of the test rig;

- theoretical and numerical analysis of a special case of a test rig with one hot spot and a single actuator, generating a constant fully reversed loading signal; test rig model validation, considering solutions for two-dimensional specimens;

- theoretical analysis of the single level formulation of the test rig problem (SLTR); proof of the damage function convexity with respect to applied forces, if the alignment parameters of each actuator are fixed; DC decomposition of the target functional; representation of the non-convex SLTR as a union of convex optimization problems on polyhedral cones, generated by hyperplanes, passing through origin;

- efficient recursive algorithm for defining representations and interior points of polyhedral cones, generated by hyperplanes, passing through origin; proof of the polynomial worst case complexity with respect to the number of hyperplanes; CPU time benchmark;

- theoretical analysis of the bilevel formulation of the test rig problem (BLTR); proof of the damage function convexity with respect to applied forces, if the alignment parameters of each actuator are fixed;

DC decomposition of the target functional; regularity properties of the BLTR;

- new branch-and-bound algorithm that exploits local maximum points of every summand to maximize the sum efficiently. Convergence proof and application to the critical plane problem, compared to DIRECT algorithm.
- worst case relative error for the discretized solution of the critical plane problem, commonly applied in industry;
- numerical solution of the BLTR for two specimens.

Through extensive numerical studies we may draw several conclusions:

- optimization with respect to alignment parameters of the actuators is crucial for the quality of the damage approximation
- introducing moments, applied at the attachments, dramatically improves test rig design, which means that torques play an important role and thus the installation of extenders at the attachments is recommended.
- numbers of stress-strain cycles in each block of loading signal can be fixed at the maximal allowed level without significant change in the overall result
- fully reversed block loading does not provide acceptable quality of approximation. Thus arbitrary mean block loading and stress correction methods such as Goodman or Gerber curves have to be used.
- load signals should have at least three blocks to achieve appropriate results

Further studies may consider robustness of the problem, sensitivity analysis, different models of damage computation and more real-world examples.

Bibliography

[1] Agmon, S. The relaxation method for linear inequalities. *Can. J. of Math. 6* (1954), 382–392.

[2] ASTM Standard E1823. *Standard Terminology Relating to Fatigue and Fracture.* ASTM Int., West Conshohocken, PA, 2000.

[3] Balakrishnan, V., Boyd, S., and Balemi, S. Branch and bound algorithm for computing the minimum stability degree of parameter-dependent linear systems. *International Journal of Robust and Nonlinear Control 1*, 4 (1991), 295–317.

[4] Basquin, O. H. The exponential law of endurance tests. *ASTM 10* (1910), 625–630.

[5] Bazaraa, M., Sherali, H., and Shetty, C. *Nonlinear Programming: Theory and Algorithms.* Wiley, 2013.

[6] Boender, C. G. E., Kan, A. H. G. R., Timmer, G. T., and Stougie, L. A stochastic method for global optimization. *Math. Program. 22*, 1 (1982), 125–140.

[7] Boyd, S., and Vandenberghe, L. *Convex Optimization.* Berichte über Verteilte Messsysteme. Cambridge University Press, 2004.

[8] Chubanov, S. A polynomial projection algorithm for linear programming. *Optim. Online. www.optimization-online.org/DBHTML/2013/07/3948.html* (2013). Accessed 7 July 2013.

[9] Cormen, T., Leiserson, C., Rivest, R., and Stein, C. *Introduction To Algorithms.* MIT Press, 2001.

[10] Endo, T., Mitsunaga, K., and Nakagawa, H. Fatigue of metals subjected to varying stress - prediction of fatigue lives. *Jpn. Soc. of Mech. Eng.* (1967), 41–44.

[11] Endo, T., Mitsunaga, K., Nakagawa, H., and Ikeda, K. Fatigue of metals subjected to varying stress - low cycle, middle cycles fatigue. *Prelim. Proc. of The Chukogu-Shikoku Dist. Meet. The Jpn Soc. of Mech. Eng.* (1967), 45–48.

[12] Fuchs, H. O. A set of fatigue failure criteria. *J. Basic Eng.* (1965).

[13] Garud, Y. S. Multiaxial fatigue: a survey of the state of the art. *J. Testing Eval. 40* (1981), 165–178.

[14] Golub, G. H., and Loan, C. F. V. *Matrix Computations.* JHU Press, Baltimore, Maryland, 2013.

[15] Gough, H. J., Aeronautical Research Council (Great Britain), Pollard, H. V., and Clenshaw, W. J. *Some Experiments on the Resistance of Metals to Fatigue Under Combined Stresses.* Aeronautical Research Council Reports and Memoranda. H.M. Stationery Office, 1951.

[16] Hecht, F. New development in FreeFem++. *J. Numer. Math. 20*, 3-4 (2012), 251–265.

[17] Ho, C., and Zimmerman, S. On the number of regions in an m-dimensional space cut by n hyperplanes. *Gaz. of the Aust. Math. Soc. 33*, 4 (2006).

[18] Horst, R., and Thoai, N. V. DC Programming Overview. *J. of Optim. Theory and Appl. 103*, 1 (1999), 1–43.

[19] Johannesson, P., and Speckert, M. *Guide to Load Analysis for Durability in Vehicle Engineering.* Automotive Series. Wiley, 2013.

[20] Johnson, S. G. The NLopt nonlinear-optimization package.

[21] Jones, D. R., Perttunen, C. D., and Stuckman, B. E. Lipschitzian Optimization Without the Lipschitz Constant. *J. Optim. Theory Appl. 79*, 1 (1993), 157–181.

[22] Karmarkar, N. K. A new polynomial-time algorithm for linear programming. *Comb. 4* (1984), 373–395.

[23] Khachiyan, L. G. A polynomial algorithm in linear programming. *Doklady Akademiia Nauk SSSR 224* (1979), 1093–1096.

[24] Klee, V., and Minty, G. J. How good is the simplex algorithm? *Inequal. 3* (1972), 159–175.

[25] Le Thi Hoai, A., and Pham Dinh, T. The DC (Difference of Convex Functions) Programming and DCA Revisited with DC Models of Real World nonconvex optimization problems. *Ann. of Oper. Res. 133* (2005), 23–46.

[26] Lee, Y., Barkey, M., and Kang, H. *Metal Fatigue Analysis Handbook: Practical Problem-solving Techniques for Computer-aided Engineering.* Butterworth-Heinemann, 2011.

[27] McDowell, D., and Ellis, R. *Advances in Multiaxial Fatigue.* No. no. 1191 in ASTM STP 1191. ASTM, 1993.

[28] Miettinen, K. *Nonlinear Multiobjective Optimization.* International series in operations research and management science. Kluwer Academic Publishers, 1999.

[29] Miner, M. A. Cumulative damage in fatigue. *J. of Appl. Mech. 12* (1945), A159–A164.

[30] Motzkin, T. S., and Schoenberg, I. J. The relaxation method for linear inequalities. *Can. J. of Math. 6* (1954), 393–404.

[31] Narula, G., Narula, K., and Gupta, V. *Materials Science.* Tata McGraw-Hill, 1988.

[32] Nelson, D. V., and Fuchs, H. O. Predictions of cumulative fatigue damage using condensed load histories. *SAE Int.* (1975).

[33] Orlov, A. V., and Strekalovskii, A. S. Numerical search for equilibria in bimatrix games. *Zh. Vychisl. Mat. Mat. Fiz. 45*, 6 (2005), 983–997. In Russian.

[34] Palmgren, A. Die Lebensdauer von Kugellagern. *Zeitschrift des Vereins Deutscher Ingenieure 68* (1924), 339–341.

[35] Parry, R. *Mohr Circles, Stress Paths and Geotechnics.* Taylor & Francis, 2004.

[36] Pshenichny, B. N. *Convex Analysis and Extremal Problems.* Nauka, 1980. In Russian.

[37] Roos, K. Speeding up Chubanov's method for solving a homogeneous inequality system. *Optim. Online.* *http://www.optimization-online.org/DB_HTML/2013/09/4049.html* (2013). Accessed 20 September 2013.

[38] Shubert, B. O. A Sequential Method Seeking the Global Maximum of a Function. *SIAM J. Numer. Anal. 9*, 3 (1982), 379–388.

[39] Simeon, B. *Computational Flexible Multibody Dynamics: A Differential-Algebraic Approach.* Differential-Algebraic Equations Forum. Springer, 2013.

[40] Stephens, R. I., Fatemi, A., Stephens, R. R., and Fuchs, H. O. *Metal Fatigue in Engineering.* Wiley Interscience, New York, 2001.

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

[42] You, B.-R., and Lee, S.-B. A critical review on multiaxial fatigue assessments of metals. *Int. J. Fatigue 18* (1996), 235–244.

Curriculum Vitae

Alexander Belyaev

Born in Vladivostok, Russian Federation

School education

Mathematics in
Far Eastern State University

Bachelor of Mathematics

Mathematics in
University of Kaiserslautern

Master of Mathematics

Doctorate studies in
University of Kaiserslautern

Publications

The following publications contain parts of this thesis or precursory work:

- Belyaev, A., Maag, V., Küfer K.-H. Test Rig Optimization and Block Loads. In *Proceedings of the International Conference "Numerical Computations: Theory and Algorithms"*, page 49, 2013.
- Belyaev, A. Inductive generation of interior points and polyhedral representations of cones in R^N cut by M planes sharing a common point. *Mathematical Methods of Operations Research*, 2014. (submitted)
- Belyaev, A., Maag, V., Speckert, M., Obermayr, M., Küfer, K.-H. Adaptive multi-criteria test rig optimization. In *Engineering Structures*, 2014. (submitted)

Lebenslauf

Alexander Belyaev

Geboren in Wladiwostok, Russland

Schulausbildung

Studium der Mathematik an der Fernöstlichen Staatlichen Universität

Bachelor of Mathematics

Studium der Mathematik an der Technischen Universität Kaiserslautern

Master of Mathematics

Doktorand der Mathematik an der Technischen Universität Kaiserslautern

Veröffentlichungen

Die folgenden Veröffentlichungen stehen im Zusammenhang mit dieser Arbeit:

- Belyaev, A., Maag, V., Küfer K.-H. Test Rig Optimization and Block Loads. In *Proceedings of the International Conference "Numerical Computations: Theory and Algorithms"*, page 49, 2013.
- Belyaev, A. Inductive generation of interior points and polyhedral representations of cones in R^N cut by M planes sharing a common point. *Mathematical Methods of Operations Research*, 2014. (submitted)
- Belyaev, A., Maag, V., Speckert, M., Obermayr, M., Küfer, K.-H. Adaptive multi-criteria test rig optimization. In *Engineering Structures*, 2014. (submitted)