Universitat Politecnica de Catalunya
Departament de Matematica Aplicada III

Programa de Doctorat de Matematica Aplicada

Decomposition Techniques For Computational Limit Analysis

Author:
Nima Rabiei

Supervisor:
Pro.Jose J.Muñoz

PhD Thesis
Barcelona,September 22,
2014

ACKNOWLEDGEMENTS

This thesis becomes a reality with the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

First of all, I would like to extend my special gratitude and thanks to my supervisor, professor **Jose J Muñoz** for tremendous support and imparting his knowledge and expertise in this study.

I would like to express my deepest gratitude towards my beloved and supportive wife, **Bahar** who is always by my side when times I needed her most and helped me a lot in making this study. Also I would like to express my gratitude to my supportive brother-in-law, Bijan Ebrahimi.

Deepest gratitude is also due to my parents for the encouragement and endless love which helped me in completion of this thesis. Special thanks also go to my sister and brothers who shared the burden with me.

I am highly in indebted to the **Technical University of Catalonia,** department of **Applied Mathematics III**, the research group **LaCàN** for giving me all facilities and work environment required for my study. Finally, My thanks and appreciation also go to my colleagues Arjuna Castrillon, Nina Asadipour and people who have willingly helped me out with their abilities.

ABSTRACT

Limit analysis is relevant in many practical engineering areas such as the design of mechanical structure or the analysis of soil mechanics. The theory of limit analysis assumes a rigid, perfectly-plastic material to model the collapse of a solid that is subjected to a static load distribution.

Within this context, the problem of limit analysis is to consider a continuum that is subjected to a fixed force distribution consisting of both volume and surfaces loads. Then the objective is to obtain the maximum multiple of this force distribution that causes the collapse of the body. This multiple is usually called collapse multiplier. This collapse multiplier can be obtained analytically by solving an infinite dimensional nonlinear optimisation problem. Thus the computation of the multiplier requires two steps, the first step is to discretise its corresponding analytical problem by the introduction of finite dimensional spaces and the second step is to solve a nonlinear optimisation problem, which represents the major difficulty and challenge in the numerical solution process.

Solving this optimisation problem, which may become very large and computationally expensive in three dimensional problems, is the second important step. Recent techniques have allowed scientists to determine upper and lower bounds of the load factor under which the structure will collapse. Despite the attractiveness of these results, their application to practical examples is still hampered by the size of the resulting optimisation process. Thus a remedy to this is the use of decomposition methods and to parallelise

the corresponding optimisation problem.

The aim of this work is to present a decomposition technique which can reduce the memory requirements and computational cost of this type of problems. For this purpose, we exploit the important feature of the underlying optimisation problem: the objective function contains one scaler variable λ. The main contributes of the thesis are, rewriting the constraints of the problem as the intersection of appropriate sets, and proposing efficient algorithmic strategies to iteratively solve the decomposition algorithm.

Contents

List of Figures

List of Tables

1 Introduction

Limit analysis aims to directly determine the collapse load of a given structural model without resorting to iterative and incremental analysis. Computational techniques in limit analysis are based on the so-called limit theorems, which are based on the minimization of the dissipation energy and the maximization of the load factor, and they furnish lower and upper bounds of the collapse load [14]. Assuming a rigid-perfectly plastic solid subject to static load distribution, the problem of limit analysis consists in finding the maximum multiple of this load distribution that will cause the collapse of the body. As it will be explained in Section 1.1, the analytical load factor results from solving an infinite dimensional saddle point problem, where the internal work rate is minimized over the linear space of kinematically admissible velocities for which the external work rate equals unity. Then load factor be also obtained by the maximum load over an admissible set of stresses in equilibrium with the applied loads [33, 34, 37].

The aim of this work is to first present a general methodology to decompose optimisation problems, and to apply this methodology to the optimisation problems encountered in limit analysis.

The second part is to propose a decomposition technique which can alleviate the memory requirements and computational cost of this type of problems.

This work has been motivated by the computational cost of the optimisation program in practical applications. It has been found that the memory requirements and CPU time of the up-to-date available software to solve optimisation problems, such as MOSEK[2], SDPT3[46], SeDuMi[45] or specific oriented software [32] are still not affordable if we want to analyse other

than academical problems. Then using decomposition methods seems an appealing technique for these analyses. For instance, Table 1.1 and Figure 1.1 show the number of elements and corresponding number of degrees of freedom (dof) for the lower bound (LB)[1] and upper bound (UB) problem, in two and three dimensions. As it can be observed, the number of dof in three dimensions is always higher than in two dimensions for similar number of elements, and becomes prohibitive for not so large meshes.

One of the possible solution is to parallelise the solution of the systems of equations, which is inherent in all optimisation process. Although this venue may alleviate the CPU time of the resolution process, the memory requirements may still remain too large. For this reason, we propose to partition *ab initio* the domain of the structure, and solve the optimisation process in a decomposed manner. In doing this, there is no need to solve nor to store the system of equations of the optimisation problem for the full domain.

In this Chapter, we introduce limit analysis of structures and briefly describe discrete forms that give rise to the lower bound optimisation problem.

1.1. Optimisation Problems in Limit Analysis

Let Ω denote the domain of a body assumed to be made of a rigid-perfectly plastic material, subjected to load volumetric load $\lambda \boldsymbol{f}$(gravity), with λ an unknown load factor to be determined. Its boundary $\partial\Omega$, consists of a Neumann part Γ^N and a Dirichlet part Γ^D, which are such that $\partial\Omega = \Gamma^N \cup \Gamma^D$ and $\Gamma^N \cap \Gamma^D = \emptyset$.

The body velocities are equal to zero at the Dirichlet boundary, while the Neumann boundary is subjected to the traction field $\lambda \boldsymbol{g}$, see Figure 1.2.

The objective of limit analysis is to compute the maximum value λ^* of the load factor at which the structure will collapse, and if possible, the velocity

[1]Discretisation of the problem in a particular fashion, i.e. combination of appropriately selected interpolations for both the stresses and velocities results in estimating of the maximum multiple of the load factor that causes the collapse of the body, either from below or from above. The optimisation problem that approaches to the solution from below is so-called Lower Bound (LB) problem and also the optimisation problem which approaches to the solution from above is called Upper Bound (UB) problem [34, 37].

2D		
# elements	dof(LB)	dof(UB)
267	3309	4005
321	3989	4861
408	5097	6241
598	7505	9265
1723	21629	27205
3110	39033	49369
5644	70945	89897
9283	84637	111665
24654	223588	296398

3D		
# elements	dof(LB)	dof(UB)
542	15719	25619
679	3433	2848
1144	32033	50597
3089	89582	154772
4185	100441	95995
8730	211672	360163
24133	579193	565801

Table 1.1.: Number of elements and corresponding number of degrees of freedom (dof) in two- and three-dimensional analysis.

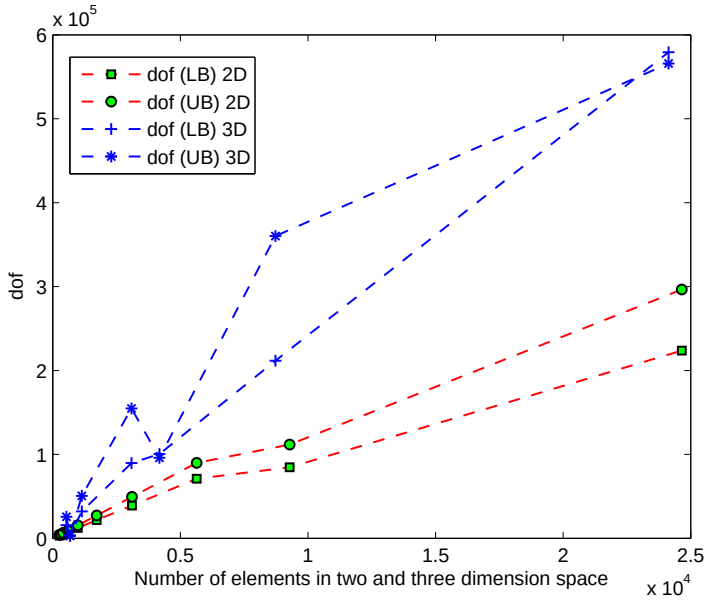


Figure 1.1.: Number of elements and corresponding number of degrees of freedom (dof) in two- and three-dimensional analysis

field $\boldsymbol{u}^*$ and tensor stress field $\boldsymbol{\sigma}^*$, which allow us to identify the collapse mechanism. The admissibility condition for the stress field is expressed by the membership condition $\boldsymbol{\sigma} \in \mathcal{B}$, where the set $\mathcal{B}$ depends on the plastic

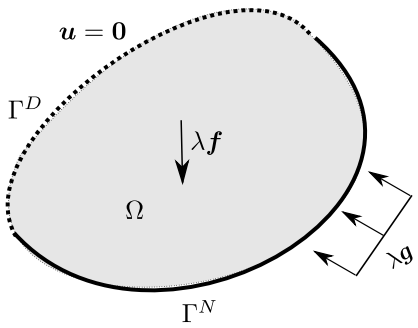


Figure 1.2.: Illustration of Neumann and Dirichlet parts of the domain Ω.

criteria adopted, and will be defined by the following general form:

$$\mathcal{B} := \{\boldsymbol{\sigma} | q(\boldsymbol{\sigma}) \leq 0\}.$$

The work rate of external loads associated with a velocity $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x})$ is given by the following linear functional:

$$L(\boldsymbol{u}) = \int_\Omega \boldsymbol{f} \cdot \boldsymbol{u} \, d\Omega + \int_{\Gamma^N} \boldsymbol{g} \cdot \boldsymbol{u} \, d\Gamma.$$

The velocity belongs to an appropriate space Y, to be specified in Subsection 1.1.1.

The work rate of the symmetric stress field $\boldsymbol{\sigma}$ associated with $\boldsymbol{u}$ is given by the bilinear form:

$$\begin{aligned} a(\boldsymbol{\sigma}, \boldsymbol{u}) &= \int_\Omega \boldsymbol{\sigma} : \boldsymbol{\varepsilon}(\boldsymbol{u}) \, d\Omega + \int_\Gamma [\![\boldsymbol{u}]\!] \cdot \boldsymbol{\sigma} \boldsymbol{n} \, d\Gamma \\ &= \int_\Omega \boldsymbol{\sigma} : \boldsymbol{\varepsilon}(\boldsymbol{u}) \, d\Omega + \int_\Gamma [\![\boldsymbol{u}]\!] \bar{\otimes} \boldsymbol{n} : \boldsymbol{\sigma} \, d\Gamma, \end{aligned}$$

where $\varepsilon(\boldsymbol{u}) = \frac{1}{2}[(\nabla \boldsymbol{u}) + (\nabla \boldsymbol{u})^T]$, is the symmetric velocity gradient, and the operator $\bar{\otimes}$ is a symmetrized dyadic product: $\boldsymbol{a} \bar{\otimes} \boldsymbol{b} = \frac{1}{2}(\boldsymbol{a} \otimes \boldsymbol{b} + \boldsymbol{b} \otimes \boldsymbol{a})$, and Γ denotes the internal surface of domain Ω where the velocity field is discontinuous. The symbol $[\![\boldsymbol{u}]\!]$, denotes the jump of field $\boldsymbol{u}$ on $d\Gamma$. The rate of dissipated energy $D(\boldsymbol{u})$ is defined as:

$$D(\boldsymbol{u}) = \sup_{\boldsymbol{\sigma} \in \mathcal{B}} a(\boldsymbol{\sigma}, \boldsymbol{u}).$$

With these definitions at hand the upper and lower bound theorems may be stated as follows:

1.1.1. Lower Bound Theorem

If for a given load factor $\bar{\lambda}$ the stress field is in static equilibrium, i.e. at all points of the domain Ω

$$a(\boldsymbol{\sigma}, \boldsymbol{u}) = \bar{\lambda} L(\boldsymbol{u}) \quad \forall \boldsymbol{u} \in Y,$$

and $\boldsymbol{\sigma}$ satisfies the Neumann boundary condition i.e. $\boldsymbol{\sigma n} = \lambda \boldsymbol{g}$, and the admissibility condition $\boldsymbol{\sigma} \in \mathcal{B}$, thus $\bar{\lambda}$ will not be larger than the optimal load factor λ^* [14]. The set Y is the set of (not necessarily continuous) velocities such that the integrals in the expressions of $a(\boldsymbol{\sigma}, \boldsymbol{u})$ and $L(\boldsymbol{u})$ remain bounded [15].

1.1.2. Upper Bound Theorem

A load factor that equalizes the rate of dissipated energy $D(\boldsymbol{u})$ to the external work rate $L(\boldsymbol{u})$ with a velocity field $\boldsymbol{u}$ that is kinetically admissible[14, 15] i.e. satisfies the Dirichlet boundary condition and associative law, will not be less than the optimal load factor λ^*. The associative law imposes that $D(\boldsymbol{u}) < \infty$, i.e.

$$\begin{aligned} \boldsymbol{\varepsilon}(\boldsymbol{u}) &\in \partial f(\boldsymbol{\sigma}) \\ [\![\boldsymbol{u}]\!] \bar{\otimes} \boldsymbol{n} &\in \partial f(\boldsymbol{\sigma}), \end{aligned}$$

where $\partial f(\boldsymbol{\sigma})$ is the subgradient of f at $\boldsymbol{\sigma}$, defined as

$$\partial f(\boldsymbol{\sigma}) = \{\boldsymbol{d} | (\boldsymbol{\sigma} - \boldsymbol{\sigma}^*) \cdot \boldsymbol{d} \geq f(\boldsymbol{\sigma}) - f(\boldsymbol{\sigma}^*) \; \forall \, \boldsymbol{\sigma}^*\}.$$

1.1.3. Saddle Point Problem

The lower and upper bound theorems of limit analysis allow us to compute the optimal load factor as two different optimisation problems:

$$\begin{aligned} LB: \; \underline{\lambda} &= \sup_{\substack{a(\boldsymbol{\sigma},\lambda)=\lambda L(\boldsymbol{u}),\, \forall \boldsymbol{u} \in Y \\ \boldsymbol{\sigma} \in \mathcal{B}}} \lambda = \sup_{\lambda, \boldsymbol{\sigma} \in \mathcal{B}} \inf_{\boldsymbol{u} \in Y} (\lambda + a(\boldsymbol{\sigma}, \boldsymbol{u}) - \lambda L(\boldsymbol{u})) \\ &= \sup_{\lambda, \boldsymbol{\sigma} \in \mathcal{B}} \inf_{\boldsymbol{u} \in Y} (a(\boldsymbol{\sigma}, \boldsymbol{u}) + \lambda (1 - L(\boldsymbol{u}))) \\ &= \sup_{\boldsymbol{\sigma} \in \mathcal{B}} \inf_{\substack{L(\boldsymbol{u})=1 \\ \boldsymbol{u} \in Y}} a(\boldsymbol{\sigma}, \boldsymbol{u}). \end{aligned} \tag{1.1.1}$$

$$UB: \; \overline{\lambda} = \inf_{\substack{D(\boldsymbol{u})=\lambda L(\boldsymbol{u}) \\ \boldsymbol{u} \in Y}} \lambda = \inf_{\substack{L(\boldsymbol{u})=1 \\ \boldsymbol{u} \in Y}} D(u) = \inf_{\substack{L(\boldsymbol{u})=1 \\ \boldsymbol{u} \in Y}} \sup_{\boldsymbol{\sigma} \in \mathcal{B}} a(\boldsymbol{\sigma}, \boldsymbol{u}). \tag{1.1.2}$$

The comparison of the results in (1.1.1) and (1.1.2) shows that the UB problem is the dual of the LB problem. Consequently, due to weak duality we have:

$$\underline{\lambda} = \sup_{\boldsymbol{\sigma}\in\mathcal{B}} \inf_{\substack{L(\boldsymbol{u})=1 \\ \boldsymbol{u}\in Y}} a(\boldsymbol{\sigma},\boldsymbol{u}) \leq \inf_{\substack{L(\boldsymbol{u})=1 \\ \boldsymbol{u}\in Y}} \sup_{\boldsymbol{\sigma}\in\mathcal{B}} a(\boldsymbol{\sigma},\boldsymbol{u}) = \overline{\lambda}.$$

If strong duality holds, the inequality above turns into equality. In this case, we denote $\lambda^* = \underline{\lambda} = \overline{\lambda} = a(\boldsymbol{\sigma}^*, \boldsymbol{u}^*)$, where $\boldsymbol{\sigma}^*$ and $\boldsymbol{u}^*$ are the optimal values of the stress and velocity field at the optimum. In addition, we can obtain bounds of the optimum value λ^* by evaluating the bilinear form $a(\boldsymbol{\sigma}, \boldsymbol{u})$ at nonoptimal fields.

To be more specific, let us assume that the following optimisation problem are computed exactly:

$$\sup_{\boldsymbol{\sigma}\in\mathcal{B}} a(\boldsymbol{\sigma},\boldsymbol{u}) = a(\boldsymbol{\sigma}^*,\boldsymbol{u}),\ \forall \boldsymbol{u}\in Y \text{ s.t } L(\boldsymbol{u}) = 1, \tag{1.1.3}$$

$$\inf_{\substack{\boldsymbol{u}\in Y \\ L(\boldsymbol{u})=1}} a(\boldsymbol{\sigma},\boldsymbol{u}) = a(\boldsymbol{\sigma},\boldsymbol{u}^*),\ \forall\, \boldsymbol{\sigma}\in\mathcal{B}. \tag{1.1.4}$$

In terms of equations (1.1.3) and (1.1.4) we have:

$$\lambda_- = a(\boldsymbol{\sigma},\boldsymbol{u}^*) \leq a(\boldsymbol{\sigma}^*,\boldsymbol{u}^*) \leq a(\boldsymbol{\sigma}^*,\boldsymbol{u}) = \lambda^-, \tag{1.1.5}$$

where $\boldsymbol{\sigma}$ is in static equilibrium and $\boldsymbol{\sigma}\in\mathcal{B}$, while $L(\boldsymbol{u}) = 1$ and $\boldsymbol{u}$ satisfies the associative law, that is, the fields $\boldsymbol{\sigma}$ and $\boldsymbol{u}$ are primal and dual feasible, respectively.

1.2. Lower Bound (LB) Problem

1.2.1. The Finite Element Triangulation

When analysing the problem above in plane stress or plane strain, we will consider the following triangular finite element discretisetion. Let τ_h denote the triangulation, where h represent the typical size of the elements. The mesh τ_h consists of ne (number of elements) triangular elements Ω^e that form a partition of the body, such that $\Omega = \cup_{e=1}^{ne}\Omega^e$, with all the element being pairwise disjoint: $\Omega^e \cap \Omega^{e'} = \emptyset\ \forall\, e, e' \in \tau_h$. The boundary of the element Ω^e is denoted by $\partial\Omega^e$. Let ξ be the set of all the edges in the mesh, which is decomposed into the following three disjoint sets: $\xi = \xi^O \cup \xi^D \cup \xi^N$ where ξ^O, ξ^D and ξ^N are sets of interior edges, Dirichlet boundary and

Neumann boundary respectively. Mixed boundary, (edges with Dirichlet and Neumann conditions), are not considered here, but the extension of the problem statement to these situations dose not pose any further complexity.

1.2.2. Discrete Spaces for Lower Bound Problem

We will introduce a set of statically admissible spaces, that is, discrete spaces $\boldsymbol{\sigma}^{LB} \in X^{LB}$ and $\boldsymbol{u}^{LB} \in Y^{LB}$ that preserve the first inequality in (1.1.5).

- X^{LB} : Piecewise linear stress field interpolated from the nodal values $\boldsymbol{\sigma}^{i,e},\ i = 1, \cdots, nn;\ e = 1, \cdots, ne$ (nn is the number of nodes per element, and ne the number of elements). Each element has a distinct set of nodal values, and thus discontinuities at each elemental boundary $\partial\Omega^{e-e'}$(between element e and e') are permitted.
- Y^{LB} : constant velocity $\boldsymbol{\mu}^{e,LB}$ at each element. Additionally, a linear velocity field $\boldsymbol{\nu}^{\xi,LB}$ is introduced at each interior boundary ξ^O and external boundary of ξ^N. We denote by $\boldsymbol{u}^{LB}$ the complete set of velocities, i.e. $\boldsymbol{u}^{LB} = (\boldsymbol{\mu}^{LB}, \boldsymbol{\nu}^{LB})$.

These spaces are depicted in Figure 1.3.

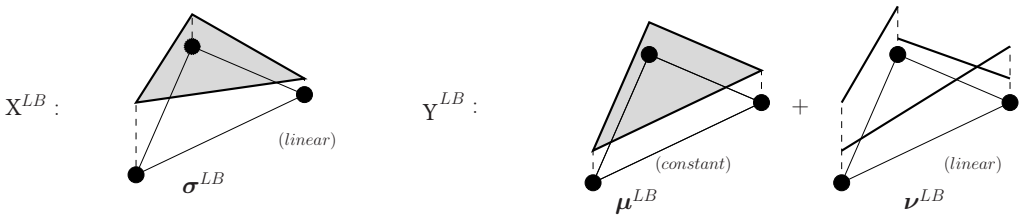


Figure 1.3.: Scheme of the lower bound discrete spaces X^{LB} and Y^{LB} used for the stresses and velocities, respectively. [37]

1.2.3. Implementation

The static equilibrium condition

$$a(\boldsymbol{\sigma}^{LB}; \boldsymbol{u}^{LB}) = \lambda L(\boldsymbol{u}^{LB}), \quad \forall\ \boldsymbol{u}^{LB} \in Y^{LB}, \tag{1.2.1}$$

is rewritten, after using the integration rules for discontinuous functions[1]:

$$\begin{aligned} a(\boldsymbol{\sigma}^{LB};\boldsymbol{\mu}^{LB},\boldsymbol{\nu}^{LB}) &= -\int_{\Omega}\boldsymbol{\mu}^{LB}\cdot(\nabla\cdot\boldsymbol{\sigma}^{LB})\,d\Omega \\ &\quad+\sum_{\partial\Omega^{\xi_e^{e'}}}\int_{\partial\Omega^{\xi_e^{e'}}}\boldsymbol{\nu}^{LB}\cdot[\![\boldsymbol{\sigma}]\!]^{LB}\cdot\boldsymbol{n}\,d\Gamma+\int_{\Gamma_N}\boldsymbol{\nu}^{LB}\cdot\boldsymbol{\sigma}^{LB}\cdot\boldsymbol{n}\,d\Gamma, \\ L(\boldsymbol{\mu}^{LB},\boldsymbol{\nu}^{LB}) &= \int_{\Omega}\boldsymbol{\mu}^{LB}\cdot\boldsymbol{f}\,d\Omega+\int_{\Gamma_N}\boldsymbol{\nu}^{LB}\cdot\boldsymbol{g}\,d\Gamma. \end{aligned} \tag{1.2.2}$$

Consequently, due to the arbitrariness of the constant velocity $\boldsymbol{\mu}^{LB}$ and the linear velocity $\boldsymbol{\nu}^{LB}$ the static equilibrium condition is equivalent to the following equations:

$$\begin{cases} \nabla\cdot\boldsymbol{\sigma}^{LB}+\lambda\boldsymbol{f}=\boldsymbol{0}, & \text{in }\Omega \\ \boldsymbol{\sigma}^{LB}\cdot\boldsymbol{n}=\boldsymbol{0}, & \text{in }\Gamma \\ \boldsymbol{\sigma}^{LB}\cdot\boldsymbol{n}=\lambda\boldsymbol{g}, & \text{in }\Gamma_N \\ \boldsymbol{\sigma}^{LB}\in\mathcal{B}. \end{cases} \tag{1.2.3}$$

The primal and dual LB problems may be obtained by inserting the discrete space X^{LB} and Y^{LB} in primal optimisation problems and using the bilinear and linear forms $a(\boldsymbol{\sigma}^{LB},\boldsymbol{u}^{LB})$ and $L(\boldsymbol{u}^{LB})$. In this case the condition in (1.2.1), is equivalent to the following equations:

$$\begin{cases} \nabla\cdot\boldsymbol{\sigma}^{e,LB}+\lambda\boldsymbol{f}^{e}=\boldsymbol{0} & \text{in }\Omega^{e}\,,\forall\, e=1,\cdots,ne \\ (\boldsymbol{\sigma}^{e,LB}-\boldsymbol{\sigma}^{e',LB})\cdot\boldsymbol{n}^{\xi_e^{e'}}=\boldsymbol{0}, & \forall\,\xi_e^{e'}\in\xi^{O} \\ \boldsymbol{\sigma}^{e,LB}\cdot\boldsymbol{n}^{\xi_e^{N}}=\lambda\boldsymbol{g}^{\xi_e^{N}} & \forall\,\xi_e^{N}\in\xi^{N} \\ \boldsymbol{\sigma}^{e,LB}\in\mathcal{B} & \text{in}\,\Omega^{e},\,\forall\, e=1,\cdots,ne \end{cases} \tag{1.2.4}$$

In Appendix A, it is shown that the equations in (1.2.4) may be transformed into a set of linear constraints. The membership condition be also rewritten, using a change of variable into a Lorentz cone membership. The

[1] We recall that whenever the product of functions fg is discontinuous, we have that, $\int_{\Omega}(f'g+g'f)=\int_{\partial\Omega}fg\,d\Gamma+\sum_{e-e'}\int_{\partial\Omega^{e-e'}}[\![fg]\!]\,d\Gamma$, where the sum is performed on all the boundaries where fg discontinuous.

resulting discrete optimisation problem is deduced in (A.4.9) and recast here:

$$\lambda^{*LB} = \max_{\lambda, \boldsymbol{x}_4, \boldsymbol{x}_{1:3}} \lambda$$

$$\begin{bmatrix} \mathbf{A}^{eq1}\mathbf{P} & \boldsymbol{F}^1 & \mathbf{A}^{eq1}\mathbf{P} \\ \mathbf{A}^{eq2}\mathbf{P} & \mathbf{0} & \mathbf{A}^{eq2}\mathbf{P} \\ \mathbf{A}^{eq3}\mathbf{P} & \boldsymbol{F}^3 & \mathbf{A}^{eq3}\mathbf{P} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{bmatrix} \begin{Bmatrix} \boldsymbol{x}_4 \\ \lambda \\ \boldsymbol{x}_{1:3} \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{b} \end{Bmatrix}, \tag{1.2.5}$$

$$\boldsymbol{x}_4 \text{ is free, } \lambda \geq 0, \ \boldsymbol{x}_{1:3} \in K,$$

$$|\boldsymbol{x}_4| = 3\text{ne}, \ |\lambda| = 1, \ |\boldsymbol{x}_{1:3}| = 9\text{ne},$$

where

$$\boldsymbol{\sigma} = \mathbf{P}\boldsymbol{x}_4 + \mathbf{Q}\boldsymbol{x}_{1:3}. \tag{1.2.6}$$

The first, second, third and fourth rows of the matrix are respectively the equilibrium constraints, inter-element equilibrium equations, boundary Neumann conditions and the membership constraints given in (1.2.3). K is the outer product of Lorentz cones L^n with $n = 3$, that is $K = L_1^n \times L_2^n \times \cdots \times L_r^n$. It follows that K is a convex cone. The matrices appearing in (1.2.5) are explicitly deduced in Appendix A.

The size of the optimisation problem in (1.2.5) as it has been explained, is proportional to the number of elements. If we want to obtain an accurate value of λ^{*LB}, (i.e. value of λ^{*LB} that is close to λ^*), we must increase the number of elements, which causes the increase of the size of the problem. Since our sources like memory are limited, we are not able to increase the number of elements as mush as we want to. Consequently, to solve larger problem, we resort here to decomposition methods.

The development of general decomposition techniques has given rise to numerous approaches, which include Benders decomposition [19, 26], proximal point strategies [13], dual decomposition [10, 25], subgradient and smoothing methods [38, 39], or block decomposition [36], among many others. In the engineering literature, some common methods inherit either decomposition methods for elliptic problems [29], or proximal point strategies [30], or methods that couple the solutions from overlapping domains [41], which reduce their applicability.

The accuracy of dual decomposition and subgradient techniques strongly depend on the step-size control, while the accuracy of proximal point and smoothing techniques depend on the regularisation and smoothing parameters, which are problem dependent and not always easy to choose. Also,

and from the experience of the authors, Benders methods have slow converge rates in non-linear optimisation problems due to the outer-linearisation process. These facts have motivated the development of the method presented here, which is specially suited for nonlinear optimisation, and in particular exploits the structure of the problems encountered in engineering applications. We aim to solve a convex optimisation problems that can be written in the following form:

$$\begin{aligned}
\lambda^* = \max_{\boldsymbol{x}_1, \boldsymbol{x}_2, \lambda} \ & \lambda \\
& \boldsymbol{f}_1(\boldsymbol{x}_1, \lambda) = \boldsymbol{0} && \text{(1.2.7a)} \\
& \boldsymbol{f}_2(\boldsymbol{x}_2, \lambda) = \boldsymbol{0} && \text{(1.2.7b)} \\
& \boldsymbol{g}_1(\boldsymbol{x}_1) + \boldsymbol{g}_2(\boldsymbol{x}_2) = \boldsymbol{0} && \text{(1.2.7c)} \\
& \boldsymbol{x}_1 \in K_1 \subseteq \Re^{n_1},\ \boldsymbol{x}_2 \in K_2 \subseteq \Re^{n_2},\ \lambda \in \Re, && \text{(1.2.7d)}
\end{aligned}$$

where $\boldsymbol{f}_1 : \Re^{n_1} \times \Re \to \Re^{m_1}$, $\boldsymbol{f}_2 : \Re^{n_2} \times \Re \to \Re^{m_2}$, $\boldsymbol{g}_1 : \Re^{n_1} \to \Re^m$ and $\boldsymbol{g}_2 : \Re^{n_2} \to \Re^m$ are given affine functions, and K_1, K_2 are nonempty closed convex sets.

The optimisation problem in (1.2.7) has one important feature, which is a requirement of the method presented here: the objective function contains one scalar variable λ. We remark though that other problems with more complicated objectives may be also posed in the form given above, and therefore may be also solved with the method proposed in this thesis. We also point out that this particular form is a common feature in some problems in engineering such as limit analysis [33, 35, 37] or general plastic analysis [29, 30, 31], where λ measures the bearing capacity of a structure or the dissipation power when it collapses. The primal problem in (1.2.7) is written as a maximisation of the objective function, in agreement with the engineering applications, but in contrast to the standard notation in optimisation. We will keep the form in (1.2.7), but of course, the algorithm explained in this thesis may be also described using standard notation.

The main contributions of the thesis are: (*i*) rewriting the constraints in (1.2.7) as the intersection of appropriate sets, (*ii*) decomposing this form of the algorithm into a master problem and two subproblems, (*iii*) applying some results of proximal point theory to this new form of the optimisation problem, and (*iv*) proposing efficient algorithmic strategies to iteratively solve the decomposition algorithm. We prove the convergence properties of the algorithm, and numerically test its efficiency.

The structure of the thesis is as follows. Some requisite background results are presented in Chapter 2. Chapter 3 describes some common methodologies to decompose general optimisation problems, and to particularise them to the problems encountered in limit analysis, which have the structure of second order conic programming. Chapter 4 address the main contributions and numerical results. Chapter 5 presents the main conclusions and future work.

2

Cone Programs

Here is the definition of a cone program, in a somewhat more general format than we would need for our work. This will introduce symmetry between the primal and the dual program.

Remark 2.1 *Refer to Appendix B for definition of cone.*

Definition 2.1 *Let $K \subset \Re^n$, $L \subset \Re^m$ be closed convex cones, $\boldsymbol{b} \in \Re^m$, $\boldsymbol{c} \in \Re^n$, $\mathbf{A} : \Re^n \to \Re^m$ a linear operator. A cone program is a constrained optimisation problem of the form:*

$$\begin{aligned} \min_{\boldsymbol{x}} \; & \boldsymbol{c} \cdot \boldsymbol{x} \\ & \mathbf{A}\boldsymbol{x} - \boldsymbol{b} \in L \\ & \boldsymbol{x} \in K. \end{aligned} \tag{2.0.1}$$

For $L = \{\mathbf{0}\}$, we get cone programs in equality form.

Following the linear programming case, we call the cone program feasible if there is some feasible solution, a vector $\bar{\boldsymbol{x}}$ with $\mathbf{A}\bar{\boldsymbol{x}} - \boldsymbol{b} \in L$, $\bar{\boldsymbol{x}} \in K$. The value (optimal value) of a feasible cone program is defined as

$$\inf\{\boldsymbol{c} \cdot \boldsymbol{x} : \mathbf{A}\boldsymbol{x} - \boldsymbol{b} \in L, \, \boldsymbol{x} \in K\}, \tag{2.0.2}$$

which includes the possibility that the value is $\pm\infty$.

An optimal solution is a feasible solution $\boldsymbol{x}^*$ such that $\boldsymbol{c} \cdot \boldsymbol{x}^* \leq \boldsymbol{c} \cdot \boldsymbol{x}$ for all feasible solutions $\boldsymbol{x}$. Consequently, if there is an optimal solution, the value of the cone program is finite, and that value is attained, meaning that the infinitum (2.0.2) is a minimum.

2.1. Cone Programming Duality

For this section, let us call the cone program (2.0.1) the primal program and name it (P):

$$\begin{aligned}(P) \quad \min_{\boldsymbol{x}} \; & \boldsymbol{c} \cdot \boldsymbol{x} \\ & \mathbf{A}\boldsymbol{x} - \boldsymbol{b} \in L \\ & \boldsymbol{x} \in K.\end{aligned} \tag{2.1.1}$$

Then we define its dual as the cone program

$$\begin{aligned}(D) \quad \max_{\boldsymbol{y}} \; & \boldsymbol{b} \cdot \boldsymbol{y} \\ & \boldsymbol{c} - \mathbf{A}^T\boldsymbol{y} \in K^* \\ & \boldsymbol{y} \in L^*.\end{aligned} \tag{2.1.2}$$

Formally, this does not have the cone program format (2.0.1), but we could easily achieve this if necessary by rewriting (D) as follows.

$$\begin{aligned}(D) \quad \min_{\boldsymbol{y}} \; & -\boldsymbol{b} \cdot \boldsymbol{y} \\ & \boldsymbol{c} - \mathbf{A}^T\boldsymbol{y} \in K^* \\ & \boldsymbol{y} \in L^*,\end{aligned} \tag{2.1.3}$$

where K^* and L^* are the dual sets of K and L respectively, (see Appendix B for definition of dual set).

Having done this, we can also compute the dual of (D) which takes us (not surprisingly) back to (P).

For the dual program (D) which is now a maximisation problem, value (optimal value) is defined through suprimum in the canonical way.

The primal and dual problem are related via the Weak Duality Theorem that has several useful consequences.

Theorem 2.1 *(Weak Duality). If $\boldsymbol{x}$ is feasible in (P) and $\boldsymbol{y}$ is feasible in (D), then the objective function of (P) evaluated at $\boldsymbol{x}$ is not less than the objective function of (D) evaluated at $\boldsymbol{y}$.*

Proof 1 *To demonstrate this result, one need only remember the definition of dual cone. Since $\boldsymbol{x} \in K$ and $\boldsymbol{c} - \mathbf{A}^T\boldsymbol{y} \in K^*$ then we have*

$$\boldsymbol{x} \cdot (\boldsymbol{c} - \mathbf{A}^T\boldsymbol{y}) \geq 0 \Rightarrow (\mathbf{A}\boldsymbol{x}) \cdot \boldsymbol{y} \leq \boldsymbol{c}\boldsymbol{x}. \tag{2.1.4}$$

On the other hand, since $\boldsymbol{y} \in L^*$ *and* $\mathbf{A}\boldsymbol{x} - \boldsymbol{b} \in L$ *then*

$$(\mathbf{A}\boldsymbol{x} - \boldsymbol{b}) \cdot \boldsymbol{y} \geq 0 \Rightarrow \boldsymbol{b} \cdot \boldsymbol{y} \leq (\mathbf{A}\boldsymbol{x}) \cdot \boldsymbol{y}, \tag{2.1.5}$$

consequently, in view of (2.1.4) *and* (2.1.5) *we have*

$$\boldsymbol{b} \cdot \boldsymbol{y} \leq \boldsymbol{c} \cdot \boldsymbol{x}.$$

□

One consequence is that any feasible solution of (D) provides a lower bound on the optimal value of (P); and any feasible solution of (P) provides an upper bound on the optimal value of (D). This can be useful in establishing termination or error control criteria when devising computational algorithms addressed to (P) and (D) ; if at some iteration feasible solutions are available to both (P) and (D) that are close to one another in value, then they must be close to being optimal in their respective problems.

From Theorem 2.1, it also follows that (D) must be infeasible if the optimal value of (P) is $-\infty$ and, similarly, (P) must be infeasible if the optimal value of (D) is $+\infty$.

Definition 2.2 *An interior point (or Slater point) of the cone program* (2.0.1) *is a point* $\boldsymbol{x} \in K$ *with the property that*

$$\mathbf{A}\boldsymbol{x} - \boldsymbol{b} \in \mathit{int}(L).$$

Let us remind the reader that $\mathrm{int}(L)$ is the set of all points of L that have a small ball around it that is completely contained in L.

Theorem 2.2 *(Strong Duality). If the primal program* (P) *is feasible, has finite optimal value* γ*, and has an interior feasible point* $\boldsymbol{x}_0$*, then the dual program* (D) *is feasible and has finite optimal value* $\beta = \gamma$*.*

Proof 2 *See [27].*

The strong Duality Theorem 2.2 is not applicable if the primal cone program (P) is in equality form $(L = \{\mathbf{0}\})$, since the cone $L = \{\mathbf{0}\}$ has no interior points. But there is a different variant constraint qualification that we can use in this case.

Theorem 2.3 *If the primal program*

$$(P) \quad \min_{\boldsymbol{x}} \boldsymbol{c} \cdot \boldsymbol{x}$$
$$\mathbf{A}\boldsymbol{x} - \boldsymbol{b} = \mathbf{0}$$
$$\boldsymbol{x} \in K,$$

is feasible, has finite value γ and has a point $\boldsymbol{x}_0 \in int(K)$ such that $\mathbf{A}\boldsymbol{x}_0 = \boldsymbol{b}$, the dual program

$$(D) \quad \max_{\boldsymbol{y}} \boldsymbol{b} \cdot \boldsymbol{y}$$
$$\boldsymbol{c} - \mathbf{A}^T \boldsymbol{y} \in K^*,$$

is feasible and has finite value $\beta = \gamma$.

Proof 3 *See [27].*

We remark that for general L, just requiring a point $\boldsymbol{x}_0 \in \text{int}(K)$ with $\mathbf{A}\boldsymbol{x}_0 - \boldsymbol{b} \in L$ is not enough to achieve strong duality.

In the next section, we consider the problem of finding a best approximation pair, i.e., two points which achieve the minimum distance between two closed convex sets in $\Re^n$. The method under consideration is termed AAR for averaged alternating reflections and produces best approximation pairs provided that they exist.

2.2. Method of Averaged Alternating Reflection (AAR Method)

2.2.1. Best Approximation Operators

Definition 2.3 *Let C be a subset of $\Re^n$, let $\boldsymbol{x} \in \Re^n$, and let $\boldsymbol{p} \in C$. Then $\boldsymbol{p}$ is a best approximation to $\boldsymbol{x}$ from C (or a projection of $\boldsymbol{x}$ onto C) if*

$$\|\boldsymbol{x} - \boldsymbol{p}\| = d_C(\boldsymbol{x}) := \inf\{\|\boldsymbol{z} - \boldsymbol{x}\| : \boldsymbol{z} \in C\}. \tag{2.2.1}$$

If every point in $\Re^n$ has at least one projection onto C, then C is proximal. If every point in $\Re^n$ has exactly one projection onto C, then C is a Chebyshev set. In this case, the projector (or projection operator) onto C is the operator, denoted by P_C, that maps every point in $\Re^n$ to its unique projection onto C.

Theorem 2.4 *Let C be a nonempty closed convex subset of $\Re^n$. Then C is a Chebyshev set and, for every $\boldsymbol{x}$ and $\boldsymbol{p}$ in $\Re^n$,*

$$\boldsymbol{p} = P_c\boldsymbol{x} \Leftrightarrow [\boldsymbol{p} \in C \quad and \quad (\forall \boldsymbol{y} \in C), \quad (\boldsymbol{x} - \boldsymbol{p}) \cdot (\boldsymbol{y} - \boldsymbol{p}) \leq 0]. \tag{2.2.2}$$

Proof 4 *See [24].*

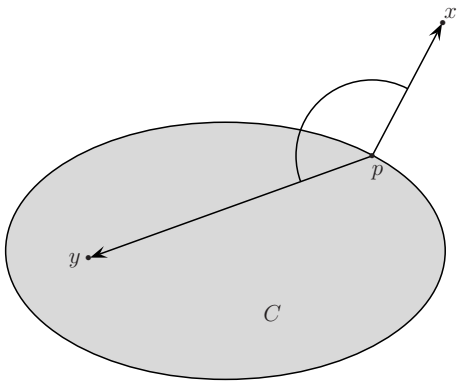


Figure 2.1.: Projection onto a nonempty closed convex set C in the Euclidean plane. The characterization (2.2.2) states that $\boldsymbol{p} \in C$ is the projection of $\boldsymbol{x}$ onto C if and only if the vectors $\boldsymbol{x} - \boldsymbol{p}$ and $\boldsymbol{y} - \boldsymbol{p}$ form a right or obtuse angle for every $\boldsymbol{y} \in C$.([6],page 47)

Remark 2.2 *Theorem 2.4 states that every nonempty closed convex set is a Chebyshev set.*

Example 2.1 *Let $C = \{\boldsymbol{x} : \|\boldsymbol{x}\| \leq 1\}$. Then*

$$(\forall \boldsymbol{x} \in \Re^n) \quad P_C\boldsymbol{x} = \frac{1}{\max\{\|\boldsymbol{x}\|, 1\}}\boldsymbol{x}. \tag{2.2.3}$$

A natural extension of the Definition 2.3 is to find a best approximation pair relative to (C, W) where C and W are subsets of $\Re^n$. i.e., to

$$\text{find } (\bar{\boldsymbol{c}}, \bar{\boldsymbol{w}}) \in C \times W \quad \text{such that } \|\bar{\boldsymbol{c}} - \bar{\boldsymbol{w}}\| = \inf_{\substack{\boldsymbol{c} \in C \\ \boldsymbol{w} \in W}} \|\boldsymbol{c} - \boldsymbol{w}\| \coloneqq d(C, W). \tag{2.2.4}$$

If $W = \{\boldsymbol{w}\}$, (2.2.4) reduces to (2.2.1) and its solution is $P_C\boldsymbol{w}$. On the other hand, when the problem is consistent, i.e., $W \cap C \neq \emptyset$, then (2.2.4) reduces to the well-known convex feasibility problem for two sets [5, 18] and its solution set is $\{(\boldsymbol{x}, \boldsymbol{x}) \in \Re^n \times \Re^n | \boldsymbol{x} \in W \cap C\}$. The formulation in (2.2.4) captures a wide range of problems in applied mathematics and engineering [17, 28, 43].

2.2.2. Nonexpansive Operators

Nonexpansive operators play a central role in applied mathematics, because many problems in nonlinear analysis reduce to finding fixed points of nonexpansive operators. In this section, we discuss nonexpansiveness and several variants. The properties of the fixed point sets of nonexpansive operators are investigated, in particular in terms of convexity.

Definition 2.4 *Let D be a nonempty subset of $\Re^n$ and let $T : D \to \Re^n$. Then T is*

(i) firmly nonexpansive if

$$(\forall \boldsymbol{x} \in D), (\forall \boldsymbol{y} \in D) : \ \|T(\boldsymbol{x}) - T(\boldsymbol{y})\|^2 + \|(I - T)(\boldsymbol{x}) - (I - T)(\boldsymbol{y})\|^2 \leq \|\boldsymbol{x} - \boldsymbol{y}\|^2,$$

(ii) nonexpansive if

$$(\forall \boldsymbol{x} \in D), (\forall \boldsymbol{y} \in D) : \ \|T(\boldsymbol{x}) - T(\boldsymbol{y})\| \leq \|\boldsymbol{x} - \boldsymbol{y}\|.$$

It is clear that firm nonexpansiveness implies nonexpansiveness.

Proposition 2.1 *Let C be a nonempty closed convex subset of $\Re^n$. Then*

(i) the projector P_C is firmly nonexpansive,

(ii) $2P_C - I$ is nonexpansive.

See [6] for a proof of this lemma. The transformation $2P_C - I$ is named the reflection operator with respect to C and will be denoted by R_C. See Figure 2.2 for an illustration of the reflection operator .

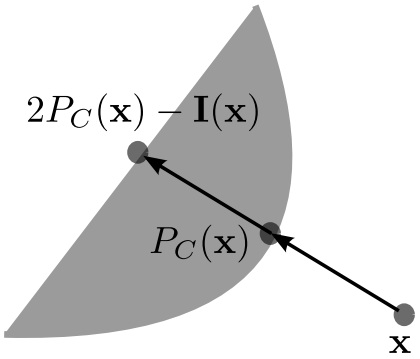


Figure 2.2.: Illustration of reflection operator, $R_C = 2P_C - I$.

Definition 2.5 *The set of fixed points of an operator $T : X \to X$ is denoted by Fix T, i.e.,*

$$Fix\ T = \{\boldsymbol{x} \in X | T(\boldsymbol{x}) = \boldsymbol{x}\}. \tag{2.2.5}$$

It is convenient to introduce the following sets, which we will use throughout this section,

$$\begin{aligned} &C = \overline{Z - W} = \overline{\{\boldsymbol{z} - \boldsymbol{w} | \boldsymbol{z} \in Z, \boldsymbol{w} \in W\}}, \\ &\boldsymbol{v} = P_C(0),\ G = W \cap (Z - \boldsymbol{v}),\ H = (W + \boldsymbol{v}) \cap Z, \end{aligned} \tag{2.2.6}$$

where Z and W are nonempty closed convex subsets of $\Re^n$ and $\overline{Z - W}$ denotes the closure of $Z - W$. See Appendix B for a brief introduction to convex sets.

Note also that if $W \cap Z \neq \emptyset$, then $G = H = W \cap Z$. However, even when $W \cap Z = \emptyset$, the sets G and H may be nonempty and they serve as substitutes for the intersection. In words, vector $\boldsymbol{v}$ joins the two sets Z and W at the point that are at the minimum distance and $\|\boldsymbol{v}\|$ measures the gap between the sets W and Z.

Proposition 2.2 *From the definitions in* (2.2.5)-(2.2.6)*, the following identities hold:*

(i) $\|\boldsymbol{v}\| = \inf \|W - Z\|$.

(ii) $G = Fix\ (P_W P_Z)$ *and* $H = Fix\ (P_Z P_W)$.

(iii) $G + \boldsymbol{v} = H$.

The proof can be found in [4], Section 5.

Proposition 2.3 *Suppose that $(\boldsymbol{w}_n)_{n\in N}$ and $(\boldsymbol{z}_n)_{n\in N}$ are sequences in W and Z, respectively. Then*

$$\boldsymbol{z}_n - \boldsymbol{w}_n \to \boldsymbol{v} \iff \|\boldsymbol{z}_n - \boldsymbol{w}_n\| \to \|\boldsymbol{v}\|.$$

Also, assume that $\boldsymbol{z}_n - \boldsymbol{w}_n \to \boldsymbol{v}$. Then the following identities hold:

(i) $\boldsymbol{z}_n - P_W(\boldsymbol{z}_n) \to \boldsymbol{v}$, *and* $P_Z(\boldsymbol{w}_n) - \boldsymbol{w}_n \to \boldsymbol{v}$.

(ii) The weak cluster points of $(\boldsymbol{w}_n)_{n\in N}$ and $(P_W(\boldsymbol{z}_n))_{n\in N}$ (resp.$(\boldsymbol{z}_n)_{n\in N}$ and $(P_Z(\boldsymbol{w}_n))_{n\in N}$) belong to G (resp.H). Consequently, the weak cluster points of the sequences

$$((\boldsymbol{w}_n, \boldsymbol{z}_n))_{n\in N}, \quad ((\boldsymbol{w}_n), P_Z(\boldsymbol{w}_n))_{n\in N} \quad ((P_W(\boldsymbol{z}_n), \boldsymbol{z}_n)_{n\in N})$$

are best approximation pairs relative to (W, Z).

(iii) If $G = \emptyset$ or, equivalently, $H = \emptyset$, then

$$\min\{\|\boldsymbol{w}_n\|, \|\boldsymbol{z}_n\|, \|P_W(\boldsymbol{z}_n)\|, \|P_Z(\boldsymbol{w}_n)\|\} \to \infty.$$

These results are proved in [7].

Proposition 2.4 *Let $T_1 : \Re^n \to \Re^n$ and $T_2 : \Re^n \to \Re^n$ be firmly nonexpansive and set*

$$T_3 = \frac{(2T_1 - I)(2T_2 - I) + I}{2}.$$

Then the following results hold:

(i) T_3 is firmly nonexpansive.

(ii) Fix T_3 = Fix $(2T_1 - I)(2T_2 - I)$.

2.2.3. Averaged Alternating Reflections (AAR)

Definition 2.6 *We define the so-called Averaged Alternating Reflections (AAR) operator, denoted by T and given by,*

$$T = \frac{R_W R_Z + I}{2}, \tag{2.2.7}$$

with R_W and R_Z the reflection operations illustrated in Figure 2.2.

Note that throughout this section, we assume that W and Z are nonempty closed convex subsets of $\Re^n$.

In view of Proposition 2.1 and 2.4, and since R_Z and R_W are firmly nonexpansive, we infer that T is firmly nonexpansive and

$$\text{Fix } T = \text{Fix } R_W R_Z.$$

Proposition 2.5 *Let W and Z be nonempty closed convex subsets of $\Re^n$ and let T be the operator in* (2.2.7)*. Then the following results hold:*

(i) $I - T = P_Z - P_W R_Z$.

(ii) $W \cap Z \neq \emptyset$ *if and only if Fix* $T \neq \emptyset$.

Proof 5 *(i):*

$$\begin{aligned} T &= \frac{R_W R_Z + I}{2} = \frac{2P_W R_Z - R_Z + I}{2} = \frac{2P_W R_Z - 2P_Z + I + I}{2} \\ &= P_W R_Z - P_Z + I \Rightarrow I - T = P_Z - P_W R_Z. \end{aligned}$$

(ii): Assume that $\boldsymbol{x} \in W \cap Z$. Clearly, we then have that $P_W(\boldsymbol{x}) = \boldsymbol{x}$ and $P_Z(\boldsymbol{x}) = \boldsymbol{x}$, which in turn imply that $R_W(\boldsymbol{x}) = \boldsymbol{x}$ and $R_Z(\boldsymbol{x}) = \boldsymbol{x}$. Using the definition in (2.2.7)*, it follows that*

$$T(\boldsymbol{x}) = \boldsymbol{x},$$

so that Fix $T \neq \emptyset$.

Conversely, if $\boldsymbol{x} \in$ Fix T, then $T(\boldsymbol{x}) = \boldsymbol{x}$, and then according to (i), we have, that $P_Z(\boldsymbol{x}) = P_W R_Z(\boldsymbol{x})$, and therefore $P_Z(\boldsymbol{x}) \in Z$ and $P_Z(\boldsymbol{x}) \in W$, which is equivalent to $P_Z(\boldsymbol{x}) \in W \cap Z$. □

We now recall the well known convergence results for the method of Averaged Alternating Reflections (AAR).

Proposition 2.6 *(Convergence of AAR method). Consider the following successive approximation method: Take $\boldsymbol{t}_0 \in \Re^n$, and set*

$$\boldsymbol{t}_n = T^n(\boldsymbol{t}_0) = T(\boldsymbol{t}_{n-1}),\ n = 1, 2, \ldots$$

where T is defined in (2.2.7)*, and W, Z are nonempty closed convex subsets of $\Re^n$. Then the following results hold*

(i) *Fix* $T \neq \emptyset \iff (T^n(\boldsymbol{t}_0))_{n\in N}$ *converges to some point in Fix* T.
(ii) *Fix* $T = \emptyset \iff \|T^n(\boldsymbol{t}_0)\| \to \infty$, *when* $n \to \infty$.
(iii) $(\|P_Z(\boldsymbol{t}_n) - P_W P_Z(\boldsymbol{t}_n)\|)$ *converges to* $\inf \|W - Z\|$, *and* $(\|P_Z(\boldsymbol{t}_n) - P_W R_Z(\boldsymbol{t}_n)\|)$ *converges to* $\inf \|W - Z\|$.
(iv) $\dfrac{\|\boldsymbol{t}_n\|}{n} \to \inf \|W - Z\|$.

Proof 6 (i) *and* (ii) *are demonstrated in [3, 40, 12] and* (iii) *in [7], while* (iv) *is demonstrated in [42].*

We will resort to these results in Chapter 4, where decomposition technique based on the AAR method is presented.

3 Decomposition Techniques

3.1. Introduction

The size of an optimisation problem can be very large and it is not hard to encounter practical problems with several hundred thousands of equations or unknowns (see Table 1.1). In order to solve such problems, it is convenient to design some special techniques. Decomposition is a general approach to solving a problem by breaking it up into smaller ones and solving each of the smaller ones separately, either in parallel or sequentially in conjunction with a master problem that couples the subproblems. (When it is done sequentially, the advantage comes from the fact that problem complexity grows more than linearly). Decomposition in optimisation is an old idea, and appears in the early works on large-scale linear problems (LPs) in the 1960s [23]. A good reference on decomposition methods is Chapter 6 of Bertsekas [9]. The original primary motivation for decomposition methods was to solve very large problems that where beyond the reach of standard techniques, possibly using multiple processors.

The idea of decomposition comes up in the context of solving linear equations, but goes by other names such as block elimination, Schur complement methods, or (for special cases) matrix inversion lemma (see [11]). The core idea, i.e., using efficient methods to solve subproblems, and combining the results in such a way as to solve the larger problem, is the same as the one exploited here when decomposing an optimisation problem, although the techniques are slightly different.

The original primary motivation for decomposition methods was to solve very large problems that were beyond the reach of standard techniques,

possibly using multiple processors [23, 8]. This remains a good reason to use decomposition methods for some problems. But other reasons are emerging as equally (or more) important. In many cases decomposition methods yield decentralized solution methods. Even if the resulting algorithms are slower (in some cases, much slower) than centralized methods, decentralized solutions might be prefered for other reasons. For example, decentralized solution methods can often be translated into, or interpreted as, simple protocols that allow a collection of subsystems to coordinate their actions to achieve global optimality [44].

Problems for which the variables can be decomposed into uncoupled sets of equations are called separable. As a general example of such a problem, suppose that variable $\boldsymbol{x}$ can be partitioned into subvectors $\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_k$, the cost function is a sum of functions of $\boldsymbol{x}_i$, and each constraint only involves the local subvectors $\boldsymbol{x}_i$. Then, evidently we can solve each problem involving $\boldsymbol{x}_i$ separately, and re-assemble the solutions into $\boldsymbol{x}$.

A more interesting situation occurs when there is some coupling or interaction between the subvectors, so the problems cannot be solved independently. For these cases there are techniques that solve the overall problem by iteratively solving a sequence of smaller problems. There are many ways to do this. In this Chapter we review some well-known decomposition techniques [19, 8, 23] and illustrate their efficiency with some simple examples and some problems in limit analysis. These results will motivate the proposed method in Chapter 4.

The efficiency and applicable of a decomposition technique depends on the structure of the problem at hand. Two basic structures arise in practice, problems with complicating constraints and problems with complicating variables structures to be described next.

3.1.1. Complicating Constraints

Let us assume a linear optimisation problem where the primal variables have been partitioned into different blocks. Complicating constraints involve variables from different blocks, which drastically complicate the solution of the problem and prevent its solution by blocks. The following example illustrates how complicating constraints impede a solution by blocks. Consider

the problem:

$$\begin{array}{lllllll} \min\limits_{x_i,y_i} & a_1x_1 & +a_2x_2 & +a_3x_3 & +b_1y_1 & +b_2y_2 & \\ & a_{11}x_1 & +a_{12}x_2 & +a_{13}x_3 & & & = e_1 \\ & a_{21}x_1 & +a_{22}x_2 & +a_{23}x_3 & & & = e_2 \\ & & & & b_{11}y_1 & +b_{12}y_2 & = g_1 \\ & & & & b_{21}y_1 & +b_{22}y_2 & = g_2 \\ & d_{11}x_1 & +d_{12}x_2 & +d_{13}x_3 & +d_{14}y_1 & +d_{15}y_2 & = h_1 \\ & x_1 \geq 0, & x_2 \geq 0, & x_3 \geq 0, & y_1 \geq 0, & y_2 \geq 0. & \end{array} \tag{3.1.1}$$

If the last equality is not enforced, i.e., it is relaxed, the above problem decomposes into the following two subproblems:

Subproblem 1:

$$\begin{aligned} \min_{x_1,x_2,x_3} \quad & a_1x_1 + a_2x_2 + a_3x_3 \\ & a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = e_1 \\ & a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = e_2 \\ & x_1 \geq 0, x_2 \geq 0, x_3 \geq 0 \end{aligned}$$

Subproblem 2:

$$\begin{aligned} \min_{y_1,y_2} \quad & b_1y_1 + b_2y_2 \\ & b_{11}y_1 + b_{12}y_2 = g_1 \\ & b_{21}y_1 + b_{22}y_2 = g_2 \\ & y_1 \geq 0, y_2 \geq 0 \end{aligned}$$

Since the last equality constraint in (3.1.1) involves all variables, preventing a solution by blocks, this equation is a complicating constraint.

Decomposition procedures are computational techniques that indirectly consider the complicating constraints and solve a set of problems with smaller size. The price that has to be paid for such a size reduction is the amount of subproblems to be solved. In other words, instead of solving the original problem with complicating constraints, two problems are solved iteratively: a simple so-called master problem, and a set of subproblems, similar to those included in the original one but without complicating constraints.

3.1.2. Complicating Variables

In linear problems, the complicating variables are those variables preventing a solution of the problem by independent blocks. For instance, let us consider the following problem:

$$
\begin{array}{lllllll}
\min\limits_{x_i,y_i,\lambda} a_1x_1 & +a_2x_2 & +a_3x_3 & +b_1y_1 & +b_2y_2 & & \\
a_{11}x_1 & +a_{12}x_2 & +a_{13}x_3 & & & +d_{11}\lambda & = e_1 \\
a_{21}x_1 & +a_{22}x_2 & +a_{23}x_3 & & & +d_{21}\lambda & = e_2 \\
 & & & b_{11}y_1 & +b_{12}y_2 & +d_{31}\lambda & = g_1 \\
 & & & b_{21}y_1 & +b_{22}y_2 & +d_{41}\lambda & = g_2 \\
x_1 \geq 0, & x_2 \geq 0, & x_3 \geq 0, & y_1 \geq 0, & y_2 \geq 0, & \lambda \geq 0 &
\end{array}
$$

If variable λ is given the fixed value $\lambda^{fixed} \geq 0$ the problem decomposes into the two subproblems:

Subproblem 1:

$$
\begin{aligned}
\min_{y_1,y_2}\ & a_1x_1 + a_2x_2 + a_3x_3 \\
& a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = e_1 - d_{11}\lambda^{fixed} \\
& a_{21}x_1 + a_{22}x_2 + a_{23}x_3 = e_2 - d_{21}\lambda^{fixed} \\
& x_1 \geq 0, x_2 \geq 0, x_3 \geq 0.
\end{aligned}
$$

Subproblem 2:

$$
\begin{aligned}
\min_{y_1,y_2}\ & b_1y_1 + b_2y_2 \\
& b_{11}y_1 + b_{12}y_2 = g_1 - d_{31}\lambda^{fixed} \\
& b_{21}y_1 + b_{22}y_2 = g_2 - d_{41}\lambda^{fixed} \\
& y_1 \geq 0, y_2 \geq 0.
\end{aligned}
$$

In the subsequent sections we will describe some decomposition methods: primal decomposition, dual decomposition, and Benders decomposition. We apply these techniques to some illustrative toy problems.

Before, we introduce an iterative method for solving optimisation problems called projected subgradient method. This method will be employed in the reminder of the present Chapter.

3.2. Projected Subgradient Method

We aim to solve the following constrained optimisation problem:

$$\begin{aligned} &\min_{\boldsymbol{x}} f(\boldsymbol{x}) \\ &\quad \boldsymbol{x} \in C, \end{aligned}$$

where $f : R^n \to R, and\ C \subseteq R^n$ are convex. The projected subgradient method consists on given a feasible candidate $\boldsymbol{x}^k$, obtain a feasible next iterate $\boldsymbol{x}^{k+1}$ as,

$$\boldsymbol{x}^{k+1} = \mathbf{P}(\boldsymbol{x}^k - \alpha_k \boldsymbol{g}^k), \tag{3.2.1}$$

where $\mathbf{P}$ is an operator that projects its argument on C, and $\boldsymbol{g}^k \in \partial f(\boldsymbol{x}^k)$, with $\partial f(\boldsymbol{x}^k)$ the set of all subgradients of function f at point $\boldsymbol{x}^k$. For instance if C is the set of linear equality constraints, that is $C = \{\boldsymbol{x} | \mathbf{A}\boldsymbol{x} = \boldsymbol{b}\}$, then the projection of $\boldsymbol{z}$ onto C reads,

$$\begin{aligned} \mathbf{P}(\boldsymbol{z}) &= \boldsymbol{z} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}(\mathbf{A}\boldsymbol{z} - \boldsymbol{b}) \\ &= (\mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A})\boldsymbol{z} + \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\boldsymbol{b}. \end{aligned} \tag{3.2.2}$$

In this case, after using $\mathbf{A}\boldsymbol{x}^k = \boldsymbol{b}$, the update process in (3.2.1) yields,

$$\begin{aligned} \boldsymbol{x}^{k+1} &= \mathbf{P}(\boldsymbol{x}^k - \alpha_k \boldsymbol{g}^k) \\ &= \boldsymbol{x}^k - \alpha_k(\mathbf{I} - \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}\mathbf{A})\boldsymbol{g}^k, \end{aligned}$$

where α_k is a step size. There are some rules for choosing appropriately α_k which will be considered below.

Note that if $\mathbf{P}$ is the identity projection, then the method turns into the subgradient method, which does not enforce the feasibility of $\boldsymbol{x}^{k+1}$.

3.3. Decomposition of Unconstrained Problems

3.3.1. Primal Decomposition

We will consider the simplest possible case, an unconstrained optimisation problem that splits into two subproblems. (But note that the most impressive applications of decomposition occur when the problem is split into many subproblems.) In our first example, we consider an unconstrained minimization problem, of the form :

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \min_{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{y}} f_1(\boldsymbol{x}_1, \boldsymbol{y}) + f_2(\boldsymbol{x}_2, \boldsymbol{y}) \tag{3.3.1}$$

where the variable is $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{y})$. Although the dimensions do not matter here, it is useful to think of $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ as having relatively high dimension, and $\boldsymbol{y}$ having relatively small dimension. The objective is almost block separable in $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$; indeed, if we fix the subvector $\boldsymbol{y}$, the problem becomes separable in $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, and therefore can be solved by solving the two subproblems independently. For this reason, $\boldsymbol{y}$ is called the complicating variable, because when it is fixed, the problem splits or decomposes. In other words, the variable $\boldsymbol{y}$ complicates the problem. It is the variable that couples the two subproblems. We can think of $\boldsymbol{x}_1(\boldsymbol{x}_2)$ as the private variable or local variable associated with the first (second) subproblem, and $\boldsymbol{y}$ as the public variable or interface variable or boundary variable between the two subproblems. Indeed, by rewriting (3.3.1) as:

$$\min_{\boldsymbol{y}} \left(\min_{\boldsymbol{x}_1} f_1(\boldsymbol{x}_1, \boldsymbol{y}) + \min_{\boldsymbol{x}_2} f_2(\boldsymbol{x}_2, \boldsymbol{y}) \right), \tag{3.3.2}$$

we observe that the problem becomes separable when $\boldsymbol{y}$ is fixed. This suggests a method for solving the problem (3.3.1). Let $g_i(\boldsymbol{y})$ denote the inner optimum in the previous expression,

$$g_i(\boldsymbol{y}) = \min_{\boldsymbol{x}_i} f_i(\boldsymbol{x}_i, \boldsymbol{y}) \quad (i = 1, 2). \tag{3.3.3}$$

Note that if f_1 and f_2 are convex, so are g_1 and g_2. We refer to (3.3.3) as subproblem(i), $(i = 1, 2)$.

Then the original problem in (3.3.1) is equivalent to the following master problem:

$$\min_{\boldsymbol{y}} g_1(\boldsymbol{y}) + g_2(\boldsymbol{y}).$$

If the original problem is convex, so is the master problem. The variables of the master problem are the complicating or coupling variables of the original problem. The objective of the master problem is the sum of the optimal values of the subproblems.

A decomposition method solves the problem (3.3.1) by solving the master problem, using an iterative method such as the subgradient method described in Section 3.2. Each iteration requires solving the two subproblems in order to evaluate $g_1(\boldsymbol{y})$ and $g_2(\boldsymbol{y})$ and their gradients and subgradients. This can be done in parallel, but even if it is done sequentially, there will be

substantial savings if the computational complexity of the problems grows more than linearly with problem size.

Lets see how to evaluate a subgradient of g_1 at $\boldsymbol{y}$, assuming the problem is convex. We first solve the associated subproblem, i.e., we find $\bar{x}_1(\boldsymbol{y})$ that minimizes $f_1(\boldsymbol{x}_1, \boldsymbol{y})$. Thus, there is a subgradient of f_1 of the form $(\mathbf{0}, \boldsymbol{q}_1)$, and not surprisingly, $\boldsymbol{q}_1$ is a subgradient of g_1 at $\boldsymbol{y}$. We can carry out the same procedure to find a subgradient $\boldsymbol{q}_2 \in \partial g_2(\boldsymbol{y})$. Then $\boldsymbol{q}_1 + \boldsymbol{q}_2$ is a subgradient of $g_1 + g_2$ at $\boldsymbol{y}$.

We can solve the master problem by a variety of methods, including subgradient method (if the functions are nondifferentiable). This basic decomposition method is called primal decomposition because the master algorithm manipulates (some of the) primal variables.

When we use a subgradient method to solve the master problem, we obtain the following primal decomposition algorithm:

Repeat

- Solve the subproblems in (3.3.3), possibly in parallel. Set $\boldsymbol{y} = \boldsymbol{y}^k$.
  - Find $\bar{\boldsymbol{x}}_1$ that minimizes $f_1(\boldsymbol{x}_1, \boldsymbol{y}^k)$, and a subgradient $\boldsymbol{q}_1 \in \partial g_1(\boldsymbol{y}^k)$.
  - Find $\bar{\boldsymbol{x}}_2$ that minimizes $f_2(\boldsymbol{x}_2, \boldsymbol{y}^k)$, and a subgradient $\boldsymbol{q}_2 \in \partial g_2(\boldsymbol{y}^k)$.
- Update complicating variable,

$$\boldsymbol{y}^{k+1} = \boldsymbol{y}^k - \alpha^k(\boldsymbol{q}_1^k + \boldsymbol{q}_2^k).$$

  Here α^k is a step length that can be chosen in any of the standard ways [9].

When a subgradient method is used for the master problem, and both g_1 and g_2 are differentiable, the update has a very simple interpretation. We interpret $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ as the gradients of the optimal value of the subproblems, with respect to the complicating variable $\boldsymbol{y}$. The update simply moves the complicating variable in a direction of improvement of the overall objective.

The basic primal decomposition method described above can be extended in several ways. We can add separable constraints, i.e., constraints of the form $\boldsymbol{x}_1 \in C_1$ and $\boldsymbol{x}_2 \in C_2$. In this case (and also, in the case when $\mathrm{dom} f_i$ is not all vectors) we have the possibility that $g_i(\boldsymbol{y}) = \infty$ (i.e., $\boldsymbol{y} \notin \mathrm{dom} g_i$) for some choices of $\boldsymbol{y}$. In this case we find a cutting-plane that separates $\boldsymbol{y}$ from dom g_i, to use in the master algorithm.

3.3.2. Dual Decomposition

We can apply decomposition to the problem (3.3.1) after introducing some new variables, and working with the dual problem. We first rewrite the problem as

$$\min_{\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{y}} f_1(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{y}) + f_2(\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{y}) = \min_{\boldsymbol{x}_1,\boldsymbol{x}_2,\boldsymbol{y}_1=\boldsymbol{y}_2} f_1(\boldsymbol{x}_1,\boldsymbol{y}_1) + f_2(\boldsymbol{x}_2,\boldsymbol{y}_2)$$

We have introduced a local version of the complicating variable $\boldsymbol{y}$, along with a consistency constraint that requires the two local versions to be equal. Note that the objective is now separable, with the variable partition $(\boldsymbol{x}_1, \boldsymbol{y}_1)$ and $(\boldsymbol{x}_2, \boldsymbol{y}_2)$. Now we form the dual problem. The Lagrangian is equal to:

$$\begin{aligned} L(\boldsymbol{x}_1,\boldsymbol{y}_1,\boldsymbol{x}_2,\boldsymbol{y}_2,\boldsymbol{v}) &= f_1(\boldsymbol{x}_1,\boldsymbol{y}_1) + f_2(\boldsymbol{x}_2,\boldsymbol{y}_2) + \boldsymbol{v}\cdot(\boldsymbol{y}_1-\boldsymbol{y}_2) \\ &= f_1(\boldsymbol{x}_1,\boldsymbol{y}_1) + f_2(\boldsymbol{x}_2,\boldsymbol{y}_2) + \boldsymbol{v}\cdot\boldsymbol{y}_1 - \boldsymbol{v}\cdot\boldsymbol{y}_2, \end{aligned}$$

which is separable. The dual function is given by

$$q(\boldsymbol{v}) = q_1(\boldsymbol{v}) + q_2(\boldsymbol{v}),$$

where

$$\begin{aligned} q_1(\boldsymbol{v}) &= \inf_{\boldsymbol{x}_1,\boldsymbol{y}_1} f_1(\boldsymbol{x}_1,\boldsymbol{y}_1) + \boldsymbol{v}\cdot\boldsymbol{y}_1, \\ q_2(\boldsymbol{v}) &= \inf_{\boldsymbol{x}_2,\boldsymbol{y}_2} f_2(\boldsymbol{x}_2,\boldsymbol{y}_2) - \boldsymbol{v}\cdot\boldsymbol{y}_2. \end{aligned} \tag{3.3.4}$$

Note that q_1 and q_2 can be evaluated completely independently, e.g, in parallel. The dual problem reads:

$$\max_{\boldsymbol{v}} q_1(\boldsymbol{v}) + q_2(\boldsymbol{v}), \tag{3.3.5}$$

with the dual variable $\boldsymbol{v}$. This is the master problem in dual decomposition. The master algorithm solve this problem using a subgradient method or other methods.

To evaluate a subgradient of $-q_1$ or $-q_2$ is easy. We find $\bar{\boldsymbol{x}}_1$ and $\bar{\boldsymbol{y}}_1$ that minimize $f_1(\boldsymbol{x}_1,\boldsymbol{y}_1)+\boldsymbol{v}\cdot\boldsymbol{y}_1$ over $\boldsymbol{x}_1$ and $\boldsymbol{y}_1$. Then a subgradient of $-q_1$ at $\boldsymbol{v}$ is given by $-\bar{\boldsymbol{y}}_1$. Similarly, if $\bar{\boldsymbol{x}}_2$ and $\bar{\boldsymbol{y}}_2$ minimize $f_2(\boldsymbol{x}_2,\boldsymbol{y}_2)-\boldsymbol{v}\cdot\boldsymbol{y}_2$ over $\boldsymbol{x}_2$ and $\boldsymbol{y}_2$, then a subgradient of $-q_2$ at $\boldsymbol{v}$ is given by $\bar{\boldsymbol{y}}_2$. Thus, a subgradient of the negative dual function $-q$ is given by $\bar{\boldsymbol{y}}_2 - \bar{\boldsymbol{y}}_1$, which is nothing more than the consistency constraint residual.

If we use a subgradient method to solve the master problem, the dual decomposition algorithm has the following form:

Repeat

- Solve the subproblems in (3.3.4), possibly in parallel. Set $\boldsymbol{v} = \boldsymbol{v}^k$.
  - Find $\boldsymbol{x}_1^k$ and $\boldsymbol{y}_1^k$ that minimize $f_1(\boldsymbol{x}_1, \boldsymbol{y}_1) + \boldsymbol{v}^k \cdot \boldsymbol{y}_1$.
  - Find $\boldsymbol{x}_2^k$ and $\boldsymbol{y}_2^k$ that minimize $f_2(\boldsymbol{x}_2, \boldsymbol{y}_2) - \boldsymbol{v}^k \cdot \boldsymbol{y}_2$.
- Update dual variables.

$$\boldsymbol{v}^{k+1} = \boldsymbol{v}^k + \beta^k(\boldsymbol{y}_1^k - \boldsymbol{y}_2^k).$$

  Here β^k is a step size which can be chosen in several ways.

If the dual function q is differentiable, then we can choose a constant step size, provided it is small enough. Another choice in this case is to carry out a line search on the dual objective. If the dual function is nondifferentiable, we can use a diminishing nonsummable step size, such as $\beta^k = \frac{\alpha}{k}$ [8], which satisfies the following properties.

$$\lim_{k\to\infty} \beta^k = 0, \quad \sum_{k=1}^{\infty} \beta^k = \infty.$$

At each step of the dual decomposition algorithm, we have a lower bound on P^*, the optimal value of the original problem, given by

$$P^* \geq q(\boldsymbol{v}) = f_1(\boldsymbol{x}_1, \boldsymbol{y}_1) + \boldsymbol{v} \cdot \boldsymbol{y}_1 + f_2(\boldsymbol{x}_2, \boldsymbol{y}_2) - \boldsymbol{v} \cdot \boldsymbol{y}_2,$$

where $\boldsymbol{x}_1$, $\boldsymbol{y}_1$, $\boldsymbol{x}_2$ and $\boldsymbol{y}_2$ are the iterates. Generally, the iterates are not feasible for the original problem, i.e., we have $\boldsymbol{y}_2 - \boldsymbol{y}_1 \neq \boldsymbol{0}$. (If they are feasible, we have maximized q.) A reasonable guess of a feasible point can be constructed from this iterate as $(\boldsymbol{x}_1, \bar{\boldsymbol{y}}), (\boldsymbol{x}_2, \bar{\boldsymbol{y}})$, where $\bar{\boldsymbol{y}} = \frac{(\boldsymbol{y}_1 + \boldsymbol{y}_2)}{2}$. In other words, we replace $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ (which are different) with their average value. (The average is the projection of $(\boldsymbol{y}_1, \boldsymbol{y}_2)$ onto the feasible set $\boldsymbol{y}_1 = \boldsymbol{y}_2$). This gives an upper bound on P^*, given by $P^* \leq f_1(\boldsymbol{x}_1, \bar{\boldsymbol{y}}) + f_2(\boldsymbol{x}_2, \bar{\boldsymbol{y}})$. A better feasible point can be found by replacing $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ with their average, and then solving the two subproblems (3.3.3) encountered in primal decomposition, i.e., by evaluating $g_1(\bar{\boldsymbol{y}}) + g_2(\bar{\boldsymbol{y}})$. This gives the bound

$$P^* \leq g_1(\bar{\boldsymbol{y}}) + g_2(\bar{\boldsymbol{y}}).$$

There is one subtlety in dual decomposition. Even if we do find the optimal dual solution $\boldsymbol{v}^*$, there is the question of finding the optimal values

of $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, and $\boldsymbol{y}$. When f_1 and f_2 are strictly convex, the points found in evaluating q_1 and g_2 are guaranteed to converge to optimal, but in general the situation can be more difficult. (For more on finding the primal solution from the dual, see ([11], section 5.5.5).

As in the primal decomposition method, we can encounter infinite values for the subproblems. In dual decomposition, we can have $q_i(\boldsymbol{v}) = \infty$. This can occur for some values of $\boldsymbol{v}$, if the functions f_i grow only linearly in $\boldsymbol{y}_i$. In this case we generate a cutting-plane that separates the current price vector from $\mathrm{dom} g_i(\bar{\boldsymbol{y}})$, and use this cutting-plane to update the price vector.

3.4. Decomposition with General Constraints

Up to now, we have considered the case where two problems are separable, except for some complicating variables that appear in both problems. We can also consider the case where the two subproblem are coupled through constraints that involve both set of variables. As a simple example, suppose our problem has the form :

$$\begin{aligned} \min\ & f_1(\boldsymbol{x}_1) + f_2(\boldsymbol{x}_2) \\ & \boldsymbol{x}_1 \in C_1, \quad \boldsymbol{x}_2 \in C_2 \\ & \boldsymbol{h}_1(\boldsymbol{x}_1) + \boldsymbol{h}_2(\boldsymbol{x}_2) \leq \boldsymbol{0}. \end{aligned} \tag{3.4.1}$$

Here C_1 and C_2 are feasible sets of the subproblems. The function $\boldsymbol{h}_1 : \Re^n \to \Re^p$ and $\boldsymbol{h}_2 : \Re^n \to \Re^p$ have components that are convex. The subproblems are coupled through p constraints that involve both $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. We refer to these as complicating constrains (since without them, the problem involving $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ can be solved separately).

3.4.1. Primal Decomposition

To use primal decomposition, we can introduce a variable $\boldsymbol{t} \in \Re^p$ that represent the amount of the resources allocated to the first subproblem. As a result, $-\boldsymbol{t}$ is allocated to the second subproblem. Then subproblems are,

$$\begin{aligned} \min_{\boldsymbol{x}_i} & f_i(\boldsymbol{x}_i) \\ & \boldsymbol{h}_i(\boldsymbol{x}_i) \leq (-1)^{i+1}\boldsymbol{t} \quad (i = 1, 2), \\ & \boldsymbol{x}_i \in C_i. \end{aligned} \tag{3.4.2}$$

Let $g_i(\boldsymbol{t})$ denote the optimal value of the subproblem (3.4.2). Evidently the original problem (3.4.1) is equivalent to the master problem of

$$\min_{\boldsymbol{t}} g(\boldsymbol{t}) = g_1(\boldsymbol{t}) + g_2(\boldsymbol{t}),$$

over the allocation vector $\boldsymbol{t}$. Subproblems in (3.4.2) can be solved separately, when $\boldsymbol{t}$ is fixed.

We can find a subgradient for the optimal value of each subproblem from an optimal dual variable associated with the coupling constraints. Let $p(\boldsymbol{z})$ be the optimal value of the convex optimisation problem

$$\begin{aligned} p(\boldsymbol{z}) = \min_{\boldsymbol{x}} \ & f(\boldsymbol{x}) \\ & \boldsymbol{h}(\boldsymbol{x}) \leq \boldsymbol{z} \\ & \boldsymbol{x} \in X, \quad (X \text{ is a closed convex set}), \end{aligned}$$

and suppose $\hat{\boldsymbol{z}} \in \text{dom}(p)$. Let $\hat{\boldsymbol{\mu}}$ be an optimal dual variable associated with the constraint $h(\boldsymbol{x}) \leq \hat{\boldsymbol{z}}$. Then, $-\hat{\boldsymbol{\mu}}$ is a subgradient of $p(\boldsymbol{z})$ at $\hat{\boldsymbol{z}}$. To see this, we consider the value of p at an arbitrary point $\boldsymbol{z}$ such that $\boldsymbol{z} \in \text{dom}(p)$:

$$\begin{aligned} p(\boldsymbol{z}) &= \max_{\boldsymbol{\mu} \geq 0} \min_{\boldsymbol{x} \in X} (f(\boldsymbol{x}) + \boldsymbol{\mu} \cdot (\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{z})) \\ &\geq \min_{\boldsymbol{x} \in X} (f(\boldsymbol{x}) + \hat{\boldsymbol{\mu}} \cdot (\boldsymbol{h}(\boldsymbol{x}) - \boldsymbol{z})) \\ &= \min_{\boldsymbol{x} \in X} (f(\boldsymbol{x}) + \hat{\boldsymbol{\mu}} \cdot (\boldsymbol{h}(\boldsymbol{x}) - \hat{\boldsymbol{z}} - \boldsymbol{z} + \hat{\boldsymbol{z}})) \\ &= \min_{\boldsymbol{x} \in X} (f(\boldsymbol{x}) + \hat{\boldsymbol{\mu}} \cdot (\boldsymbol{h}(\boldsymbol{x}) - \hat{\boldsymbol{z}})) + \hat{\boldsymbol{\mu}} \cdot (\hat{\boldsymbol{z}} - \boldsymbol{z}) \\ &= p(\hat{\boldsymbol{z}}) + (-\hat{\boldsymbol{\mu}}) \cdot (\boldsymbol{z} - \hat{\boldsymbol{z}}). \end{aligned}$$

It follows that $-\hat{\boldsymbol{\mu}}$ is a subgradient of p at $\hat{\boldsymbol{z}}$ (see[8]). Consequently, in order to find a subgradient of g, we solve the two subproblems, we find the optimal vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, as well as the optimal dual variables $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$, associated with the constraints $\boldsymbol{h}_1(\boldsymbol{x}_1) \leq \boldsymbol{t}$ and $\boldsymbol{h}_2(\boldsymbol{x}_2) \leq -\boldsymbol{t}$, respectively. Then we have that $\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1 \in \partial g(\boldsymbol{t})$. It is also possible that $\boldsymbol{t} \notin \text{dom}(g)$. In this case we can generate a cutting plane that separates $\boldsymbol{t}$ from $\text{dom}(g)$, for use in the master algorithm.

Primal decomposition, using a subgradient method algorithm, can be achieved following the next steps:

Repeat

- Solve the subproblems in (3.4.2), possibly in parallel.

  - Find an optimal $\boldsymbol{x}_1$ and $\boldsymbol{\mu}_1^k$.
  - Find an optimal $\boldsymbol{x}_2$ and $\boldsymbol{\mu}_2^k$.

- Update resource allocation,

$$\boldsymbol{t}^{k+1} = \boldsymbol{t}^k - \alpha^k(\boldsymbol{\mu}_2^k - \boldsymbol{\mu}_1^k).$$

  with α^k an appropriate step size. At every step of this algorithm we have points that are feasible for the original problem.

3.4.2. Dual Decomposition

Dual decomposition for this example is straightforward. We form the partial Lagrangian,

$$\begin{aligned} L(\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{\mu}) &= f_1(\boldsymbol{x}_1) + f_2(\boldsymbol{x}_2) + \boldsymbol{\mu} \cdot (\boldsymbol{h}_1(\boldsymbol{x}_1) + \boldsymbol{h}_2(\boldsymbol{x}_2)) \\ &= (f_1(\boldsymbol{x}_1) + \boldsymbol{\mu} \cdot \boldsymbol{h}_1(\boldsymbol{x}_1)) + (f_2(\boldsymbol{x}_2) + \boldsymbol{\mu} \cdot \boldsymbol{h}_2(\boldsymbol{x}_2)), \end{aligned}$$

which is separable for a some $\boldsymbol{\mu}$, so we can minimize over $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ separately, given the dual variable $\boldsymbol{\mu}$ to find $q(\boldsymbol{\mu}) = q_1(\boldsymbol{\mu}) + q_2(\boldsymbol{\mu})$. For example, to find $q_i(\boldsymbol{\mu})$; $(i = 1, 2)$, we solve the subproblem

$$\begin{aligned} q_i(\boldsymbol{\mu}) = \min\ & f_i(\boldsymbol{x}_i) + \boldsymbol{\mu} \cdot \boldsymbol{h}_i(\boldsymbol{x}_i) \\ & \boldsymbol{x}_i \in C_i. \end{aligned} \tag{3.4.3}$$

A subgradient of q at $\boldsymbol{\mu}$ is $\boldsymbol{h}_1(\boldsymbol{x}_1) + \boldsymbol{h}_2(\boldsymbol{x}_2)$, where $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are any solutions of subproblems. The master algorithm updates $\boldsymbol{\mu}$ based on this subgradient.

If we use a projected subgradient method to update $\boldsymbol{\mu}$ we get a very simple algorithm.

Repeat

- Solve the subproblems in (3.4.3), possibly in parallel.
  - Find an optimal $\boldsymbol{x}_1^k$
  - Find an optimal $\boldsymbol{x}_2^k$
- Update dual variables:

$$\boldsymbol{\mu}^{k+1} = (\boldsymbol{\mu}^k + \beta^k(\boldsymbol{h}_1(\boldsymbol{x}_1^k) + \boldsymbol{h}_2(\boldsymbol{x}_2^k)))_+.$$

At each step we have a lower bound on P^*, given by

$$q(\boldsymbol{\mu}) = q_1(\boldsymbol{\mu}) + q_2(\boldsymbol{\mu}) = f_1(\boldsymbol{x}_1) + f_2(\boldsymbol{x}_2) + \boldsymbol{\mu} \cdot (\boldsymbol{h}_1(\boldsymbol{x}_1) + \boldsymbol{h}_2(\boldsymbol{x}_2)).$$

The iterates in the dual decomposition method need not be feasible, i.e., we can have $\boldsymbol{h}_1(\boldsymbol{x}_1) + \boldsymbol{h}_2(\boldsymbol{x}_2) \leq \mathbf{0}$. At the cost of solving two additional subproblems, however, we can (often) construct a feasible set of variables, which will give us an upper bound on P^*. When $\boldsymbol{h}_1(\boldsymbol{x}_1) + \boldsymbol{h}_2(\boldsymbol{x}_2) \not\leq \mathbf{0}$, we define

$$\boldsymbol{t} = \frac{\boldsymbol{h}_1(\boldsymbol{x}_1) - \boldsymbol{h}_2(\boldsymbol{x}_2)}{2}, \tag{3.4.4}$$

and solve the primal subproblems (3.4.2). As in primal decomposition, it can happen that $\boldsymbol{t} \notin \mathrm{dom} g$. But when $\boldsymbol{t} \in \mathrm{dom} g$, this method gives a feasible point, and an upper bound on P^*.

Note that for the dual problem we can find a subgradient as follows. Consider the following optimisation problem in general:

$$\begin{aligned} \min_{\boldsymbol{x}} \ & f(\boldsymbol{x}) \\ & \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0} \\ & \boldsymbol{h}(\boldsymbol{x}) = \mathbf{0} \\ & \boldsymbol{x} \in X, \quad X \text{ is a nonempty set.} \end{aligned}$$

Its dual problem is given by,

$$\begin{aligned} \max_{\boldsymbol{\mu},\boldsymbol{\lambda}} \ & q(\boldsymbol{\mu}, \boldsymbol{\lambda}) \\ & \boldsymbol{\mu} \geq \mathbf{0}, \boldsymbol{\lambda} \text{ free,} \end{aligned}$$

where $q(\boldsymbol{\mu}, \boldsymbol{\lambda})$ is defined as follows:

$$q(\boldsymbol{\mu}, \boldsymbol{\lambda}) = \min_{\boldsymbol{x} \in X} L(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\lambda})$$

with

$$L(\boldsymbol{x}; \boldsymbol{\mu}, \lambda) = f(\boldsymbol{x}) + \boldsymbol{\mu} \cdot \boldsymbol{g}(\boldsymbol{x}) + \boldsymbol{\lambda} \cdot \boldsymbol{h}(\boldsymbol{x}) = f(\boldsymbol{x}) + \begin{Bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\lambda} \end{Bmatrix} \cdot \begin{Bmatrix} \boldsymbol{g}(\boldsymbol{x}) \\ \boldsymbol{h}(\boldsymbol{x}) \end{Bmatrix}.$$

Assume that for $(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\lambda}})$, some vector $\widehat{\boldsymbol{x}}$ minimizes $L(\boldsymbol{x}, \widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\lambda}})$ over X that is:

$$q(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\lambda}}) = \min_{\boldsymbol{x} \in X} L(\boldsymbol{x}; \widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\lambda}}) = f(\widehat{\boldsymbol{x}}) + \begin{Bmatrix} \widehat{\boldsymbol{\mu}} \\ \widehat{\boldsymbol{\lambda}} \end{Bmatrix} \cdot \begin{Bmatrix} \boldsymbol{g}(\widehat{\boldsymbol{x}}) \\ \boldsymbol{h}(\widehat{\boldsymbol{x}}) \end{Bmatrix},$$

then we have:

$$\begin{aligned} q(\boldsymbol{\mu}, \boldsymbol{\lambda}) &\leq f(\widehat{\boldsymbol{x}}) + \begin{Bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\lambda} \end{Bmatrix} \cdot \begin{Bmatrix} \boldsymbol{g}(\widehat{\boldsymbol{x}}) \\ \boldsymbol{h}(\widehat{\boldsymbol{x}}) \end{Bmatrix} \\ &= f(\widehat{\boldsymbol{x}}) + \begin{Bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\lambda} \end{Bmatrix} \cdot \begin{Bmatrix} \boldsymbol{g}(\widehat{\boldsymbol{x}}) \\ \boldsymbol{h}(\widehat{\boldsymbol{x}}) \end{Bmatrix} + \begin{Bmatrix} \widehat{\boldsymbol{\mu}} \\ \widehat{\boldsymbol{\lambda}} \end{Bmatrix} \cdot \begin{Bmatrix} \boldsymbol{g}(\widehat{\boldsymbol{x}}) \\ \boldsymbol{h}(\widehat{\boldsymbol{x}}) \end{Bmatrix} - \begin{Bmatrix} \widehat{\boldsymbol{\mu}} \\ \widehat{\boldsymbol{\lambda}} \end{Bmatrix} \cdot \begin{Bmatrix} \boldsymbol{g}(\widehat{\boldsymbol{x}}) \\ \boldsymbol{h}(\widehat{\boldsymbol{x}}) \end{Bmatrix} \\ &= q(\widehat{\boldsymbol{\mu}}, \widehat{\boldsymbol{\lambda}}) + \begin{Bmatrix} \boldsymbol{g}(\widehat{\boldsymbol{x}} \\ \boldsymbol{h}(\widehat{\boldsymbol{x}} \end{Bmatrix} \cdot (\begin{Bmatrix} \boldsymbol{\mu} \\ \boldsymbol{\lambda} \end{Bmatrix} - \begin{Bmatrix} \widehat{\boldsymbol{\mu}} \\ \widehat{\boldsymbol{\lambda}} \end{Bmatrix}). \end{aligned}$$

It means that:

$$\begin{Bmatrix} \boldsymbol{g}(\widehat{\boldsymbol{x}}) \\ \boldsymbol{h}(\widehat{\boldsymbol{x}}) \end{Bmatrix} \in \partial q(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\lambda}}).$$

Except for the details of computing the relevant subgradients, primal and dual decomposition for problems with coupling variables and coupling constraints seem quite similar. In fact, we can readily transform each into the other. For example, we can start with the problem with coupling constraints (3.4.1), and introduce new variables $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, that bound the subsystem coupling constraint functions, to obtain

$$\begin{aligned} \min_{\boldsymbol{x}_1, \boldsymbol{y}_1, \boldsymbol{x}_2, \boldsymbol{y}_2} \quad & f_1(\boldsymbol{x}_1) + f_2(\boldsymbol{x}_2) \\ & \boldsymbol{h}_1(\boldsymbol{x}_1) \leq \boldsymbol{y}_1 \\ & \boldsymbol{h}_2(\boldsymbol{x}_2) \leq \boldsymbol{y}_2 \\ & \boldsymbol{y}_1 + \boldsymbol{y}_2 = 0 \\ & \boldsymbol{x}_1 \in C_1, \quad \boldsymbol{x}_2 \in C_2. \end{aligned} \tag{3.4.5}$$

We now have a problem that is separable, except for a consistency constraint, that requires two (vector) variables of the subproblems to be equal.

Any problem that can be decomposed into two subproblems that are coupled by some common variables, or equality or inequality constraints, can be put in this standard form, i.e., two subproblems that are independent except for one consistency constraint, that requires a subvariable of one to be equal to a subvariable of the other. Primal or dual decomposition is then readily applied; only the details of computing the needed subgradients for the master problem vary from problem to problem.

3.5. Decomposition with Linear Constraints

In this section we apply decomposition methods described previously for a toy linear problem and we compare our results for different choices of the step sizes (α or β).

3.5.1. Splitting Primal and Dual Variables

Let us consider the parallelisation of the following linear optimisation problem:

$$\begin{aligned} (P)\ \boldsymbol{c}\cdot\boldsymbol{x}^* = \min_{\boldsymbol{x}}\ & \boldsymbol{c}\cdot\boldsymbol{x} \\ & \mathbf{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \geq \mathbf{0}, \end{aligned} \tag{3.5.1}$$

whose dual is:

$$\begin{aligned} (D)\ \boldsymbol{b}\cdot\boldsymbol{y}^* = \max_{\boldsymbol{y}}\ & \boldsymbol{b}\cdot\boldsymbol{y} \\ & \mathbf{A}^T\boldsymbol{y} \leq \boldsymbol{c}. \end{aligned}$$

3.5.2. Primal Decomposition

We split the primal $\boldsymbol{x}$ variables as $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ which allows us to rewrite the problem in (3.5.1) as,

$$\begin{aligned} \min_{\boldsymbol{x}_1,\boldsymbol{x}_2}\ & \boldsymbol{c}_1\cdot\boldsymbol{x}_1 + \boldsymbol{c}_2\cdot\boldsymbol{x}_2 \\ & \mathbf{A}_1\boldsymbol{x}_1 + \mathbf{A}_2\boldsymbol{x}_2 = \boldsymbol{b} = \boldsymbol{b}_1 + \boldsymbol{b}_2 \\ & \boldsymbol{x}_1 \geq \mathbf{0}\ ,\ \boldsymbol{x}_2 \geq \mathbf{0}. \end{aligned} \tag{3.5.2}$$

The problem above is equivalent to

$$\begin{aligned} \min_{\boldsymbol{t}}\ \min_{\boldsymbol{x}_1,\boldsymbol{x}_2}\ & \boldsymbol{c}_1\cdot\boldsymbol{x}_1 + \boldsymbol{c}_2\cdot\boldsymbol{x}_2 \\ & \mathbf{A}_1\boldsymbol{x}_1 - \boldsymbol{t} = \boldsymbol{b}_1 \\ & \mathbf{A}_2\boldsymbol{x}_2 + \boldsymbol{t} = \boldsymbol{b}_2 \\ & \boldsymbol{x}_1 \geq \mathbf{0},\ \boldsymbol{x}_2 \geq \mathbf{0},\ \boldsymbol{t} \text{ is free.} \end{aligned} \tag{3.5.3}$$

This problem can be written as $\min_{\boldsymbol{t}} f_1(\boldsymbol{t}) + f_2(\boldsymbol{t})$ where $f_1(\boldsymbol{t})$ and $f_2(\boldsymbol{t})$

are the solution of the following subproblems:

$$\begin{aligned} f_1(\boldsymbol{t}) = \min_{\boldsymbol{x}_1} & \boldsymbol{c}_1 \cdot \boldsymbol{x}_1 & f_2(\boldsymbol{t}) = \min_{\boldsymbol{x}_2} & \boldsymbol{c}_2 \cdot \boldsymbol{x}_2 \\ & \mathbf{A}_1\boldsymbol{x}_1 = \boldsymbol{b}_1 + \boldsymbol{t} & & \mathbf{A}_2\boldsymbol{x}_2 = \boldsymbol{b}_2 - \boldsymbol{t} \\ & \boldsymbol{x}_1 \geq \mathbf{0}, & & \boldsymbol{x}_2 \geq \mathbf{0}, \end{aligned} \tag{3.5.4}$$

the Lagrangian functions of these subproblems are given by:

$$\begin{aligned} L_1(\boldsymbol{x}_1, \boldsymbol{t}; \boldsymbol{y}_1, \boldsymbol{w}_1) &= \boldsymbol{c}_1 \cdot \boldsymbol{x}_1 + \boldsymbol{y}_1 \cdot (\boldsymbol{b}_1 + \boldsymbol{t} - \boldsymbol{A}_1\boldsymbol{x}_1) - \boldsymbol{w}_1 \cdot \boldsymbol{x}_1, \\ L_2(\boldsymbol{x}_2, \boldsymbol{t}; \boldsymbol{y}_2, \boldsymbol{w}_2) &= \boldsymbol{c}_2 \cdot \boldsymbol{x}_2 + \boldsymbol{y}_2 \cdot (\boldsymbol{b}_2 - \boldsymbol{t} - \mathbf{A}_2\boldsymbol{x}_2) - \boldsymbol{w}_2 \cdot \boldsymbol{x}_2. \end{aligned}$$

For a fixed master variables $\boldsymbol{t}$, the optimum values of the slave variables $\boldsymbol{x}_i$ may be obtained as the solution of the following two subproblems, $(i = 1, 2)$,

$$\begin{aligned} f_i(\boldsymbol{t}) = \min_{\boldsymbol{x}_i} & \boldsymbol{c}_i \cdot \boldsymbol{x}_i \\ & \mathbf{A}_i\boldsymbol{x}_i = \boldsymbol{b}_i + (-1)^{i+1}\boldsymbol{t} \\ & \boldsymbol{x}_i \geq 0. \end{aligned}$$

The Lagrangian function of problem (3.5.3) is given by:

$$\begin{aligned} L(\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{w}_1, \boldsymbol{w}_2) &= \boldsymbol{c}_1 \cdot \boldsymbol{x}_1 + \boldsymbol{c}_2 \cdot \boldsymbol{x}_2 + \boldsymbol{y}_1 \cdot (\boldsymbol{b}_1 + \boldsymbol{t} - \mathbf{A}_1\boldsymbol{x}_1) \\ &\quad + \boldsymbol{y}_2 \cdot (\boldsymbol{b}_2 - \boldsymbol{t} - \mathbf{A}_2\boldsymbol{x}_2) - \boldsymbol{w}_1 \cdot \boldsymbol{x}_1 - \boldsymbol{w}_2 \cdot \boldsymbol{x}_2 \\ &= \boldsymbol{c}_1 \cdot \boldsymbol{x}_1 + \boldsymbol{y}_1 \cdot (\boldsymbol{b}_1 - \mathbf{A}_1\boldsymbol{x}_1) + \boldsymbol{c}_2 \cdot \boldsymbol{x}_2 \\ &\quad + \boldsymbol{y}_2 \cdot (\boldsymbol{b}_2 - \mathbf{A}_1\boldsymbol{x}_1) + \boldsymbol{t} \cdot (\boldsymbol{y}_1 - \boldsymbol{y}_2) \\ &= L_1(\boldsymbol{x}_1, \boldsymbol{t}; \boldsymbol{y}_1, \boldsymbol{w}_1) + L_2(\boldsymbol{x}_2, \boldsymbol{t}; \boldsymbol{y}_2, \boldsymbol{w}_2), \end{aligned}$$

with

$$L_i(\boldsymbol{x}_i, \boldsymbol{t}; \boldsymbol{y}_i, \boldsymbol{w}_i) = \boldsymbol{c}_i \cdot \boldsymbol{x}_1 + \boldsymbol{y}_i \cdot (\boldsymbol{b}_i + (-1)^{i+1}\boldsymbol{t} - \mathbf{A}_1\boldsymbol{x}_1) - \boldsymbol{w}_i \cdot \boldsymbol{x}_i, \ (i = 1, 2).$$

It then follows that we can rewrite the optimum primal objective $\boldsymbol{c} \cdot \boldsymbol{x}^*$ as,

$$\boldsymbol{c} \cdot \boldsymbol{x}^* = \boldsymbol{c}_1 \cdot \boldsymbol{x}_1^* + \boldsymbol{c}_2 \cdot \boldsymbol{x}_2^* = \min_{\boldsymbol{t}} \sum_{i=1}^{2} \min_{\boldsymbol{x}_i} \max_{\boldsymbol{y}_i, \boldsymbol{w}_i} L_i(\boldsymbol{x}_i, \boldsymbol{t}; \boldsymbol{y}_i, \boldsymbol{w}_i).$$

After observing the equation above, we have that $\nabla_{\boldsymbol{t}} L = (\boldsymbol{y}_1 - \boldsymbol{y}_2)$, which allows us to update the master variables with the following descent method,

$$\boldsymbol{t}^{k+1} = \boldsymbol{t}^k - \alpha^k(\boldsymbol{y}_1^k - \boldsymbol{y}_2^k) = \boldsymbol{t}^k + \alpha^k(\boldsymbol{y}_2^k - \boldsymbol{y}_1^k).$$

We point out that the feasibility of the subproblems in (3.5.4) may be affected by the value of α_k. Given a descent direction $\boldsymbol{y}_2^k - \boldsymbol{y}_1^k$, the maximum of α_k that preserves the feasibility of each subproblem may be obtained by solving the following optimisation problems, $(i = 1, 2)$:

$$\begin{aligned} \alpha_i^k = \max_{\alpha} \ & \alpha \\ & \mathbf{A}_i \boldsymbol{x}_i = \boldsymbol{b}_i + (-1)^{i+1}(\boldsymbol{t}^k + \alpha(\boldsymbol{y}_2^k - \boldsymbol{y}_1^k)) \\ & \boldsymbol{x}_i \geq \mathbf{0}, \alpha \geq 0, \end{aligned}$$

and use $\alpha = \min(\alpha_1^k, \alpha_2^k)$.We can set $\alpha_k = b\alpha$ where b is the step size coefficient.

3.5.3. Dual Decomposition

We recall the same problem in (3.5.1) with the splitting in (3.5.2). Dual decomposition for this example is straightforward. We form the Lagrangian as,

$$\begin{aligned} L(\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{y}, \boldsymbol{w}_1, \boldsymbol{w}_2) &= \boldsymbol{c}_1 \cdot \boldsymbol{x}_1 + \boldsymbol{c}_2 \cdot \boldsymbol{x}_2 + \boldsymbol{y} \cdot (\boldsymbol{b}_1 - \mathbf{A}_1\boldsymbol{x}_1 + \boldsymbol{b}_2 - \mathbf{A}_2\boldsymbol{x}_2) \\ &\quad - \boldsymbol{w}_1 \cdot \boldsymbol{x}_1 - \boldsymbol{w}_2 \cdot \boldsymbol{x}_2 \\ &= (\boldsymbol{c}_1 \cdot \boldsymbol{x}_1 + \boldsymbol{y} \cdot (\boldsymbol{b}_1 - \mathbf{A}_1\boldsymbol{x}_1) - \boldsymbol{w}_1 \cdot \boldsymbol{x}_1) \\ &\quad + (\boldsymbol{c}_2 \cdot \boldsymbol{x}_2 + \boldsymbol{y} \cdot (\boldsymbol{b}_2 - \mathbf{A}_2\boldsymbol{x}_2) - \boldsymbol{w}_2^T \boldsymbol{x}_2), \end{aligned}$$

so we can minimize over $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ separately given the dual variable $\boldsymbol{y}$, to find $g(\boldsymbol{y}) = g_1(\boldsymbol{y}) + g_2(\boldsymbol{y})$ where $g(\boldsymbol{y})$ is given as,

$$g(\boldsymbol{y}) = \min_{\boldsymbol{x}_1, \boldsymbol{x}_2} L(\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{y}, \boldsymbol{w}_1, \boldsymbol{w}_2) = \min_{\boldsymbol{x}_1, \boldsymbol{x}_2} L_1(\boldsymbol{x}_1; \boldsymbol{y}, \boldsymbol{w}_1) + L_2(\boldsymbol{x}_2; \boldsymbol{y}, \boldsymbol{w}_2).$$

In order to find $g_1(\boldsymbol{y})$ and $g_2(\boldsymbol{y})$, respectively, we solve the following two subproblems:

$$g_1(\boldsymbol{y}) = \min_{\boldsymbol{x}_1 \geq 0} \boldsymbol{c}_1 \cdot \boldsymbol{x}_1 + \boldsymbol{y} \cdot (\boldsymbol{b}_1 - \mathbf{A}_1\boldsymbol{x}_1) = \min_{\boldsymbol{x}_1 \geq 0} (\boldsymbol{c}_1 - \mathbf{A}_1^T \boldsymbol{y})\boldsymbol{x}_1 + \boldsymbol{y} \cdot \boldsymbol{b}_1.$$

$$g_2(\boldsymbol{y}) = \min_{\boldsymbol{x}_2 \geq 0} \boldsymbol{c}_2 \cdot \boldsymbol{x}_2 + \boldsymbol{y} \cdot (\boldsymbol{b}_2 - \mathbf{A}_2\boldsymbol{x}_2) = \min_{\boldsymbol{x}_2 \geq 0} (\boldsymbol{c}_2 - \mathbf{A}_2^T \boldsymbol{y})\boldsymbol{x}_2 + \boldsymbol{y} \cdot \boldsymbol{b}_2.$$

The master algorithm updates $\boldsymbol{y}$ based on subgradient as follows

$$\boldsymbol{y}^{k+1} = \boldsymbol{y}^k + \beta^k(\boldsymbol{b}_1 - \mathbf{A}_1\boldsymbol{x}_1 + \boldsymbol{b}_2 - \mathbf{A}_2\boldsymbol{x}_2) = \boldsymbol{y}^k + \beta^k(\boldsymbol{b} - \mathbf{A}\boldsymbol{x}).$$

As before, this update may yield infeasible dual variables $\boldsymbol{y}^{k+1}$. For this reason, the value of β^k may be found by solving the following optimisation problems:

$$\max_{\beta_i} 0$$
$$\boldsymbol{c}_i^T - \mathbf{A}_i^T(\boldsymbol{y}^k + \beta_i(\boldsymbol{b} - \mathbf{A}\boldsymbol{x})) \geq \mathbf{0},\ i = 1, 2,$$

and choose $\beta = b\min(\beta_1, \beta_2)$ where b is a coefficient,$(b \in (0, 1))$. We point out that in fact, the optimal solution of the problem above can be simply obtained by computing β_i^k as,

$$\beta_i^k = \min_j \frac{(\boldsymbol{c}_i - \mathbf{A}_i^T\boldsymbol{y}^k)_j}{(\boldsymbol{b} - \mathbf{A}\boldsymbol{x})_j} \tag{3.5.5}$$

where $(a)_j$ denotes component j of vector $\boldsymbol{a}$ and $\mathbf{A}_i^T$ is row i of matrix $\mathbf{A}^T$.

Subgradient methods are the simplest and among the most popular methods for dual optimisation. As it is known they generate a sequence of dual feasibility points, using a single subgradient at each iteration. As explained in previous sections, the simplest type of subgradient method is given by:

$$\boldsymbol{\mu}^{k+1} = P_X(\boldsymbol{\mu}^k + \beta^k\boldsymbol{g}^k),$$

where $\boldsymbol{g}^k$ denotes the subgradient $\boldsymbol{g}(\boldsymbol{x}_{\mu^k})$, $P_X(.)$ denotes projection on the closed convex set X, and β^k is a positive scalar step size. Note that when the dual function is differentiable, the new iteration improve the dual cost but when it is not differentiable, the new iteration may not improve the dual cost for all values of step size; i.e. for some k, we may have

$$q(P_X(\boldsymbol{\mu}^k + \beta\boldsymbol{g}^k)) < q(\boldsymbol{\mu}^k), \quad \forall\, \beta > 0.$$

However, if the step size is small enough, the distance of the current iterate to the optimal solution set is reduced.

The following provides a formal proof and also provides an estimate for the range of appropriate step sizes. We have

$$\begin{aligned}\|\boldsymbol{\mu}^{k+1} - \boldsymbol{\mu}^*\|^2 &= \|\boldsymbol{\mu}^k + \beta^k\boldsymbol{g}^k - \boldsymbol{\mu}^*\|^2 \\ &= \|\boldsymbol{\mu}^k - \boldsymbol{\mu}^*\|^2 - 2\beta^k(\boldsymbol{\mu}^* - \boldsymbol{\mu}^k)^T\boldsymbol{g}^k + (\beta^k)^2\|\boldsymbol{g}^k\|^2,\end{aligned}$$

and by using the subgradient inequality,

$$(\boldsymbol{\mu}^* - \boldsymbol{\mu}^k)^k\boldsymbol{g}^k \geq q(\boldsymbol{\mu}^*) - q(\boldsymbol{\mu}^k),$$

we obtain

$$\|\boldsymbol{\mu}^{k+1} - \boldsymbol{\mu}^*\|^2 \leq \|\boldsymbol{\mu}^k - \boldsymbol{\mu}^*\|^2 - 2\beta^k(q(\boldsymbol{\mu}^*) - q(\boldsymbol{\mu}^k)) + (\beta^k)^2\|\boldsymbol{g}^k\|^2.$$

If $-2\beta^k(q(\boldsymbol{\mu}^*) - q(\boldsymbol{\mu}^k)) + (\beta^k)^2\|\boldsymbol{g}^k\|^2 < 0$ that is:

$$0 < \beta^k < \frac{2(q(\boldsymbol{\mu}^*) - q(\boldsymbol{\mu}^k))}{\|\boldsymbol{g}^k\|^2},$$

where $q(\boldsymbol{\mu}^*) = q^*$ is the optimal dual value, we have that:

$$\|\boldsymbol{\mu}^{k+1} - \boldsymbol{\mu}^*\| < \|\boldsymbol{\mu}^k - \boldsymbol{\mu}^*\|.$$

In practice the value of q^* is not known, in which case we estimate q^* with value q^k, and choose β^k:

$$\begin{aligned} &\beta^k = \gamma^k \frac{q^k - q(\boldsymbol{\mu}^k)}{\|\boldsymbol{g}^k\|^2 + 1} \\ &0 < \gamma_1 \leq \gamma^k \leq \gamma_2 < 2, \quad \forall\, k \geq 1. \end{aligned} \tag{3.5.6}$$

The value q^k is equal to the best function value $\max_{1\leq j\leq k} q(\boldsymbol{\mu}^j)$ achieved up to the kth iteration plus a positive amount δ^k which is adjusted based on the progress of the algorithm, i.e. $q^k = \max_{1\leq j\leq k} q(\boldsymbol{\mu}_j) + \delta^k$.

The parameter δ^k is updated according to

$$\delta^{k+1} = \begin{cases} \rho\delta^k & \text{if } q(\boldsymbol{\mu}^{k+1}) > q^k, \\ \max\{b\delta^k, \delta\} & \text{if } q(\boldsymbol{\mu}^{k+1}) \leq q^k, \end{cases} \tag{3.5.7}$$

where δ, b and ρ are fixed positive constants with $b < 1$ and $\rho \geq 1$. This step size rule is called *Dynamic step size rule* [9].

3.5.4. Benders Decomposition

Benders decomposition is a method for solving certain largescale optimisation problems. Instead of considering all decision variables and constraints of a largescale problem simultaneously, Benders decomposition partitions the problem into multiple smaller problems. Benders decomposition is a useful technique when dealing with complication variables. We recall the

partitioned problem in (3.5.3),

$$\begin{array}{lll} (P)\ \min\limits_{\boldsymbol{t}}\ \min\limits_{\boldsymbol{x}_1,\boldsymbol{x}_2} \boldsymbol{c}_1\cdot\boldsymbol{x}_1+\boldsymbol{c}_2\cdot\boldsymbol{x}_2 & & (D)\ \max\limits_{\boldsymbol{y}_1,\boldsymbol{y}_2} \boldsymbol{b}_1\cdot\boldsymbol{y}_1+\boldsymbol{b}_2\cdot\boldsymbol{y}_2 \\ \mathbf{A}_1\boldsymbol{x}_1=\boldsymbol{b}_1+\boldsymbol{t} & & \mathbf{A}_1^T\boldsymbol{y}_1\le\boldsymbol{c}_1 \\ \mathbf{A}_2\boldsymbol{x}_2=\boldsymbol{b}_2-\boldsymbol{t} & & \mathbf{A}_2^T\boldsymbol{y}_2\le\boldsymbol{c}_2 \\ \boldsymbol{x}_1\ge 0,\ \boldsymbol{x}_2\ge 0,\ \boldsymbol{t}\text{ is free.} & & \boldsymbol{y}_1=\boldsymbol{y}_2. \end{array} \tag{3.5.8}$$

and rewrite it as the following master problem:

$$\min_{\boldsymbol{t}\ \text{free}} \alpha_1(\boldsymbol{t})+\alpha_2(\boldsymbol{t})=\alpha(\boldsymbol{t}) \tag{3.5.9}$$

and the two following subproblems,$(i=1,2)$

$$\begin{array}{lll} \alpha_i(\boldsymbol{t})= \min\limits_{\boldsymbol{x}_i} \boldsymbol{c}_i\cdot\boldsymbol{x}_i & & =\max\limits_{\boldsymbol{y}_i}(\boldsymbol{b}_i+(-1)^{i+1}\boldsymbol{t})\cdot\boldsymbol{y}_i \\ \mathbf{A}_i\boldsymbol{x}_i=\boldsymbol{b}_i+(-1)^{i+1}\boldsymbol{t} & & A_i^T\boldsymbol{y}_i\le\boldsymbol{c}_i \\ \boldsymbol{x}_i\ge 0 & & \boldsymbol{y}_i\text{ free.} \end{array} \tag{3.5.10}$$

For a given (nonoptimal) $\boldsymbol{t}^k\in \text{dom}\alpha$, we find optimal values $\boldsymbol{x}_i^k$ and $\boldsymbol{y}_i^k$ of the subproblems in (3.5.10). Since we are minimizing on $\boldsymbol{t}$, we have that $\sum_i \boldsymbol{c}_i\cdot\boldsymbol{x}_i^k=\alpha_1(\boldsymbol{t}^k)+\alpha_1(\boldsymbol{t}^k)=\alpha(\boldsymbol{t}^k)$ is an upper bound of $\boldsymbol{c}\cdot\boldsymbol{x}^*$, i.e.

$$\boldsymbol{c}\cdot\boldsymbol{x}^*\le\sum_i \boldsymbol{c}_i\cdot\boldsymbol{x}_i\cdot=\alpha_1(\boldsymbol{t}^k)+\alpha_1(\boldsymbol{t}^k)=\alpha(\boldsymbol{t}^k). \tag{3.5.11}$$

For given values of $\boldsymbol{x}_i^k$ and $\boldsymbol{y}_i^k$, we compute optimal values of the master problem in (3.5.9), but imposing only the feasibility of the dual subproblems. Since the dual variables $\boldsymbol{y}_i^k$ are feasible, and the feasibility region of the dual subproblem is independent of $\boldsymbol{t}$, the master problem is rewritten as:

$$\begin{array}{l} \min\limits_{\alpha_1,\alpha_2,\boldsymbol{t}} \alpha_1+\alpha_2 \\ \qquad (\boldsymbol{b}_1+\boldsymbol{t})\cdot\boldsymbol{y}_1^k\le\alpha_1 \\ \qquad (\boldsymbol{b}_2-\boldsymbol{t})\cdot\boldsymbol{y}_2^k\le\alpha_2 \\ \qquad \boldsymbol{t}\text{ free.} \end{array} \tag{3.5.12}$$

Since we are minimizing on α_i, and the feasibility of dual subproblems is retained, the optimal values of $(\boldsymbol{b}_i+(-1)^{i+1})\cdot\boldsymbol{y}_i^k$ are lower bounds of the

subproblems, and consequently, $\sum_i \alpha_i$ is a lower bound of $\boldsymbol{c}\cdot\boldsymbol{x}^*$. The process is then stopped whenever $\sum_i \boldsymbol{c}_i \cdot \boldsymbol{x}_i^k - \sum_i \alpha_i \leq TOl$. The fact that $\sum_i \alpha_i$ is a lower bound can be also verified by analyzing the lagrangian function:

$$\begin{aligned}\boldsymbol{c}\cdot\boldsymbol{x}^* &= \max_{\boldsymbol{w}_i,\boldsymbol{y}_i}\min_{\boldsymbol{t}}\min_{\boldsymbol{x}_1,\boldsymbol{x}_2} \boldsymbol{c}_i\cdot\boldsymbol{x}_i + \boldsymbol{y}_i\cdot(\boldsymbol{b}_i - \mathbf{A}_i\boldsymbol{x}_i + (-1)^{i+1}t) - \boldsymbol{w}_i\cdot\boldsymbol{x}_i \\ &\geq \min_{\boldsymbol{t}}\min_{\boldsymbol{x}_1,\boldsymbol{x}_2} \boldsymbol{c}_i\cdot\boldsymbol{x}_i + (\boldsymbol{b}_i - \mathbf{A}_i\boldsymbol{x}_i + (-1)^{i+1}\boldsymbol{t})\cdot\boldsymbol{y}_i^k - \boldsymbol{x}_i\cdot\boldsymbol{w}_i^k\big|_{\boldsymbol{w}_i^k\geq 0} \\ &= \min_{\boldsymbol{t}}(\boldsymbol{b}_i + (-1)^{i+1}\boldsymbol{t})\cdot\boldsymbol{y}_i^k\big|_{\boldsymbol{c}_i\geq\mathbf{A}_i^T\boldsymbol{y}_i^k}.\end{aligned} \tag{3.5.13}$$

Therefore in view of (3.5.13), we have that,

$$\begin{aligned}&\max_{\substack{\mathbf{A}_1^T\boldsymbol{y}_1\leq\boldsymbol{c}_1\\ \mathbf{A}_2^T\boldsymbol{y}_2\leq\boldsymbol{c}_2}} \quad \min_{\boldsymbol{t}\in\mathrm{dom}\alpha} \boldsymbol{y}_i\cdot(\boldsymbol{b}_i + (-1)^{i+1}\boldsymbol{t}) = \\ &\max_{\substack{\mathbf{A}_1^T\boldsymbol{y}_1\leq\boldsymbol{c}_1\\ \mathbf{A}_2^T\boldsymbol{y}_2\leq\boldsymbol{c}_2}} \quad \min_{\substack{\boldsymbol{t}\in\mathrm{dom}\alpha\\ \boldsymbol{y}_1\cdot(\boldsymbol{b}_1+\boldsymbol{t})\leq\alpha_1\\ \boldsymbol{y}_2\cdot(\boldsymbol{b}_2-\boldsymbol{t})\leq\alpha_1}} \alpha_1+\alpha_2 = \boldsymbol{c}\cdot\boldsymbol{x}^*\end{aligned} \tag{3.5.14}$$

In practice, in order to increase the lower bound, the conditions in (3.5.10) are completed for all the values of $\boldsymbol{y}_i^k$ found throughout the iterations. This is a relatively small increase on the size of the master problem.

3.6. Simple Linear Example

In the following, we illustrate through an example how the step size mentioned in the previous sections affects the progress of convergence. Let us consider the parallelisation of the following *linear* optimisation problem:

$$\begin{aligned}\min_{\boldsymbol{x}} &\begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}\boldsymbol{x} \\ &\begin{bmatrix}4 & 4 & 3 & 2 & 1 & 1\\ 1 & 2 & 3 & 4 & 1 & 1\end{bmatrix}\boldsymbol{x} = \begin{bmatrix}1\\1\end{bmatrix} \\ &\boldsymbol{x}\geq 0.\end{aligned} \tag{3.6.1}$$

3.6.1. Primal Decomposition

We solve this problem through primal decomposition method described in Section 3.5.2 and update the $\boldsymbol{t}$ through subgradient method using different step sizes α_k. Figure 3.1 shows exact global optimal solution with a black

dash line, and the global optimal solution at each iteration with squares. As it may be observed, for $b = 1$ we have oscillation, and for $b = \frac{1}{10}$, oscillation is removed but the rate of convergence is severely affected.

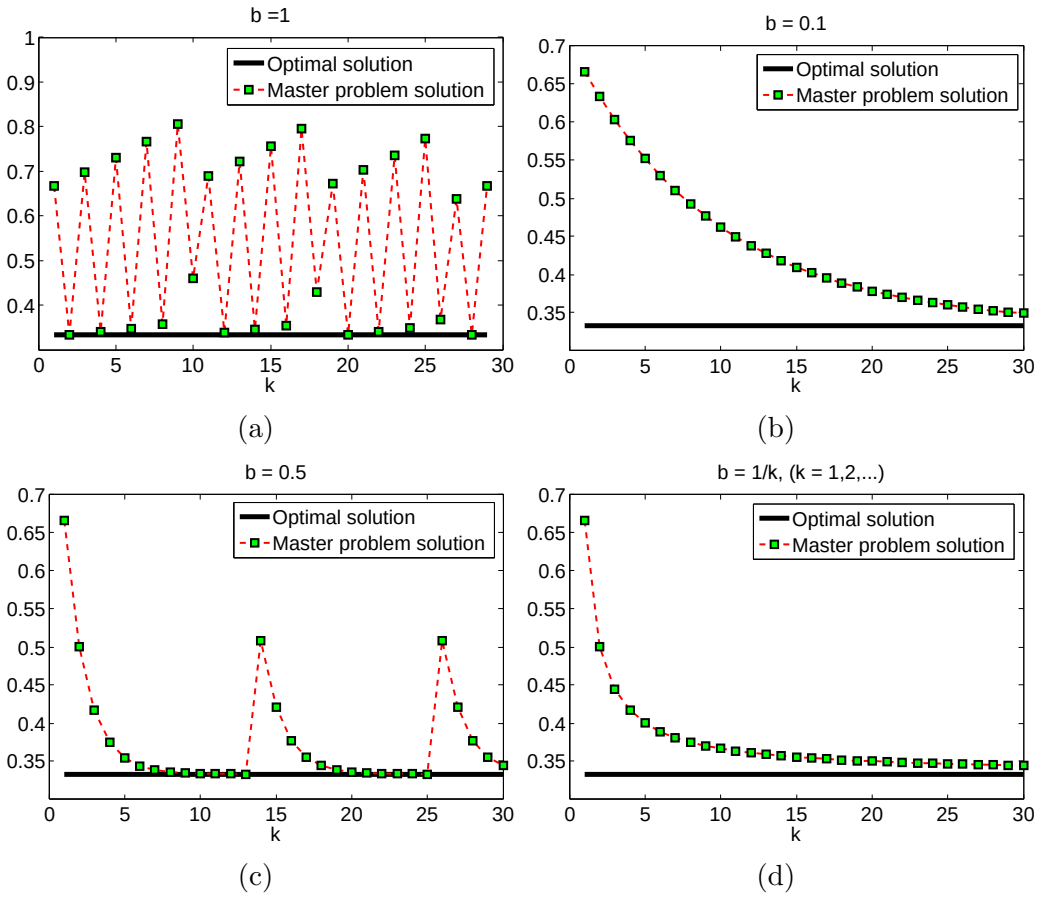


Figure 3.1.: Convergence of global objective function using primal decomposition for different rules of the step-size b.

3.6.2. Dual Decomposition

Now we illustrate dual decomposition explained in Section 3.5.3, with the same example used earlier. Like in the previous example, we choose different values for coefficient b for updating $\boldsymbol{y}$ in the master problem and compare

with each others. Figure 3.2 shows exact global optimal solution with a

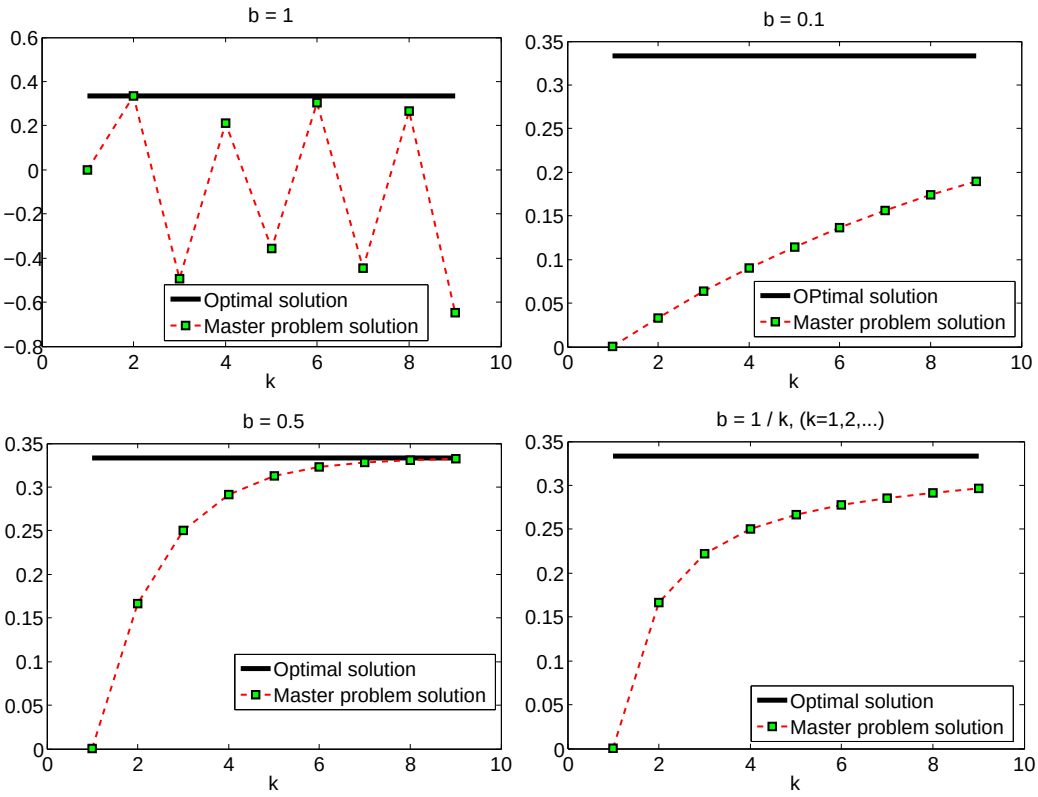


Figure 3.2.: Convergence of global objective function using dual decomposition for different rules of the step-size b.

black dash line, and global optimal solution at each iteration with squares. As it can be observed the convergence is strongly affected by the value of b, the larger it is, the more oscillations are obtained, and the smaller it is, the slower is the convergence rate. For $b = \frac{1}{2}$ has been found a reasonable compromise, although the value $b = \frac{1}{j}$ ensure convergence [9].

We illustrate the use of the dynamic step size rule with the same example shown in Section 3.6. In figure 3.3 the black dash line indicates global optimal solution for the original problem, the white squares indicate the global optimal solution at each iteration obtained from the sum of the optimal solutions of subproblems, i.e. $q(\boldsymbol{\mu}) = q_1(\boldsymbol{\mu}) + q_2(\boldsymbol{\mu})$, and black squares

correspond to q^k.

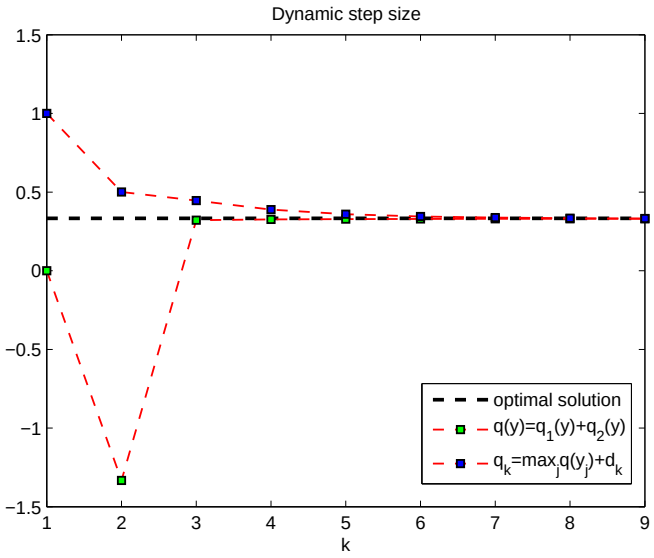


Figure 3.3.: Dual decomposition using dynamic step size rule.

3.6.3. Benders Decomposition

We illustrate Benders decomposition with the same simple example in equation (3.6.1). Figure 3.4 shows the lower bound and upper bound of the optimal solution as a function of the number of iterations. After 10 iterations, the gap between the upper bound and lower bound is equal to $0.2e-13$.

3.7. Decomposition of Limit Analysis Optimisation Problem

The general decomposition techniques described in the previous sections are here adapted to optimisation problem encountered in limit analysis.

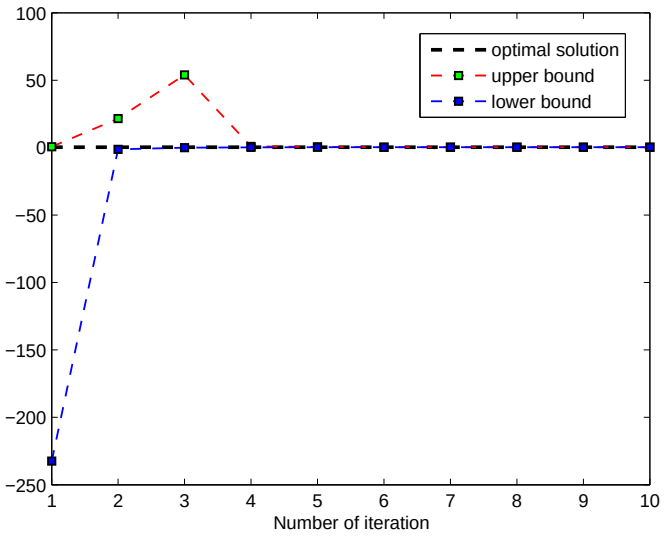


Figure 3.4.: Evolution of upper and lower bound using Benders decomposition.

3.7.1. Decomposition of LB Problem

As described in Chapter 1, the lower bound (LB) problem of limit analysis may be stated as the following optimisation problem (see (1.2.5)):

$$\begin{aligned} \min_{\boldsymbol{\sigma},\lambda} \; & -\lambda \\ & \mathbf{A}^{eq1}\boldsymbol{\sigma} + \boldsymbol{F}^{eq1} = \mathbf{0} \\ & \mathbf{A}^{eq2}\boldsymbol{\sigma} = \mathbf{0} \\ & \mathbf{A}^{eq3}\boldsymbol{\sigma} + \lambda \boldsymbol{F}^{eq3} = \mathbf{0} \\ & \boldsymbol{\sigma} \in \mathcal{B}. \end{aligned} \tag{3.7.1}$$

We translate the ideas of decomposition techniques explained in the previous sections to (LB) problem of limit analysis.

Decomposition of LB problem corresponds to the split of the stress variables $\boldsymbol{\sigma}$ into two sets $\boldsymbol{\sigma}^1$ and $\boldsymbol{\sigma}^2$. In view of equation (1.2.6), it means that the variable $\boldsymbol{x} = (\boldsymbol{x}_4, \boldsymbol{x}_{1:3})$ is split into two variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ where $\boldsymbol{x}_1 = (\boldsymbol{x}_4^1, \boldsymbol{x}_{1:3}^1)$ and $\boldsymbol{x}_2 = (\boldsymbol{x}_4^2, \boldsymbol{x}_{1:3}^2)$.

As described in Chapter 1 the equation $\mathbf{A}^{eq1}\boldsymbol{\sigma} + \lambda \boldsymbol{F}^{eq1} = \mathbf{0}$ in (1.2.5) is related to the equilibrium constraint. In view of the structure of matrix $\mathbf{A}^{eq1}$ in (A.1.4) and how it is built, we can decompose the matrix $\mathbf{A}^{eq1}$ into two

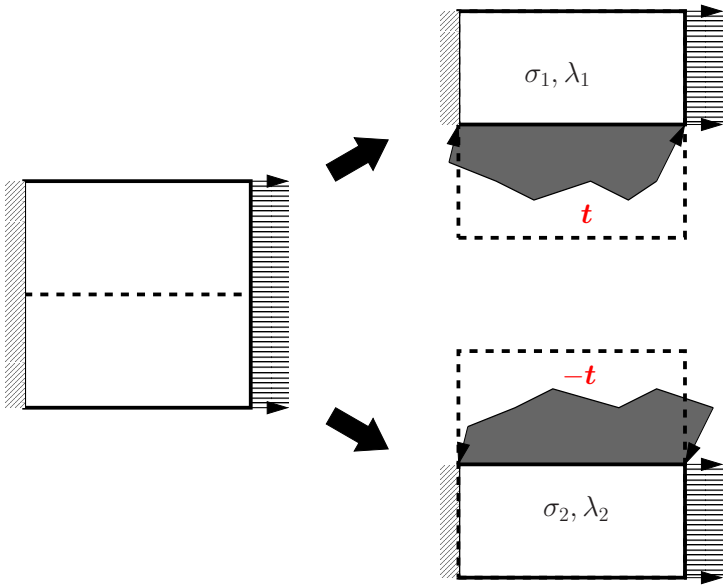


Figure 3.5.: Decomposition of global problem into two subproblems. The global variables are the boundary traction $\boldsymbol{t}$ at the fictitious Neumann condition and global load factor λ.

matrixes $\mathbf{A}^{eq1,1}$ and $\mathbf{A}^{eq1,2}$. In other words the equality $\mathbf{A}^{eq1}\boldsymbol{\sigma} + \lambda \boldsymbol{F}^{eq1} = \mathbf{0}$ can be split into two equalities as follows:

$$\begin{aligned} \mathbf{A}^{eq1,1}\boldsymbol{\sigma}^1 + \lambda \boldsymbol{F}^{eq1,1} &= \mathbf{0}, \\ \mathbf{A}^{eq1,2}\boldsymbol{\sigma}^2 + \lambda \boldsymbol{F}^{eq1,2} &= \mathbf{0}. \end{aligned} \tag{3.7.2}$$

The split of $\boldsymbol{\sigma}$ into two vectors $\boldsymbol{\sigma}^1$ and $\boldsymbol{\sigma}^2$, it leads to the decomposition of the domain into two parts, with a common boundary that couples the common nodes. It means that the vectors $\boldsymbol{\sigma}^1$ and $\boldsymbol{\sigma}^2$ are decomposed into two vectors $\boldsymbol{\sigma}^1 = (\boldsymbol{\sigma}^{1,1}, \boldsymbol{\sigma}^{1,2})$ and $\boldsymbol{\sigma}^2 = (\boldsymbol{\sigma}^{2,1}, \boldsymbol{\sigma}^{2,2})$ such that the vectors $\boldsymbol{\sigma}^{1,1}$ and $\boldsymbol{\sigma}^{2,2}$ are coupled. In other words, in view of the structure of matrix $\mathbf{A}^{eq2}$ the equation $\mathbf{A}^{eq2}\boldsymbol{\sigma} = 0$ in (1.2.5), which is related to the inter-element equilibrium constraint, can be decomposed into three equations as follows:

$$\begin{aligned} \mathbf{A}^{eq2,1}\boldsymbol{\sigma}^1 &= \mathbf{0}, \\ \mathbf{A}^{eq2,2}\boldsymbol{\sigma}^2 &= \mathbf{0}, \\ \mathbf{B}^{eq2,1}\boldsymbol{\sigma}^1 + \mathbf{B}^{eq2,2}\boldsymbol{\sigma}^2 &= \mathbf{0}. \end{aligned} \tag{3.7.3}$$

We note that the last equation in (3.7.3) is a complicating constraint. The equation $\mathbf{A}^3\boldsymbol{\sigma} + \lambda \boldsymbol{F}^{eq3} = \mathbf{0}$, in view of the structure of matrix $\mathbf{A}^{eq3}$ in

equation (A.3.1) can be decomposed as,

$$\begin{aligned} \mathbf{A}^{eq3,1}\boldsymbol{\sigma}^1 + \lambda \boldsymbol{F}^{eq3,1} &= \mathbf{0}, \\ \mathbf{A}^{eq3,2}\boldsymbol{\sigma}^2 + \lambda \boldsymbol{F}^{eq3,2} &= \mathbf{0}. \end{aligned} \tag{3.7.4}$$

Consequently, we can rewrite the optimisation problem in (1.2.5) as,

$$\begin{aligned} &\min_{\boldsymbol{x}_i,\lambda} -\lambda \\ &\quad (\mathbf{A}^{eq1,i}\mathbf{P})\boldsymbol{x}_4^i + (\mathbf{A}^{eq1,i}\mathbf{Q})\boldsymbol{x}_{1:3}^i + \lambda \boldsymbol{F}^{eq1,i} = \mathbf{0} \\ &\quad (\mathbf{A}^{eq2,i}\mathbf{P})\boldsymbol{x}_4^i + (\mathbf{A}^{eq2,i}\mathbf{Q})\boldsymbol{x}_{1:3}^i = \mathbf{0} \\ &\quad (\mathbf{A}^{eq3,i}\mathbf{P})\boldsymbol{x}_4^i + (\mathbf{A}^{eq3,i}\mathbf{Q})\boldsymbol{x}_{1:3}^i + \lambda \boldsymbol{F}^{eq3,i} = \mathbf{0} \\ &\quad \mathbf{R}_i \boldsymbol{x}_{1:3}^i = \boldsymbol{b}_i \\ &\quad (\mathbf{B}^{eq2,1}\mathbf{P})\boldsymbol{x}_4^1 + (\mathbf{B}^{eq2,1}\mathbf{Q})\boldsymbol{x}_{1:3}^1 + (\mathbf{B}^{eq2,2}\mathbf{P})\boldsymbol{x}_4^2 + (\mathbf{B}^{eq2,2}\mathbf{Q})\boldsymbol{x}_{1:3}^2 = \mathbf{0} \\ &\quad \boldsymbol{x}_{1:3}^i \in K_i, \\ &\quad \boldsymbol{x}_4^i,\ \lambda \text{ are free}, (i = 1, 2). \end{aligned} \tag{3.7.5}$$

The last equation in the previous optimisation problem (3.7.5) is a complicating constraint. The variables $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ can be regarded as local variables, while the variable λ can be regarded as a global variable. In order to decompose the problem in (3.7.5) we first introduce a variable $\boldsymbol{t}$ such that

$$(\mathbf{B}^{eq2,1}\mathbf{P})\boldsymbol{x}_4^1 + (\mathbf{B}^{eq2,1}\mathbf{Q})\boldsymbol{x}_{1:3}^1 = \boldsymbol{t},$$

Then, we can rewrite the optimisation problem in the following form:

$$\begin{aligned} &\min_{\boldsymbol{t},\boldsymbol{x}_i,\lambda} -\lambda \\ &\quad (\mathbf{A}^{eq1,i}\mathbf{P})\boldsymbol{x}_4^i + (\mathbf{A}^{eq1,i}\mathbf{Q})\boldsymbol{x}_{1:3}^i + \lambda \boldsymbol{F}^{eq1,i} = \mathbf{0} \\ &\quad (\mathbf{A}^{eq2,i}\mathbf{P})\boldsymbol{x}_4^i + (\mathbf{A}^{eq2,i}\mathbf{Q})\boldsymbol{x}_{1:3}^i = \mathbf{0} \\ &\quad (\mathbf{A}^{eq3,i}\mathbf{P})\boldsymbol{x}_4^i + (\mathbf{A}^{eq3,i}\mathbf{Q})\boldsymbol{x}_{1:3}^i + \lambda \boldsymbol{F}^{eq3,i} = \mathbf{0} \\ &\quad \mathbf{R}_i \boldsymbol{x}_{1:3}^i = \boldsymbol{b}_i \\ &\quad (\mathbf{B}^{eq2,i}\mathbf{P})\boldsymbol{x}_4^i + (\mathbf{B}^{eq2,i}\mathbf{Q})\boldsymbol{x}_{1:3}^i = (-1)^{i+1}\boldsymbol{t} \\ &\quad \boldsymbol{x}_{1:3}^i \in K_i \\ &\quad \boldsymbol{x}_4^i, \boldsymbol{t}, \lambda \text{ are free}, \ (i = 1, 2). \end{aligned} \tag{3.7.6}$$

Note that since the complicating constraint in optimisation problem (3.7.5) is built through the common boundary, the coupling constraint can be interpreted as a *fictitious Neumann* condition for each subdomain.

Let $\boldsymbol{y} = (\boldsymbol{t}, \lambda)$, $\boldsymbol{x}^i = (\boldsymbol{x}_4^i, x_{1:3}^i)$ and $C_i, (i = 1, 2)$ be local constraints that are defined as follows:

$$C_i = \left\{ \begin{bmatrix} \boldsymbol{x}^i \\ \boldsymbol{y} \end{bmatrix} : \begin{array}{l} (\mathbf{A}^{eq1,i}\mathbf{P})\boldsymbol{x}_4^i + (\mathbf{A}^{eq1,i}\mathbf{Q})\boldsymbol{x}_{1:3}^i + \lambda \boldsymbol{F}^{eq1,i} = \mathbf{0} \\ (\mathbf{A}^{eq2,i}\mathbf{P})\boldsymbol{x}_4^i + (\mathbf{A}^{eq2,i}\mathbf{Q})\boldsymbol{x}_{1:3}^i = \mathbf{0} \\ (\mathbf{A}^{eq3,i}\mathbf{P})\boldsymbol{x}_4^i + (\mathbf{A}^{eq3,i}\mathbf{Q})\boldsymbol{x}_{1:3}^i + \lambda \boldsymbol{F}^{eq3,i} = \mathbf{0} \\ \mathbf{R}_i \boldsymbol{x}_{1:3}^i = \boldsymbol{b}_i \\ (\mathbf{B}^{eq2,i}\mathbf{P})\boldsymbol{x}_4^i + (\mathbf{B}^{eq2,i}\mathbf{Q})\boldsymbol{x}_{1:3}^i = (-1)^{i+1}\boldsymbol{t} \\ \boldsymbol{x}_{1:3}^i \in K_i,\ \boldsymbol{x}_4^i;\ \boldsymbol{t}, \lambda \text{ are free.} \end{array} \right\} \tag{3.7.7}$$

Then, problem (3.7.6) reads:

$$\begin{aligned} \min_{\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{y}} & f_1(\boldsymbol{x}^1, \boldsymbol{y}) + f_2(\boldsymbol{x}^2, \boldsymbol{y}) \\ & (\boldsymbol{x}^1, \boldsymbol{y}) \in C_1, (\boldsymbol{x}^2, \boldsymbol{y}) \in C_2, \end{aligned} \tag{3.7.8}$$

where $f_i(\boldsymbol{x}^i, \boldsymbol{y}) = -\frac{\lambda}{2}$.

In the above problem, The variable $\boldsymbol{y}$ is a public or so to speak a complicating variable and the variable $\boldsymbol{x}^i$ is called a private variable or local variable.

We now apply primal and dual decomposition for the optimisation problem in (3.7.8).

3.7.2. Primal Decomposition (LB)

In primal decomposition, at each iteration we fix the public variable $\boldsymbol{y}$. Thus by fixing the variable $\boldsymbol{y}$, the optimisation problem (3.7.8) is separable.

Each subproblem can separately find optimal values for its local variables $\boldsymbol{x}^i$. Let us denote, $q_i(\boldsymbol{y})$ the optimal value of the following subproblem:

$$\begin{aligned} q_i(\boldsymbol{y}) = \min_{\boldsymbol{x}^i} & f_i(\boldsymbol{x}^i, \boldsymbol{y}) \\ & (\boldsymbol{x}^i, \boldsymbol{y}) \in C_i, \end{aligned} \tag{3.7.9}$$

with variable $\boldsymbol{x}^i$, as a function of $\boldsymbol{y}$. The original problem (3.7.8) is equivalent to the primal master problem

$$\min_{\boldsymbol{y}} q(\boldsymbol{y}) = q_1(\boldsymbol{y}) + q_2(\boldsymbol{y}), \tag{3.7.10}$$

with variable $\boldsymbol{y}$. In order to find a subgradient of q, we find $\boldsymbol{g}^i \in \partial q_i(\boldsymbol{y})$(which can be done separately), then we have

$$\boldsymbol{g} = \boldsymbol{g}^1 + \boldsymbol{g}^2.$$

3.7.3. Dual Decomposition (LB)

By introducing the new variables $\boldsymbol{t}^i$, λ^i, $\boldsymbol{y}^i = (\boldsymbol{t}^i, \lambda^i)$, $i = 1, 2$, and $\boldsymbol{z} = (\boldsymbol{t}, \lambda)$ we can rewrite the optmisation problem at (3.7.8) in the following form:

$$\begin{aligned} \min_{\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{y}^1, \boldsymbol{y}^2, \boldsymbol{z}} & f_1(\boldsymbol{x}^1, \boldsymbol{y}^1) + f_2(\boldsymbol{x}^2, \boldsymbol{y}^2) \\ & \boldsymbol{y}^1 - \boldsymbol{z} = \boldsymbol{0} \\ & \boldsymbol{y}^2 - \boldsymbol{z} = \boldsymbol{0} \\ & (\boldsymbol{x}^1, \boldsymbol{y}^1) \in C_1, \quad (\boldsymbol{x}^2, \boldsymbol{y}^2) \in C_2. \end{aligned} \tag{3.7.11}$$

We form the partial Lagrangian of problem (3.7.11),

$$\begin{aligned} L(\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{y}^1, \boldsymbol{y}^2, \boldsymbol{z}; \boldsymbol{v}^1, \boldsymbol{v}^2) = & f_1(\boldsymbol{x}^1, \boldsymbol{y}^1) + f_2(\boldsymbol{x}^2, \boldsymbol{y}^2) + \boldsymbol{v}^1 \cdot (-\boldsymbol{y}^1 + \boldsymbol{z}) + \\ & \boldsymbol{v}^2 \cdot (-\boldsymbol{y}^2 + \boldsymbol{z}) = (f_1(\boldsymbol{x}^1, \boldsymbol{y}^1) - \boldsymbol{v}^1 \cdot \boldsymbol{y}^1) + \\ & (f_2(\boldsymbol{x}^2, \boldsymbol{y}^2) - \boldsymbol{v}^2 \cdot \boldsymbol{y}^2) + (\boldsymbol{v}^1 + \boldsymbol{v}^2) \cdot \boldsymbol{z}, \end{aligned}$$

where $\boldsymbol{v}^i$ is the Lagrangian multiplier associated with $\boldsymbol{y}^i - \boldsymbol{z} = \boldsymbol{0}$. To find the dual function, we first minimize over $\boldsymbol{z}$, which results in the condition $\boldsymbol{v}^1 + \boldsymbol{v}^2 = \boldsymbol{0}$. In other words,

$$q(\boldsymbol{v}^1, \boldsymbol{v}^2) = \min_{\substack{\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{y}^1, \boldsymbol{y}^2 \\ (\boldsymbol{x}^1, \boldsymbol{y}^1) \in C_1 \\ (\boldsymbol{x}^2, \boldsymbol{y}^2) \in C_2}} \min_{\boldsymbol{z}} (f_1(\boldsymbol{x}^1, \boldsymbol{y}^1) - \boldsymbol{v}^1 \cdot \boldsymbol{y}^1) + (f_2(\boldsymbol{x}^2, \boldsymbol{y}^2) - \boldsymbol{v}^2 \cdot \boldsymbol{y}^2) + (\boldsymbol{v}^1 + \boldsymbol{v}^2) \cdot \boldsymbol{z},$$

then

$$q(\boldsymbol{v}^1, \boldsymbol{v}^2) = \min_{\substack{\boldsymbol{x}^1, \boldsymbol{y}^1, \boldsymbol{x}^2, \boldsymbol{y}^2 \\ (\boldsymbol{x}^1, \boldsymbol{y}^1) \in C_1 \\ (\boldsymbol{x}^2, \boldsymbol{y}^2) \in C_2 \\ \boldsymbol{v}^1 + \boldsymbol{v}^2 = \boldsymbol{0}}} (f_1(\boldsymbol{x}^1, \boldsymbol{y}^1) - \boldsymbol{v}^1 \cdot \boldsymbol{y}^1) + (f_2(\boldsymbol{x}^2, \boldsymbol{y}^2) - \boldsymbol{v}^2 \cdot \boldsymbol{y}^2).$$

We define $q_i(\boldsymbol{v}^i)$, $(i = 1, 2)$ as the optimal value of the subproblem

$$\begin{aligned} q_i(\boldsymbol{v}^i) = \min_{\boldsymbol{x}^i, \boldsymbol{y}^i} & (f_i(\boldsymbol{x}^i, \boldsymbol{y}^i) - \boldsymbol{v}^i \cdot \boldsymbol{y}^i) \\ & (\boldsymbol{x}^i, \boldsymbol{y}^i) \in C_i, \end{aligned} \tag{3.7.12}$$

as a function of $\boldsymbol{v}_i$. A subgradient of q_i at $\boldsymbol{v}^i$ is just $-\boldsymbol{y}^i$, an optimal value of $\boldsymbol{y}^i$ in the subproblem (3.7.12).

Therefore the dual of the original problem (3.7.11) is

$$\max q(\boldsymbol{v}^1, \boldsymbol{v}^2) = q_1(\boldsymbol{v}^1) + q_2(\boldsymbol{v}^2)$$
$$\mathbf{E}^T \begin{Bmatrix} \boldsymbol{v}^1 \\ \boldsymbol{v}^2 \end{Bmatrix} = \mathbf{0},$$

with variable $\boldsymbol{v}^1$, $\boldsymbol{v}^2$ and $\mathbf{E}^T = [\mathbf{I} \quad \mathbf{I}]$. We can solve this dual decomposition master problem using a projected subgradient method. The projection onto the feasible set $\left\{(\boldsymbol{v}^1, \boldsymbol{v}^2) | \boldsymbol{v}^1 + \boldsymbol{v}^2 = \mathbf{0}\right\}$, is using expression in (3.2.2) given by:

$$\mathbf{P} = \mathbf{I} - \mathbf{E}(\mathbf{E}^T\mathbf{E})^{-1}\mathbf{E}^T.$$

It can be verified that

$$\mathbf{P} \begin{Bmatrix} \boldsymbol{v}^1 \\ \boldsymbol{v}^2 \end{Bmatrix} = \frac{1}{2} \begin{Bmatrix} \boldsymbol{v}^1 - \boldsymbol{v}^2 \\ \boldsymbol{v}^2 - \boldsymbol{v}^1 \end{Bmatrix}.$$

3.7.4. Benders Decomposition(LB)

As it can be verified, the lower bound (LB) problem of limit analysis could be stated as the following structure:

$$\begin{aligned} &\min -\lambda \\ &\quad \mathbf{A}_1\boldsymbol{x}_1 + \lambda \boldsymbol{F}_1 = \boldsymbol{b}_1 \\ &\quad \mathbf{A}_2\boldsymbol{x}_2 + \lambda \boldsymbol{F}_2 = \boldsymbol{b}_2 \\ &\quad \mathbf{B}_1\boldsymbol{x}_1 + \mathbf{B}_2\boldsymbol{x}_2 = \mathbf{0} \\ &\quad \boldsymbol{x}_1 \in K_1, \boldsymbol{x}_2 \in K_2, \lambda \text{ is free}, \end{aligned} \tag{3.7.13}$$

where K_1 and K_2 are closed convex cones.

As before, we shall use a free complicating variable $\boldsymbol{t}$ to rewrite the previous equations as the following two sets of coupled equations,

$$\begin{aligned} \min_{\boldsymbol{x}_1, \boldsymbol{x}_2, \lambda} \; & -\lambda \\ & \begin{cases} \mathbf{A}_1\boldsymbol{x}_1 + \lambda \boldsymbol{F}_1 = \boldsymbol{b}_1 \\ \mathbf{B}_1\boldsymbol{x}_1 + \boldsymbol{t} = \mathbf{0} \\ \boldsymbol{x}_1 \in K_1 \end{cases} \\ & \begin{cases} \mathbf{A}_2\boldsymbol{x}_2 + \lambda \boldsymbol{F}_2 = \boldsymbol{b}_2 \\ \mathbf{B}_2\boldsymbol{x}_2 - \boldsymbol{t} = \mathbf{0} \\ \boldsymbol{x}_2 \in K_2. \end{cases} \end{aligned} \tag{3.7.14}$$

The LB optimisation problem can be then restated as the two subproblems,

$$
\begin{array}{ll}
(P_i)\ \min\limits_{\boldsymbol{x}_i} 0 & (D_i)\ \max(\boldsymbol{b}_i - \lambda^k \boldsymbol{F}_i)\cdot \boldsymbol{y}_{i\lambda} + (-1)^i \boldsymbol{t}^k \cdot \boldsymbol{y}_{i\boldsymbol{t}} \\
\quad \boldsymbol{y}_{i\lambda} : \mathbf{A}_i \boldsymbol{x}_i = \boldsymbol{b}_i - \lambda^k \boldsymbol{F}_i & \quad -(\mathbf{A}_i^T \boldsymbol{y}_{i\lambda} + \mathbf{B}_i^T \boldsymbol{y}_{i\boldsymbol{t}}) \in K_i^* \\
\quad \boldsymbol{y}_{i\boldsymbol{t}} : \mathbf{B}_i \boldsymbol{x}_i = (-1)^i \boldsymbol{t}^k & \\
\quad \boldsymbol{x}_i \in K_i. & \\
\quad (i = 1, 2) &
\end{array}
\tag{3.7.15}
$$

where $\boldsymbol{t}^k$ and λ^k are the solution of the following master problem:

$$
\begin{array}{l}
\min\limits_{\boldsymbol{t},\lambda,\alpha_1,\alpha_2} \ -\lambda + \alpha_1 + \alpha_2 \\
\qquad \alpha_1 \geq (\boldsymbol{b}_1 - \lambda \boldsymbol{F}_1)\cdot \boldsymbol{y}_{1\lambda}^k - \boldsymbol{t}\cdot \boldsymbol{y}_{1\boldsymbol{t}}^k \\
\qquad \alpha_2 \geq (\boldsymbol{b}_2 - \lambda \boldsymbol{F}_2)\cdot \boldsymbol{y}_{2\lambda}^k + \boldsymbol{t}\cdot \boldsymbol{y}_{2\boldsymbol{t}}^k \quad , \ k = 1, 2, \ldots
\end{array}
\tag{3.7.16}
$$

The inequality constraints in the previous optimisation problem are applied for all the available optimal dual variables $\boldsymbol{y}_{i\boldsymbol{t}}$ and $\boldsymbol{y}_{i\lambda}$ of the subproblems obtained by using different values of $\boldsymbol{t}^k$ and λ^k, $k = 1, 2, \ldots$ in subproblem (3.7.15). The initial values $\boldsymbol{t}^0 = \mathbf{0}$ and $\lambda^0 = 0$ may be employed.

The inequality constraint $\alpha_i \geq (\boldsymbol{b}_i - \lambda \boldsymbol{F}_i)\cdot \boldsymbol{y}_{i\lambda}^k + (-1)^i \boldsymbol{t}\cdot \boldsymbol{y}_{i\boldsymbol{t}}^k$ in (3.7.16) which is obtained by optimal dual variables $\boldsymbol{y}_{i\boldsymbol{t}}$ and $\boldsymbol{y}_{i\lambda}$ of subproblem (3.7.15) is so called *optimality* cut. The optimality cut is obtained when the primal subproblem (P_i) (3.7.15) is feasible and finite. If primal subproblem (P_i) for $(\lambda^k, \boldsymbol{t}^k)$ is not feasible then in term of the Frakas Lemma B.5, the dual subproblem (D_i) (3.7.15) is infinite, it means that, there exists $\bar{\boldsymbol{y}}_{i\boldsymbol{t}}^k$ and $\bar{\boldsymbol{y}}_{i\lambda}^k$ such that

$$
-(\mathbf{A}_i^T \bar{\boldsymbol{y}}_{i\lambda}^k + \mathbf{B}_i^T \bar{\boldsymbol{y}}_{i\boldsymbol{t}}^k) \in K_i^* \quad \text{and} \quad (\boldsymbol{b}_i - \lambda^k \boldsymbol{F}_i)\cdot \bar{\boldsymbol{y}}_{i\lambda}^k + (-1)^i \boldsymbol{t}^k \cdot \bar{\boldsymbol{y}}_{i\boldsymbol{t}}^k > 0.
\tag{3.7.17}
$$

Since K_i^* is a cone then we have

$$
\begin{array}{l}
\beta(\boldsymbol{b}_i - \lambda^k \boldsymbol{F}_i)\cdot \bar{\boldsymbol{y}}_{i\lambda}^k + \beta(-1)^i \boldsymbol{t}^k \cdot \bar{\boldsymbol{y}}_{i\boldsymbol{t}}^k > 0, \quad \forall \beta > 0, \\
-\beta(\mathbf{A}_i^T \bar{\boldsymbol{y}}_{i\lambda}^k + \mathbf{B}_i^T \bar{\boldsymbol{y}}_{i\boldsymbol{t}}^k) \in K_i^*.
\end{array}
$$

Thus, when β tends to $+\infty$, the objective function of dual subproblem (D_i) (3.7.15) tends to $+\infty$. In other words, dual subproblem (D_i) (3.7.15) is finite if and only if

$$(\boldsymbol{b}_i - \lambda^k \boldsymbol{F}_i) \cdot \boldsymbol{y}_{i\lambda} + (-1)^i \boldsymbol{t}^k \cdot \boldsymbol{y}_{i\boldsymbol{t}} \leq 0, \quad \text{when} \quad -(\mathbf{A}_i^T \boldsymbol{y}_{i\lambda} + \mathbf{B}_i^T \boldsymbol{y}_{i\boldsymbol{t}}) \in K_i^*.$$

Consequently we append these constraints which are so called *feasibility* cuts to the master problem to ensure that the dual is always bounded.

By modifying the primal subproblem (P_i) in (3.7.15) as follows, we obtain either feasibility cut or optimality cut :

$$\begin{aligned} (P_i) : \min\, & s_i^+ + s_i^- + \boldsymbol{e} \cdot \boldsymbol{w}_i^+ + \boldsymbol{e} \cdot \boldsymbol{w}_i^- \\ & \boldsymbol{y}_{i\lambda} : \mathbf{A}_i \boldsymbol{x}_i + s_i^+ \boldsymbol{F}_i - s_i^- \boldsymbol{F}_i = \boldsymbol{b}_i - \lambda^k \boldsymbol{F}_i \\ & \boldsymbol{y}_{i\boldsymbol{t}} : \mathbf{B}_i^T \boldsymbol{x}_i + \mathbf{I}\boldsymbol{w}_i^+ - \mathbf{I}\boldsymbol{w}_i^- = (-1)^i \boldsymbol{t}^k \\ & \boldsymbol{x}_i \in K_i, s_i^+ \geq 0, s_i^- \geq 0, \boldsymbol{w}_i^+ \geq \mathbf{0}, \boldsymbol{w}_i^- \geq \mathbf{0} \end{aligned} \tag{3.7.18}$$

$$\begin{aligned} (D_i) : \max\, & (\boldsymbol{b}_i - \lambda^k \boldsymbol{F}_i) \cdot \boldsymbol{y}_{i\lambda} + (-1)^i \boldsymbol{t}^k \cdot \boldsymbol{y}_{i\boldsymbol{t}} \\ & - (\mathbf{A}_i^T \boldsymbol{y}_{i\lambda} + \mathbf{B}_i^T \boldsymbol{y}_{i\boldsymbol{t}}) \in K_i^* \\ & - 1 \leq \boldsymbol{F}_i \boldsymbol{y}_{i\lambda} \leq 1 \\ & - \boldsymbol{e} \leq \boldsymbol{y}_{i\boldsymbol{t}} \leq \boldsymbol{e} \end{aligned} \tag{3.7.19}$$

where $\boldsymbol{e}$ is a vector that all components are equal 1.

If the optimal solution of primal problem (P_i) (3.7.18) is zero, then we have optimality cut, and if it is strictly positive, we have feasibility cuts. The full master problem with feasibility and optimality cuts is obtained as follows :

$$\begin{aligned} \min_{\boldsymbol{t},\lambda,\alpha_1,\alpha_2} \quad & -\lambda + \alpha_1 + \alpha_2 \\ & \alpha_1 \geq (\boldsymbol{b}_1 - \lambda \boldsymbol{F}_1) \cdot \boldsymbol{y}_{1\lambda}^k - \boldsymbol{t} \cdot \boldsymbol{y}_{1\boldsymbol{t}}^k \\ & \alpha_2 \geq (\boldsymbol{b}_2 - \lambda \boldsymbol{F}_2) \cdot \boldsymbol{y}_{2\lambda}^k + \boldsymbol{t} \cdot \boldsymbol{y}_{2\boldsymbol{t}}^k \;,\; k = 1, 2, \ldots \\ & 0 \geq (\boldsymbol{b}_1 - \lambda \boldsymbol{F}_1) \cdot \bar{\boldsymbol{y}}_{1\lambda}^l - \boldsymbol{t} \cdot \bar{\boldsymbol{y}}_{1\boldsymbol{t}}^l \\ & 0 \geq (\boldsymbol{b}_2 - \lambda \boldsymbol{F}_2) \cdot \bar{\boldsymbol{y}}_{2\lambda}^l + \boldsymbol{t} \cdot \bar{\boldsymbol{y}}_{2\boldsymbol{t}}^l \;,\; l = 1, 2, \ldots \end{aligned} \tag{3.7.20}$$

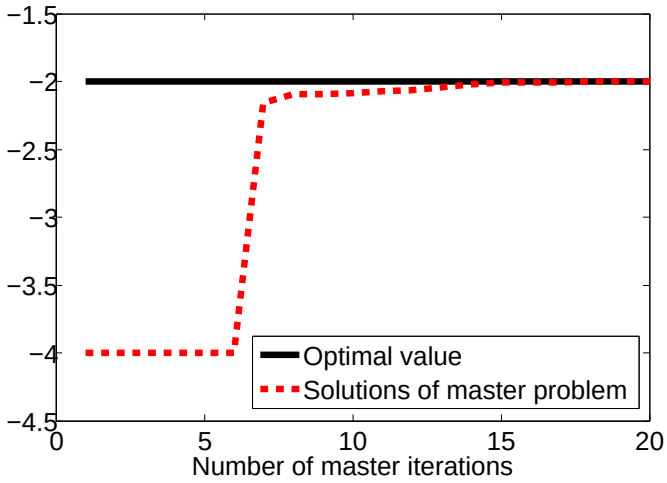
Now we apply the Benders decomposition method for the following different sizes of LB problem with the same structure defined in (3.7.13). The size of the problems are given in Table 3.1, where n and m denote the number of rows and and columns of matrix $\mathbf{A}$ in (1.2.5), and n_T is the total number

Problem	n	m	n_T
a	96	97	20
b	368	385	101
c	2240	2401	700

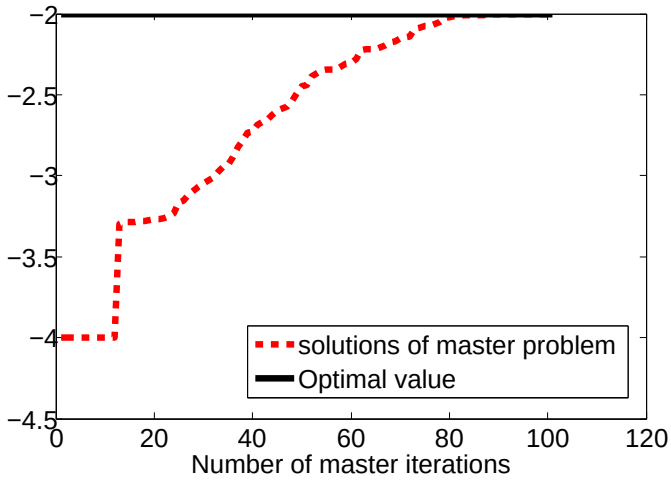
Table 3.1.: Size and total master iterations of each problem solved using SDPT3 [47].

of master iterations. The problems have been solved using the optimisation software SDPT3 [47]. Figure 3.6 shows exact global optimal solution and optimal solution of master problem at each iteration respectively.

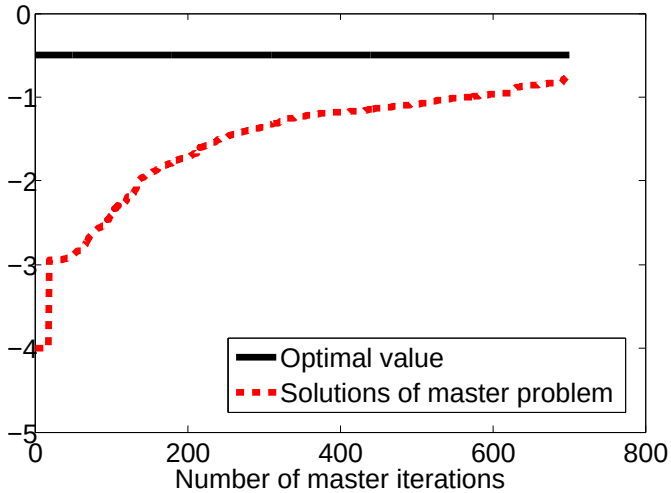
As it can be observed, the number of iteration increases dramatically due to nonlinearity of the conic constraints, which can not be properly represented with the linear feasibility and optimality cuts. This fact motivates, the use of the proposed method in the next Chapter.



(a)



(b)



(c)

Figure 3.6.: Benders decomposition, apply to the LB limit analysis problem.

4

AAR-Based Decomposition Algorithm for Non-linear Convex Optimisation

4.1. Alternative Definition of Global Feasibility Region

The objective of this section is to provide a description of a decomposition method that exploits the results of the AAR method given in Chapter 2. The method is suitable for convex optimisation problems that have the structure given in (1.2.7), which we aim to transform into the computation of the minimal distance between two feasible sets. For this aim, we rewrite problem (1.2.7) by introducing a new complicating variable $\boldsymbol{t}$ as follows:

$$\begin{aligned} \lambda^* = \max_{\boldsymbol{x}_1,\boldsymbol{x}_2,\lambda,\boldsymbol{t}} \ & \lambda \\ & \boldsymbol{f}_1(\boldsymbol{x}_1,\lambda) = \boldsymbol{0} \\ & \boldsymbol{f}_2(\boldsymbol{x}_2,\lambda) = \boldsymbol{0} \\ & \boldsymbol{g}_1(\boldsymbol{x}_1) = \boldsymbol{t} \\ & \boldsymbol{g}_2(\boldsymbol{x}_2) = -\boldsymbol{t} \\ & \boldsymbol{x}_1 \in K_1,\ \boldsymbol{x}_2 \in K_2,\ \lambda \in \Re. \end{aligned} \tag{4.1.1}$$

We next define the feasibility region of this problem with the help of the following definitions:

Definition 4.1 *Consider the following two feasibility regions:*

$$\begin{aligned} X_1(\lambda) &= \{\boldsymbol{x}_1 | \boldsymbol{f}_1(\boldsymbol{x}_1,\lambda) = \boldsymbol{0}\} \cap K_1, \\ X_2(\lambda) &= \{\boldsymbol{x}_2 | \boldsymbol{f}_2(\boldsymbol{x}_2,\lambda) = \boldsymbol{0}\} \cap K_2, \end{aligned} \tag{4.1.2}$$

and also let $\lambda = \bar{\lambda}$ *be a given real value. Then, we define the following feasibility sets for variable* $\boldsymbol{t}$*:*

$$\begin{aligned} Z(\bar{\lambda}) &= \boldsymbol{g}_1(X_1(\bar{\lambda})) = \{\boldsymbol{g}_1(\boldsymbol{x}_1)|\boldsymbol{x}_1 \in X_1(\bar{\lambda})\}, \\ W(\bar{\lambda}) &= -\boldsymbol{g}_2(X_1(\bar{\lambda})) = \{-\boldsymbol{g}_2(\boldsymbol{x}_2)|\boldsymbol{x}_2 \in X_2(\bar{\lambda})\}. \end{aligned} \tag{4.1.3}$$

Throughout the subsequent sections, the sets $Z(\lambda)$ and $W(\lambda)$ defined in (4.1.3) are assumed closed and convex. In addition, the functions $\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{g}_1$ and $\boldsymbol{g}_2$ are affine maps.

By using definitions (4.1.2) and (4.1.3), the optimisation problem in (4.1.1) may be recast as,

$$\begin{aligned} \lambda^* = \max_{\lambda} & \lambda \\ & Z(\lambda) \cap W(\lambda) \neq \emptyset. \end{aligned} \tag{4.1.4}$$

The algorithm proposed in this Chapter is based on the form (4.1.4). In brief, the algorithm consists on updating the value of $\bar{\lambda}$ (master problem) and analysing in the subproblems whether the intersection between the sets $Z(\bar{\lambda})$ and $W(\bar{\lambda})$ is empty or not, or equivalently, whether $d(W(\bar{\lambda}), Z(\bar{\lambda})) > 0$.

According to the definitions in Section 2.2 we have that:

$$d(W(\lambda), Z(\lambda)) = \inf_{\substack{\boldsymbol{x}_1 \in X_1(\lambda) \\ \boldsymbol{x}_2 \in X_2(\lambda)}} ||\boldsymbol{g}_1(\boldsymbol{x}_1) + \boldsymbol{g}_2(\boldsymbol{x}_2)|| \tag{4.1.5}$$

where $X_i(\lambda)$, $i = 1, 2$ is defined in (4.1.2).

In order to determine this and compute upper bounds of the global problem in (4.1.1), we will need the following two propositions:

Proposition 4.1 *Let* λ_0 *and* $\bar{\lambda}$ *be given real values such that* $(\boldsymbol{x}_{01}, \boldsymbol{x}_{02}, \boldsymbol{t}_0, \lambda_0)$ *is a feasible solution for problem* (4.1.1)*, and* $\lambda_0 < \bar{\lambda}$*. After using the definition in* (4.1.3)*, the following relation holds:*

$$Z(\bar{\lambda}) \cap W(\bar{\lambda}) \neq \emptyset \iff \bar{\lambda} \leq \lambda^*.$$

Proof 7 *First suppose that* $Z(\bar{\lambda}) \cap W(\bar{\lambda}) \neq \emptyset$ *and* $\bar{\boldsymbol{t}}$ *belongs to* $Z(\bar{\lambda}) \cap W(\bar{\lambda})$*, thus there exist* $\boldsymbol{x}_1 \in X_1(\bar{\lambda})$ *and* $\boldsymbol{x}_2 \in X_2(\bar{\lambda})$ *such that* $\boldsymbol{g}_1(\boldsymbol{x}_1) = \bar{\boldsymbol{t}}$ *and* $-\boldsymbol{g}_2(\boldsymbol{x}_2) = \bar{\boldsymbol{t}}$*. Therefore, in view of* (4.1.2)*,* $(\boldsymbol{x}_1, \boldsymbol{x}_2, \bar{\lambda}, \bar{\boldsymbol{t}})$ *is a feasible solution for problem* (4.1.1)*, and consequently* $\bar{\lambda} \leq \lambda^*$*.*

Conversely, let $\lambda_0 < \bar{\lambda} < \lambda^*$*. Hence, there exists* $\gamma \in (0, 1)$ *such that* $\bar{\lambda} = (1 - \gamma)\lambda_0 + \gamma\lambda^*$*. Since* $(\boldsymbol{x}_{01}, \boldsymbol{x}_{02}, \boldsymbol{t}_0, \lambda_0)$ *and* $(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, \boldsymbol{t}^*, \lambda^*)$ *are feasible*

solutions for problem (4.1.1) *and since the feasible region of problem* (4.1.1) *is convex, it follows that the convex combination of these two points is a feasible solution for problem* (4.1.1). *Formally, we have that*

$$\begin{aligned}&(1-\gamma)(\boldsymbol{x}_{01},\boldsymbol{x}_{02},\boldsymbol{t}_0,\lambda_0)+\gamma(\boldsymbol{x}_1^*,\boldsymbol{x}_2^*,\boldsymbol{t}^*,\lambda^*)\\&=((1-\gamma)\boldsymbol{x}_{01}+\gamma\boldsymbol{x}_1^*,(1-\gamma)\boldsymbol{x}_{02}+\gamma\boldsymbol{x}_2^*,(1-\gamma)\boldsymbol{t}_0+\gamma\boldsymbol{t}^*,(1-\gamma)\lambda_0+\gamma\lambda^*))\\&=((1-\gamma)\boldsymbol{x}_{01}+\gamma\boldsymbol{x}_1^*,(1-\gamma)\boldsymbol{x}_{02}+\gamma\boldsymbol{x}_2^*,(1-\gamma)\boldsymbol{t}_0+\gamma\boldsymbol{t}^*,\bar{\lambda}),\end{aligned}$$

which shows that $(1-\gamma)\boldsymbol{t}_0+\gamma\boldsymbol{t}^*$ *belongs to* $Z(\bar{\lambda})\cap W(\bar{\lambda})$, *i.e.*

$$Z(\bar{\lambda})\cap W(\bar{\lambda})\neq\emptyset.$$

□

Proposition 4.2 *Let* $(\bar{\boldsymbol{t}},\bar{\lambda})$ *be an arbitrary given vector and* $\Delta\bar{s}_i$, $i=1,2$, *be optimal solutions of the following optimisation problems:*

$$\begin{aligned}\Delta\bar{s}_i=\max_{\boldsymbol{x}_i,\Delta\boldsymbol{w},\Delta s_i}\ &\Delta s_i\\&\boldsymbol{f}_i(\boldsymbol{x}_i,\bar{\lambda}+\Delta s_i)=\boldsymbol{0}\\&\boldsymbol{g}_i(\boldsymbol{x}_i)=(-1)^{i+1}(\bar{\boldsymbol{t}}+\Delta\boldsymbol{w}_i)\\&\boldsymbol{x}_i\in K_i,\ \Delta\boldsymbol{w}_i\in\Re^{n_m},\ \Delta s_i\in\Re,\end{aligned}\tag{4.1.6}$$

with i=1,2. Then $\lambda^*\leq\bar{\lambda}+\Delta\bar{s}_i$.

Proof 8 *There exists a real value* Δs^* *and a vector* $\Delta\boldsymbol{w}^*$ *such that*

$$\begin{aligned}\lambda^*&=\bar{\lambda}+\Delta s^*;\\\boldsymbol{t}^*&=\bar{\boldsymbol{t}}+\Delta\boldsymbol{w}^*.\end{aligned}\tag{4.1.7}$$

Since $(\boldsymbol{x}_1^*,\boldsymbol{x}_2^*,\boldsymbol{t}^*,\lambda^*)$ *is a feasible solution for problem* (4.1.1), *and in view of* (4.1.7) *we have that*

$$\begin{cases}\boldsymbol{f}_i(\boldsymbol{x}_i^*,\lambda^*)=\boldsymbol{0}\\\boldsymbol{g}_i(\boldsymbol{x}_i^*)=(-1)^{i+1}\boldsymbol{t}^*\\\boldsymbol{x}_i^*\in K_i\end{cases}\Rightarrow\qquad\begin{cases}\boldsymbol{f}_i(\boldsymbol{x}_i^*,\bar{\lambda}+\Delta s^*)=\boldsymbol{0}\\\boldsymbol{g}_i(\boldsymbol{x}_i^*)=(-1)^{i+1}(\bar{\boldsymbol{t}}+\Delta\boldsymbol{w}^*)\\\boldsymbol{x}_i^*\in K_i,\ \Delta\boldsymbol{w}^*\in\Re^{n_m},\ \Delta\boldsymbol{s}^*\in\Re.\end{cases}$$

But since $\Delta\bar{s}_i$ *is an optimal solution of problem* (4.1.6), *it follows from* (4.1.7) *that*

$$\Delta s^*\leq\Delta\bar{s}_i\Rightarrow\bar{\lambda}+\Delta s^*\leq\bar{\lambda}+\Delta\bar{s}_i\Rightarrow\lambda^*\leq\bar{\lambda}+\Delta\bar{s}_i.$$

□

It will become convenient to consider the dual form of the global problem (1.2.7),

$$q^* = \inf_{\boldsymbol{y}_1,\boldsymbol{y}_2,\boldsymbol{y}_3} q(\boldsymbol{y}_1,\boldsymbol{y}_2,\boldsymbol{y}_3) \tag{4.1.8}$$

where

$$\begin{aligned} q(\boldsymbol{y}_1,\boldsymbol{y}_2,\boldsymbol{y}_3) = \sup_{\boldsymbol{x}_1,\boldsymbol{x}_2,\lambda} & \boldsymbol{y}_1 \cdot \boldsymbol{f}_1(\boldsymbol{x}_1,\lambda) + \boldsymbol{y}_2 \cdot \boldsymbol{f}_2(\boldsymbol{x}_2,\lambda) \\ & + \boldsymbol{y}_3 \cdot (\boldsymbol{g}_1(\boldsymbol{x}_1) + \boldsymbol{g}_2(\boldsymbol{x}_2)) + \lambda \\ & \boldsymbol{x}_1 \in K_1,\ \boldsymbol{x}_2 \in K_2,\ \lambda \in \Re. \end{aligned} \tag{4.1.9}$$

4.2. Definition of Subproblems

Let λ_0 and $\bar{\lambda}$ be given real values such that $(\boldsymbol{x}_{01}, \boldsymbol{x}_{02}, \lambda_0)$ is a feasible solution for (1.2.7) and $\lambda_0 < \bar{\lambda}$, which means that $W(\bar{\lambda})$ and $Z(\bar{\lambda})$ are nonempty (closed convex) sets. Take $\boldsymbol{t}_0$ and set

$$\boldsymbol{t}_n = T^n(\boldsymbol{t}_0) = T(\boldsymbol{t}_{n-1}),\ n = 1, 2, 3, \cdots,$$

where

$$T = \frac{R_W R_Z + I}{2} = P_W R_Z - P_Z + I, \tag{4.2.1}$$

is the transformation of the AAR method described in Section 2.2.

We next define the optimisation subproblems that will allow us to retrieve the projections P_Z and P_W, required for computing the transformation T. In view of (2.2.1), $P_Z(\boldsymbol{t}_n)$ can be obtained by solving the following optimisation problem:

$$\begin{aligned} \min_{\boldsymbol{z}} & \|\boldsymbol{z} - \boldsymbol{t}_n\| \\ & \boldsymbol{z} \in Z(\bar{\lambda}), \end{aligned}$$

which is equivalent to the following so-called Subproblem 1 :

$$\begin{aligned} \min_{\boldsymbol{x}_1,\boldsymbol{d}^1} & \|\boldsymbol{d}^1\| \\ & \boldsymbol{f}_1(\boldsymbol{x}_1,\bar{\lambda}) = \boldsymbol{0} \\ & \boldsymbol{g}_1(\boldsymbol{x}_1) - \boldsymbol{d}^1 = \boldsymbol{t}_n \\ & \boldsymbol{x}_1 \in K_1. \end{aligned} \tag{4.2.2}$$

From the optimal solution of Subproblem 1, $\boldsymbol{d}_n^1$, we can compute the projection $P_Z(\boldsymbol{t}_n)$ and reflection $R_Z(\boldsymbol{t}_n)$ as,

$$\begin{aligned} &P_Z(\boldsymbol{t}_n) = \boldsymbol{g}_1(\boldsymbol{x}_{1n}) = \boldsymbol{t}_n + \boldsymbol{d}_n^1, \text{ with } \boldsymbol{x}_{1n} \in X_1(\bar{\lambda}), \\ &R_Z(\boldsymbol{t}_n) = 2P_Z(\boldsymbol{t}_n) - \boldsymbol{t}_n = \boldsymbol{t}_n + 2\boldsymbol{d}_n^1. \end{aligned} \tag{4.2.3}$$

From $P_Z(\boldsymbol{t}_n)$, the point $P_W R_Z(\boldsymbol{t}_n) = P_W(\boldsymbol{t}_n + 2\boldsymbol{d}_n^1)$ is obtained by solving the following optimisation problem:

$$\begin{aligned} \min_{\boldsymbol{w}} &\|\boldsymbol{w} - R_Z(\boldsymbol{t}_n)\| \\ &\boldsymbol{w} \in W(\bar{\lambda}), \end{aligned}$$

which is equivalent to the following so-called Subproblem 2:

$$\begin{aligned} \min_{\boldsymbol{x}_2, \boldsymbol{d}^2} &\|\boldsymbol{d}^2\| \\ &\boldsymbol{f}_2(\boldsymbol{x}_2, \bar{\lambda}) = \boldsymbol{0} \\ &\boldsymbol{g}_2(\boldsymbol{x}_2) + \boldsymbol{d}^2 = -R_Z(\boldsymbol{t}_n) \\ &\boldsymbol{x}_2 \in K_2, \end{aligned} \tag{4.2.4}$$

with $R_Z(\boldsymbol{t}_n) = \boldsymbol{t}_n + \boldsymbol{d}_n^1$, (see 4.2.3).

After solving this problem we have that,

$$\begin{aligned} &P_W R_Z(\boldsymbol{t}_n) = -\boldsymbol{g}_2(\boldsymbol{x}_{2n}) = R_Z(\boldsymbol{t}_n) + \boldsymbol{d}_n^2 = \boldsymbol{t}_n + 2\boldsymbol{d}_n^1 + \boldsymbol{d}_n^2, \text{ with } \boldsymbol{x}_{2n} \in X_2(\bar{\lambda}) \\ &R_W R_Z(\boldsymbol{t}_n) = 2P_W(R_Z(\boldsymbol{t}_n)) - R_Z(\boldsymbol{t}_n) = \boldsymbol{t}_n + 2\boldsymbol{d}_n^1 + 2\boldsymbol{d}_n^2, \end{aligned} \tag{4.2.5}$$

and according to (4.2.1),(4.2.3) and (4.2.5),

$$\boldsymbol{t}_{n+1} = T(\boldsymbol{t}_n) = \boldsymbol{t}_n + \boldsymbol{d}_n^1 + \boldsymbol{d}_n^2, \tag{4.2.6}$$

with $\boldsymbol{d}_n^1$ and $\boldsymbol{d}_n^2$ optimal solutions of (4.2.2) and (4.2.4), respectively. This iterative process and the associated projections and reflections are illustrated in Figure 4.1. Since T is nonexpansive, and in view of (4.2.1), (4.2.3),(4.2.5) and (4.2.6), we have the following results,

$$\begin{aligned} (i) \quad &\boldsymbol{t}_{n+1} - \boldsymbol{t}_n = \boldsymbol{d}_n^1 + \boldsymbol{d}_n^2 = T(\boldsymbol{t}_n) - \boldsymbol{t}_n = \\ &P_W(R_Z(\boldsymbol{t}_n)) - P_Z(\boldsymbol{t}_n) = -\boldsymbol{g}_2(\boldsymbol{x}_{2n}) - \boldsymbol{g}_1(\boldsymbol{x}_{1n}). \end{aligned} \tag{4.2.7}$$

$$\begin{aligned} (ii) \quad &\|\boldsymbol{d}_n^1 + \boldsymbol{d}_n^2\| = \|\boldsymbol{t}_{n+1} - \boldsymbol{t}_n\| = \|T(\boldsymbol{t}_n) - T(\boldsymbol{t}_{n-1})\| \leq \\ &\|\boldsymbol{t}_n - \boldsymbol{t}_{n-1}\| = \|\boldsymbol{d}_{n-1}^1 + \boldsymbol{d}_{n-1}^2\|. \end{aligned} \tag{4.2.8}$$

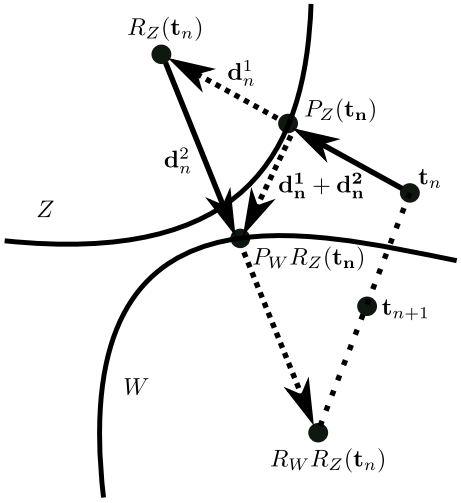


Figure 4.1.: Illustration of the iterative process

According to (4.2.8), the sequence $(\|\boldsymbol{d}_n^1 + \boldsymbol{d}_n^2\|)_{n\in N}$ is decreasing. Therefore, we can define the following parameter:

$$\alpha = \inf_{n\geq 1} \|\boldsymbol{d}_n^1 + \boldsymbol{d}_n^2\| = \lim_{n\to\infty} \|\boldsymbol{d}_n^1 + \boldsymbol{d}_n^2\| \in [0, \infty), \tag{4.2.9}$$

which measures the distance $d(W(\bar{\lambda}), Z(\bar{\lambda}))$. The next theorem relates α to the optimal objective λ^*:

Theorem 4.1 *Consider Subproblem 1 and Subproblem 2 defined in* (4.2.2) *and* (4.2.4) *respectively. Let* $\lambda_0 < \bar{\lambda}$*, with* $(\boldsymbol{x}_{01}, \boldsymbol{x}_{0,2}, \lambda_0)$ *a feasible solution of the global problem in* (1.2.7) *and* λ^* *its optimal solution. Then, with* α *defined in* (4.2.9)*, and if* $\lambda^* = q^*$*, i.e. there is no duality gap in* (4.1.8)*, the following implications hold:*

(i) $\alpha = 0$ *if and only if* $\bar{\lambda} \leq \lambda^*$.

(ii) $\alpha > 0$ *if and only if* $\bar{\lambda} > \lambda^*$.

Proof 9 *(i): If* $\alpha = 0$*, we have from* (4.2.7) *that*

$$\lim_{n\to\infty} \|\boldsymbol{d}_n^1 + \boldsymbol{d}_n^2\| = \lim_{n\to\infty} \|\boldsymbol{g}_1(\boldsymbol{x}_{1n}) + \boldsymbol{g}_2(\boldsymbol{x}_{2n})\| = 0. \tag{4.2.10}$$

Since $(\boldsymbol{x}_{1n}, \boldsymbol{x}_{2n}) \in X_1(\bar{\lambda}) \times X_2(\bar{\lambda})$, *the vector* $(\boldsymbol{x}_{1n}, \boldsymbol{x}_{2n}, \bar{\lambda})$ *satisfies the following conditions:*

$$\begin{aligned} &\boldsymbol{f}_1(\boldsymbol{x}_{1n}, \bar{\lambda}) = \boldsymbol{0} \\ &\boldsymbol{f}_2(\boldsymbol{x}_{2n}, \bar{\lambda}) = \boldsymbol{0} \\ &\boldsymbol{x}_{1n} \in K_1,\ \boldsymbol{x}_{2n} \in K_2. \end{aligned} \tag{4.2.11}$$

Suppose that $(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3)$ *is an arbitrary feasible solution for the dual problem in* (4.1.8). *Since* $(\boldsymbol{x}_{1n}, \boldsymbol{x}_{2n}, \bar{\lambda}) \in X_1(\bar{\lambda}) \times X_2(\bar{\lambda}) \times \Re$, *according to* (4.1.9) *we have that*

$$\begin{aligned} q(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3) \geq & \boldsymbol{y}_1 \cdot \boldsymbol{f}_1(\boldsymbol{x}_{1n}, \bar{\lambda}) + \boldsymbol{y}_2 \cdot \boldsymbol{f}_2(\boldsymbol{x}_{2n}, \bar{\lambda}) + \\ & \boldsymbol{y}_3 \cdot (\boldsymbol{g}_1(\boldsymbol{x}_{1n}) + \boldsymbol{g}_2(\boldsymbol{x}_{2n})) + \bar{\lambda} = \\ & \boldsymbol{y}_3 \cdot (\boldsymbol{g}_1(\boldsymbol{x}_{1n}) + \boldsymbol{g}_2(\boldsymbol{x}_{2n})) + \bar{\lambda}, \end{aligned}$$

and then

$$q(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3) \geq \lim_{n\to\infty} \boldsymbol{y}_3 \cdot (\boldsymbol{g}_1(\boldsymbol{x}_{1n}) + \boldsymbol{g}_2(\boldsymbol{x}_{2n})) + \bar{\lambda}.$$

Therefore, in view of (4.2.10), *we have that* $q(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3) \geq \bar{\lambda}$. *On the other hand, since* $(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3)$ *is an arbitrary feasible solution for the dual problem, and due to assumption* $q^* = \lambda^*$, *we have the following result :*

$$\bar{\lambda} \leq \inf_{\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3} q(\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3) = q^* = \lambda^* \Rightarrow \bar{\lambda} \leq \lambda^*.$$

Conversely, assume $\bar{\lambda} \leq \lambda^*$. *Thus, in view of Lemma 2.6 and Propositions 2.5 and 4.1, we can infer that the sequence* $(\boldsymbol{t}_n)_{n\in N}$ *converges to a point in Fix* T, *i.e.*

$$\lim_{n\to\infty} \boldsymbol{t}_n = \bar{\boldsymbol{t}} \in Fix\ T.$$

Since $\boldsymbol{t}_{n+1} - \boldsymbol{t}_n = \boldsymbol{d}_n^1 + \boldsymbol{d}_n^2$, *we deduce that,*

$$\lim_{n\to\infty} \boldsymbol{d}_n^1 + \boldsymbol{d}_n^2 = \lim_{n\to\infty} \boldsymbol{t}_{n+1} - \boldsymbol{t}_n = \bar{\boldsymbol{t}} - \bar{\boldsymbol{t}} = 0 \Rightarrow\ \alpha = \lim_{n\to\infty} \|\boldsymbol{d}_n^1 + \boldsymbol{d}_n^2\| = 0.$$

(*ii*) *Since* $\alpha \geq 0$, *the result in* (*ii*) *follows from* (*i*). □

The result of Theorem 4.1 is illustrated in Figure 4.2, which represents the distance of the sets $Z(\bar{\lambda})$ and $W(\bar{\lambda})$ for the cases $\bar{\lambda} \leq \lambda^*$ and $\bar{\lambda} > \lambda^*$. Figure 4.3 also shows the same idea but on the $(\lambda, \boldsymbol{t})$ plane. Consequently, the optimisation problem in (4.1.4) also reads:

$$\begin{aligned} \lambda^* = & \max_{\lambda} \lambda \\ & d(W(\lambda), Z(\lambda)) = 0 \end{aligned} \tag{4.2.12}$$

Note that for a given value of $\bar{\lambda}$, from the result in Theorem 4.1(ii), Proposition 4.1, and Lemma 2.6, we infer the following corollary :

Corollary 1 $\alpha > 0 \iff \|\boldsymbol{t}_n\| \to \infty$.

According to this corollary and Lemma 2.6 (iv), the values of $\|\frac{\boldsymbol{t}_n}{n}\|$ and $\|\boldsymbol{t}_n\|$ could be used to monitor the gap between $Z(\bar{\lambda})$ and $W(\bar{\lambda})$. We will though use other parameters to monitor this distance, as explained in the next section.

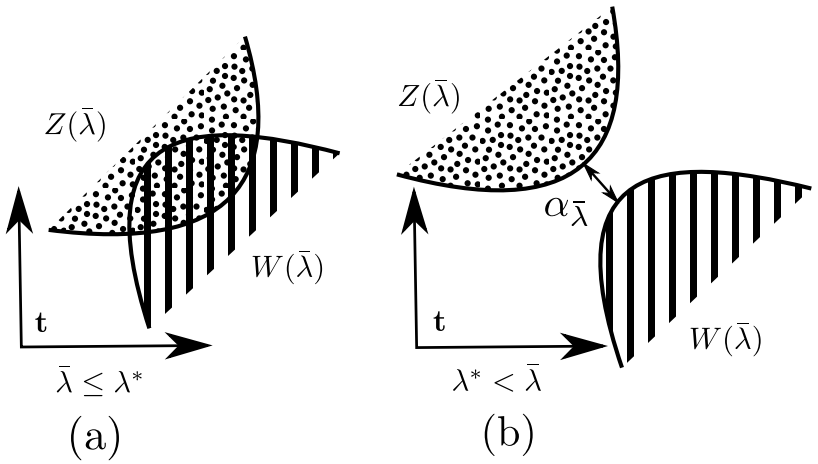


Figure 4.2.: Illustration of the sets $W(\bar{\lambda})$ and $Z(\bar{\lambda})$ for the case $\bar{\lambda} \leq \lambda^*$ and $\bar{\lambda} > \lambda^*$.

4.3. Algorithmic Implementation of AAR-based Decomposition Algorithm

As it has been explained in the previous sections, the objective is to solve an optimisation problem with the structure in (1.2.7), recasted in the form in (4.1.1).

The master problem computes at each master iteration k a new value of λ^k, while the subproblems determine whether this value λ^k is an upper or

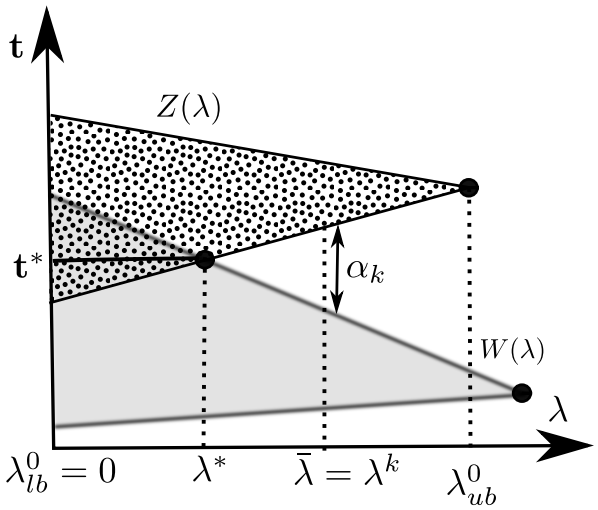


Figure 4.3.: Illustration of the sets $W(\lambda)$ and $Z(\lambda)$ on the $(\lambda, \boldsymbol{t})$ plane.

lower bound of λ^*. To determine this, a set of n_k subiterations, $n = 1, \ldots, n_k$ are required at each master iteration k.

The procedure of the master and subproblems are detailed in next two subsections, which use the following notation:

- $\alpha_n = \|\boldsymbol{d}_n^1 + \boldsymbol{d}_n^2\|, \quad \beta_n = \|\boldsymbol{d}_n^1\| + \|\boldsymbol{d}_n^2\|.$
- $\Delta\alpha_n = \alpha_n - \alpha_{n-1}.$

Also λ_{lb}^k, λ_{ub}^k denote algorithmic lower and upper bounds of λ^* in the master iteration k, respectively.

4.3.1. Master Problem

The steps that define the master problem are given in Box 1.

M0. Find real values λ_{lb}^0 and λ_{ub}^0 such that $\lambda^* \in [\lambda_{lb}^0, \lambda_{ub}^0]$. Set $k = 1$.

M1. Set $\lambda^k = (1-s)\lambda_{lb}^{k-1} + s\lambda_{ub}^{k-1}$, $s \in (0,1)$ $(k = 1, 2, \cdots)$.

M3. Solve Subproblems in Box 2 to determine whether $\lambda^k \leq \lambda^*$ or $\lambda^* < \lambda^k$.

M4. If $|\lambda_{ub}^k - \lambda_{lb}^k| \leq \epsilon_\lambda$

- $\lambda^* \approx \lambda^k$,
- Stop.

Else

- $k = k + 1$,
- Go to M1,

End.

Box 1. Master problem

In step M1, λ_{lb}^0 is a feasible solution for problem (4.1.1), which means that it exists a vector $\boldsymbol{t}$ such that $\boldsymbol{t} \in Z(\lambda_{lb}^0) \cap W(\lambda_{lb}^0)$, with $W(\lambda_{lb}^0)$ and $Z(\lambda_{lb}^0)$ defined in (4.1.3). λ_{ub}^0 is an arbitrary upper bound for λ^* that can be obtained via any possible way. In this article, we use Proposition 4.2 to obtain an upper bound of λ^*: we solve two subproblems defined in (4.1.6), and we obtain two upper bounds λ_{1ub}^0 and λ_{2ub}^0. Then we set $\lambda_{ub}^0 = \min(\lambda_{1ub}^0, \lambda_{2ub}^0)$.

In step M1, if $X_1(\lambda_{ub}^{k-1}) \neq \emptyset$ and $X_2(\lambda_{ub}^{k-1}) \neq \emptyset$ defined in (4.1.2), we clearly have that $X_1(\lambda^k) \neq \emptyset$ and $X_2(\lambda^k) \neq \emptyset$, since $\boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{g}_1, \boldsymbol{g}_2$ are affine functions and K_1, K_2 are convex sets. The constant value $s = \frac{1}{2}$ has been employed for the update of λ^k in Box 1, Step M1.

4.3.2. Subproblems

The iterative process of each subproblem is summarised in Box 2.

S1. Set $\bar{\lambda} = \lambda^k$, $\boldsymbol{t}_0^k = \boldsymbol{t}^{k-1}$, $n = 0$.

S2. Solve Subproblem 1 defined in (4.2.2). Obtain $\boldsymbol{d}_n^1$ and set $\boldsymbol{t}_n^k = \boldsymbol{t}_n^k + 2\boldsymbol{d}_n^1$.

S3. Solve Subproblem 2 defined in (4.2.4). Obtain $\boldsymbol{d}_n^2$ and set $\boldsymbol{t}_n^k = \boldsymbol{t}_n^k + 2\boldsymbol{d}_n^2$.

S4. Set $\beta_n = \|\boldsymbol{d}_n^1\| + \|\boldsymbol{d}_n^2\|$ and $\alpha_n = \|\boldsymbol{d}_n^1 + \boldsymbol{d}_n^2\|$.

S5. If $\beta_n < \beta_{n-1}$ or $\alpha_n \leq \epsilon_0$, then $\lambda^k \leq \lambda^*$:

- $\lambda^k_{lb} = \lambda^k,\ \lambda^k_{ub} = \lambda^{k-1}_{ub}$. Update Δ^k.
- Go to Box 1.M1.

Else if $\beta_n > \beta_{n-1}$ and $\alpha_n > \frac{\Delta^{k-1}}{k}$ and $\frac{|\Delta\alpha_n|}{|\alpha_n|} < \epsilon_1$, then $\lambda^k > \lambda^*$:

- $\lambda^k_{lb} = \lambda^{k-1}_{lb}$, $\lambda^k_{ub} = \lambda^k$. Update Δ^k.
- Go to Box 1.M1.

Else

- $\boldsymbol{t}^k_n = \frac{\boldsymbol{t}^k_{n-1} + \boldsymbol{t}^k_n}{2}$, $n = n + 1$,
- Go to S2.

End

Box 2. Subproblems 1 and 2.

The algorithm in Box 2 uses the control parameters β_n, α_n and Δ^k to detect whether λ^k is an upper bound or a lower bound of λ^* . Indeed, the numerical results show that β_n decreases when λ^k is a lower bound of λ^*. The values of β_n satisfy in fact the following corollary:

Corollary 2 *Assume that $\bar{\lambda}$ is a given real value and $W = W(\bar{\lambda})$, $Z = Z(\bar{\lambda})$ defined in* (4.1.3)*. Then the following relations hold.*

(i) **:** *If* $\mathbf{0} \in \text{int}(W \cap Z)$ *then* $\lim_{n\to\infty} \beta_n = 0$.

(ii) **:** *If* $\lim_{n\to\infty} \beta_n = 0$ *then* $W \cap Z \neq \emptyset$.

Proof 10 (i) *: Since* $\mathbf{0} \in \text{int}(W \cap Z)$*, then $W \cap Z \neq \emptyset$, and it follows that Fix $P_W \neq \emptyset$, Fix $P_Z \neq \emptyset$, and also Fix $P_W P_Z$ = Fix $P_W \cap$ Fix P_Z. According to the AAR method, since $W \cap Z \neq \emptyset$, we have that*

$$\lim_{n\to\infty} \boldsymbol{t}_n = \bar{\boldsymbol{t}} \in \mathit{Fix}\ T.$$

On the other hand, since $\mathbf{0} \in \text{int}(W \cap Z)$*, then Fix $T = W \cap Z$ [7], and therefore we have in turn that,*

$$P_Z(\bar{\boldsymbol{t}}) = \bar{\boldsymbol{t}},\ P_W(\bar{\boldsymbol{t}}) = \bar{\boldsymbol{t}},\ R_Z(\bar{\boldsymbol{t}}) = 2P_Z(\bar{\boldsymbol{t}}) - \bar{\boldsymbol{t}} = \bar{\boldsymbol{t}}. \tag{4.3.1}$$

In view of (4.2.3), (4.2.6) *and* (4.3.1) *, the following results can be derived:*

$$\begin{aligned} \lim_{n\to\infty} \boldsymbol{d}_n^1 &= \lim_{n\to\infty} P_Z(\boldsymbol{t}_n) - \boldsymbol{t}_n = P_Z(\bar{\boldsymbol{t}}) - \bar{\boldsymbol{t}} = \mathbf{0} \\ \lim_{n\to\infty} \boldsymbol{d}_n^2 &= \lim_{n\to\infty} \boldsymbol{t}_{n+1} - \boldsymbol{t}_n - \boldsymbol{d}_n^1 = \mathbf{0}. \end{aligned} \tag{4.3.2}$$

Consequently, in view of (4.3.2), *we infer that*

$$\lim_{n\to\infty} \beta_n = \lim_{n\to\infty} \|\boldsymbol{d}_n^1\| + \|\boldsymbol{d}_n^2\| = 0. \tag{4.3.3}$$

(ii) : For each iteration n we have that $\alpha_n = \|\boldsymbol{d}_n^1 + \boldsymbol{d}_n^2\| \leq \|\boldsymbol{d}_n^1\| + \|\boldsymbol{d}_n^2\| = \beta_n$. *Therefore,*

$$0 = \lim_{n\to\infty} \beta_n \geq \lim_{n\to\infty} \alpha_n = \alpha \geq 0, \tag{4.3.4}$$

which implies that $\alpha = 0$. *Consequently, in view of Proposition 4.1 and Theorem 4.1, we infer that* $W \cap Z \neq \emptyset$. □

We note that the update of λ^k in step M1 of the master problem mimics the update process of the bisection method. Other faster updates could be envisaged, but at the expense of estimating more accurately the distance between the sets $W(\lambda^k)$ and $Z(\lambda^k)$. In our implementation of the subproblems, we just detect from the trends of β_n and α_n whether the set $W(\lambda^k) \cap Z(\lambda^k)$ is empty or not, but do not actually compute the distance $d(W(\lambda^k), Z(\lambda^k))$. The accurate computation of the distance would require far more subiterations, and in the authors experience, this extra cost does not compensate the gain when more sophisticated updates for λ^k are implemented.

The algorithms in Box 1 and 2 use three tolerance parameters: ϵ_λ, ϵ_0 and ϵ_1. Their meaning is the following:

- ϵ_λ: this is the desired tolerance for the objective λ, and it is such that $\lambda^* \in [\lambda_{lb}, \lambda_{ub}]$, with $|\lambda_{ub} - \lambda_{lb}| < \epsilon_\lambda$.

- ϵ_0: is a tolerance for $d(W(\lambda^k), Z(\lambda^k))$. If $d(W(\lambda^k), Z(\lambda^k)) < \epsilon_0$, we will consider that $W(\lambda^k) \cap Z(\lambda^k) \neq \emptyset$.

- ϵ_1 is used to detect when the sequence α_n (generated by the solution of subproblems) has converged

In all our numerical tests, we have used the values $(\epsilon_\lambda, \epsilon_0, \epsilon_1) = (10^{-3}, 5 * 10^{-4}, 10^{-2})$. The vectors $(\boldsymbol{t}^k, \boldsymbol{x}_1^k, \boldsymbol{x}_2^k)$ resulting from the subproblems for the highest (latest) value of λ_{lb}^k furnish the algorithmic approximations of $(\boldsymbol{t}^*, \boldsymbol{x}_1^*, \boldsymbol{x}_2^*)$. In addition, the algorithm in Box 2 uses the parameter Δ^k to control the convergence of α_n, which is an approximation to $d(W(\lambda^k), Z(\lambda^k)$. We suggest two possible updates:

U1:

- If $\lambda^k \leq \lambda^*$, then $\Delta^k = \lambda_{ub}^{k-1} - \lambda^k$.
- If $\lambda^k > \lambda^*$, then $\Delta^k = \lambda^k - \lambda_{lb}^{k-1}$.

U2:

- If $\lambda^k \leq \lambda^*$, then $\Delta^k = \Delta^{k-1}$.
- If $\lambda^k > \lambda^*$, then $\Delta^k = s||\boldsymbol{d}_n^1 - \boldsymbol{d}_n^2||$.

Figure 4.4 illustrates the update proposed in **U1**. The update given in **U2**, will be justified in the next section.

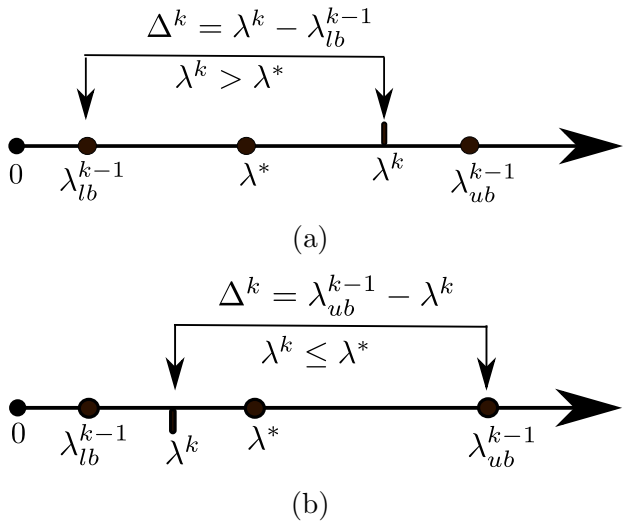


Figure 4.4.: Updating parameter Δ^k. (a): λ^k is considered as an upper bound, (b): λ^k is considered as a lower bound.

4.3.3. Justification of Update U2 for Δ^k

The aim of the update suggested here is to avoid obtaining false lower bounds, false upper bounds, or endless loops. Indeed, large values of Δ^k, will prevent the algorithm from stopping when λ^k is an upper bound (endless loop), and small values of Δ^k may give false upper bounds, i.e. a value $\lambda^k < \lambda^*$ that is detected as an upper bound.

Throughout this subsection, λ_{lb} and λ_{ub} denote a feasible lower bound and an upper bound of problem (4.1.1) respectively. Note that we may have that $\lambda_{lb} = \lambda^*$.

Proposition 4.3 *Let us denote by $\alpha(\lambda)$ the distance between two sets $Z(\lambda)$ and $W(\lambda)$ defined in (4.1.3), as a function of λ i.e.*

$$\alpha(\lambda) = d(Z(\lambda), W(\lambda)) = \inf_{\substack{\boldsymbol{x}_1 \in X_1(\lambda) \\ \boldsymbol{x}_2 \in X_2(\lambda)}} ||\boldsymbol{g}_1(\boldsymbol{x}_1) + \boldsymbol{g}_2(\boldsymbol{x}_2)||, \tag{4.3.5}$$

where $X_1(\lambda)$ and $X_2(\lambda)$ are defined in (4.1.2). Then the following hold:

(i) If $\lambda_s = (1-s)\lambda_{lb} + s\lambda_{ub}$ with $s \in (0,1)$, then $\alpha(\lambda_s) \leq s\alpha(\lambda_{ub})$.

(ii) $\alpha(\lambda)$ is strictly increasing on $[\lambda^*, \lambda_{ub}]$.

(iii) $\lim_{\lambda \to \lambda^*} \alpha(\lambda) = 0$ with $\lambda \in (\lambda^*, \lambda_{ub}]$.

(iv) Let $s \in (0,1)$ be a constant parameter that has been used to generate a sequence $(\lambda_{ub}^k)_{k \in N}$ of upper bounds of λ^* in the following manner:

$$\begin{aligned} &\lambda_{ub}^0 = \lambda_{ub} \\ &\lambda_{ub}^k = (1-s)\lambda_{lb} + s\lambda_{ub}^{k-1}, k = 1, 2, \cdots \end{aligned} \tag{4.3.6}$$

Then $\alpha(\lambda_{ub}^k) \leq (s)^k \alpha(\lambda_{ub})$.

(v) $\alpha(\lambda)$ is a convex function on $[\lambda^*, \lambda_{ub}]$.

Proof 11 *(i) :Let $(\boldsymbol{x}_{1ub}, \boldsymbol{x}_{2ub}) \in X_1(\lambda_{ub}) \times X_2(\lambda_{ub})$ and $(\boldsymbol{x}_{1lb}, \boldsymbol{x}_{2lb}) \in X_1(\lambda_{lb}) \times X_2(\lambda_{lb})$ with $\boldsymbol{g}_1(\boldsymbol{x}_{1lb}) + \boldsymbol{g}_2(\boldsymbol{x}_{2lb}) = \boldsymbol{0}$. Set :*

$$(\boldsymbol{x}_i, \lambda_s) = (1-s)(\boldsymbol{x}_{ilb}, \lambda_{lb}) + s(\boldsymbol{x}_{iub}, \lambda_{ub}) \, (i = 1, 2). \tag{4.3.7}$$

According to the definition of the set $X_i(\lambda_s)$, the vector $\boldsymbol{x}_i$ obviously belongs to $X_i(\lambda_s), (i = 1, 2)$.

Since the functions $\boldsymbol{g}_1$ *and* $\boldsymbol{g}_2$ *are both affine maps then we have:*

$$\begin{array}{l}\boldsymbol{g}_1(\boldsymbol{x}_1)+\boldsymbol{g}_2(\boldsymbol{x}_2)= \\ \boldsymbol{g}_1((1-s)\boldsymbol{x}_{1lb}+s\boldsymbol{x}_{1ub})+\boldsymbol{g}_2((1-s)\boldsymbol{x}_{2lb}+s\boldsymbol{x}_{2ub})= \\ (1-s)(\boldsymbol{g}_1(\boldsymbol{x}_{1lb})+\boldsymbol{g}_2(\boldsymbol{x}_{2lb}))+s(\boldsymbol{g}_1(\boldsymbol{x}_{1ub})+\boldsymbol{g}_2(\boldsymbol{x}_{2ub}))= \\ s(\boldsymbol{g}_1(\boldsymbol{x}_{1ub})+\boldsymbol{g}_2(\boldsymbol{x}_{2ub})).\end{array} \tag{4.3.8}$$

Thus in view of (4.3.8) *and* (4.3.5) *we have that :*

$$\alpha(\lambda_s)\leq||\boldsymbol{g}_1(\boldsymbol{x}_1)+\boldsymbol{g}_2(\boldsymbol{x}_2)||=s||\boldsymbol{g}_1(\boldsymbol{x}_{1ub})+\boldsymbol{g}_2(\boldsymbol{x}_{2ub})||,$$

and since $\boldsymbol{x}_{iub}$ *is an arbitrary element of* $X_i(\lambda_{ub})$*,*

$$\alpha(\lambda_s)\leq s\alpha(\lambda_{ub}).$$

(ii) Assume that $\lambda^*<\lambda_1<\lambda_2\leq\lambda_{ub}$*. Then there exists* $s\in(0,1)$ *such that* $\lambda_1=(1-s)\lambda^*+s\lambda_2$*, and according to* (i) *we have that* $\alpha(\lambda_1)\leq s\alpha(\lambda_2)<\alpha(\lambda_2)$*, i.e.* $\alpha(\lambda)$ *is strictly increasing on* $[\lambda^*,\lambda_{ub}]$*.*

(iii) The results in (iii) *and* (iv) *easily follow from* (i)*.*

(v) Fix λ_1 *and* λ_2 *in* $(\lambda^*,\lambda_{ub}]$*, and* $t\in(0,1)$*. Set* $\lambda=(1-t)\lambda_1+t\lambda_2$*. Now let* $(\boldsymbol{z}_1,\boldsymbol{z}_2)\in X_1(\lambda_1)\times X_2(\lambda_1)$ *and* $(\boldsymbol{w}_1,\boldsymbol{w}_2)\in X_1(\lambda_2)\times X_2(\lambda_2)$*. Then according to the definition of the set* $X_i(\lambda_j)$ *:*

$$(1-t)\boldsymbol{z}_i+t\boldsymbol{w}_i\in X_i(\lambda),(i=1,2). \tag{4.3.9}$$

It follows from definition of $\alpha(\lambda)$ *in* (4.3.5)*, that*

$$\alpha(\lambda)\leq||\boldsymbol{g}_1((1-t)\boldsymbol{z}_1+t\boldsymbol{w}_1)+\boldsymbol{g}_2((1-t)\boldsymbol{z}_2+t\boldsymbol{w}_2)||. \tag{4.3.10}$$

Since $\boldsymbol{g}_i,(i=1,2)$ *is an affine map, we have that:*

$$\begin{array}{l}||\boldsymbol{g}_1((1-t)\boldsymbol{z}_1+t\boldsymbol{w}_1)+\boldsymbol{g}_2((1-t)\boldsymbol{z}_2+t\boldsymbol{w}_2)||= \\ ||(1-t)(\boldsymbol{g}_1(\boldsymbol{z}_1)+\boldsymbol{g}_2(\boldsymbol{z}_2))+t(\boldsymbol{g}_1(\boldsymbol{w}_1)+\boldsymbol{g}_2(\boldsymbol{w}_2))||\leq \\ (1-t)||\boldsymbol{g}_1(\boldsymbol{z}_1)+\boldsymbol{g}_2(\boldsymbol{z}_2)||+t||\boldsymbol{g}_1(\boldsymbol{w}_1)+\boldsymbol{g}_2(\boldsymbol{w}_2)||.\end{array} \tag{4.3.11}$$

Thus, in view of (4.3.10) *and* (4.3.11)*,*

$$\alpha(\lambda)=\alpha((1-t)\lambda_1+t\lambda_2)\leq(1-t)\alpha(\lambda_1)+t\alpha(\lambda_2),$$

i.e. $\alpha(\lambda)$ *is convex.* □

According to Proposition 4.3 (i), parameter Δ^k is approximately proportional to the distance between two sets $W(\lambda^k)$ and $Z(\lambda^k)$. Figure 4.5, illustrates this fact.

Therefore, in view of Proposition 4.3 (i), we could update the parameter Δ^k as follows. Let λ^k be a convex combination of λ_{lb}^{k-1} and λ_{ub}^{k-1}, i.e. $\lambda^k = (1-s)\lambda_{lb}^{k-1} + s\lambda_{ub}^{k-1}$, with constant $s \in (0,1)$ then:

- If $\lambda^k > \lambda^*$, set $\alpha(\lambda^k) \approx ||\boldsymbol{d}_{n_k}^1 + \boldsymbol{d}_{n_k}^2||$, and $\Delta^k = s\alpha(\lambda^k)$.
- If $\lambda^k \leq \lambda^*$, we set $\Delta^k = \Delta^{k-1}$.

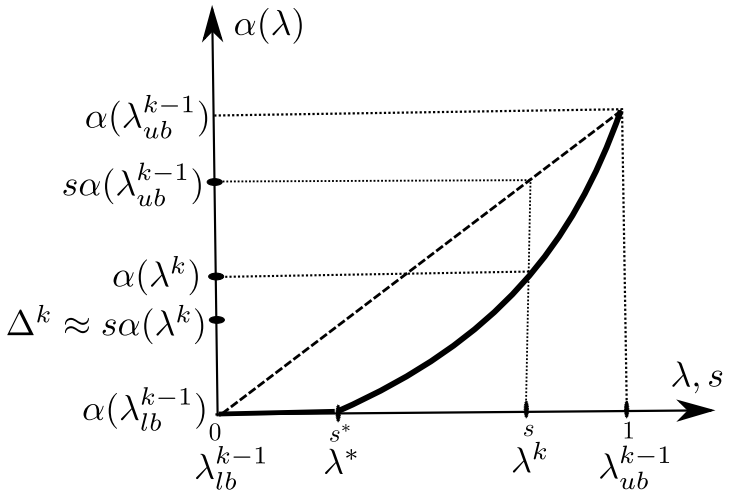


Figure 4.5.: Updating parameter Δ^k, when λ^k is an upper bound.

In other words, when λ^k is an upper bound, we update Δ^k based on the distance between two sets $W(\lambda^k)$ and $Z(\lambda^k)$ instead of using the distance between λ_{lb}^{k-1} and λ_{ub}^{k-1}, as illustrated in Figure 4.4.

In fact some other value $\bar{s} < s$ could be used to update Δ^k when $\lambda^k > \lambda^*$, i.e. $\Delta^k = \bar{s}\alpha(\lambda^k)$, $\bar{s} < s$.

4.4. Mechanical Interpretation of AAR-based Decomposition Method

Consider the following global form of the problem in (1.2.5):

$$\lambda^* = \max_{\boldsymbol{\sigma},\lambda} \lambda$$
$$\begin{cases} \nabla \cdot \boldsymbol{\sigma}_i^{e,LB} + \lambda \boldsymbol{f}_i^e = \mathbf{0} & \text{in } \Omega_i^e\,, \forall\, e = 1, \cdots, ne_i \\ (\boldsymbol{\sigma}_i^{e,LB} - \boldsymbol{\sigma}_i^{e',LB}) \cdot \boldsymbol{n}_i^{\xi_e^{e'}} = \mathbf{0}, & \forall\, \xi_e^{e'} \in \xi_i^O \\ \boldsymbol{\sigma}_i^{e,LB} \cdot \boldsymbol{n}_i^{\xi_e^N} = \lambda \boldsymbol{g}^{\xi_e^N} & \forall\, \xi_e^N \in \xi_i^N \\ \boldsymbol{\sigma}_i^{e,LB} \cdot \boldsymbol{n}_i^{\xi_e^N} = (-1)^i \boldsymbol{t} & \forall\, \xi_e^N \in \xi_i^{\bar{N}} \\ \boldsymbol{\sigma}_i^{e,LB} \in \mathcal{B} & \text{in } \Omega_i^e,\ \forall\, e = 1, \cdots, ne_i \end{cases} \qquad (i = 1, 2) \tag{4.4.1}$$

where $\xi_i^{\bar{N}}$ indicates the fictitious Neumann, at each subdomain that is, the common domain of the two subdomains (see Figure 4.6). $\boldsymbol{t}$ is in fact the traction vector $\boldsymbol{t} = \boldsymbol{\sigma} \cdot \boldsymbol{n}$ along this common boundary. By solving the following two subproblems, where $\boldsymbol{t}$ is left as a free variable, not necessarily equal of each subdomain, we obtain upper bounds for λ^* :

$$\lambda_i^* = \max_{\boldsymbol{\sigma},\lambda,\boldsymbol{t}} \lambda$$
$$\begin{cases} \nabla \cdot \boldsymbol{\sigma}_i^{e,LB} + \lambda \boldsymbol{f}_i^e = \mathbf{0} & \text{in } \Omega_i^e\,, \forall\, e = 1, \cdots, ne_i \\ (\boldsymbol{\sigma}_i^{e,LB} - \boldsymbol{\sigma}_i^{e',LB}) \cdot \boldsymbol{n}_i^{\xi_e^{e'}} = \mathbf{0}, & \forall\, \xi_e^{e'} \in \xi_i^O \\ \boldsymbol{\sigma}_i^{e,LB} \cdot \boldsymbol{n}_i^{\xi_e^N} = \lambda \boldsymbol{g}^{\xi_e^N} & \forall\, \xi_e^N \in \xi_i^N \\ \boldsymbol{\sigma}_i^{e,LB} \cdot \boldsymbol{n}_i^{\xi_e^N} = (-1)^i \boldsymbol{t} & \forall\, \xi_e^N \in \xi_i^{\bar{N}} \\ \boldsymbol{\sigma}_i^{e,LB} \in \mathcal{B} & \text{in } \Omega_i^e,\ \forall\, e = 1, \cdots, ne_i, \end{cases} \tag{4.4.2}$$

that is, $\lambda^* \leq \lambda_i^*, \quad i = 1, 2$. At each master iteration, we choose a value of the load factor λ^k, and detect whether, there exists a common value of the traction $\boldsymbol{t}$ such that the global domain is in equilibrium. In other words, we detect whether the two subdomain can be in equilibrium for the given load factor λ^k. More specifically, we set $\lambda = \bar{\lambda}$ and solve the following two optimisation problems sequentially, in order to detect if $\bar{\lambda} \leq \lambda^*$ or $\lambda^* < \bar{\lambda}$.

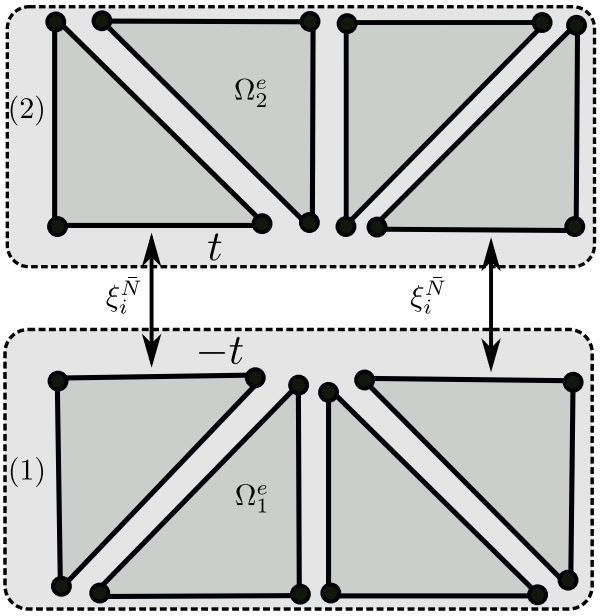


Figure 4.6.: Decomposition of global problem into two subproblems. The global variables are the boundary traction $\boldsymbol{t}$ at the fictitious Neumann condition and global load factor λ.

Subproblem 1

$$\|\boldsymbol{d}_n^1\| = \min_{\boldsymbol{\sigma},\boldsymbol{d}^1} \|\boldsymbol{d}^1\|$$

$$\begin{cases} \nabla \cdot \boldsymbol{\sigma}_1^{e,LB} + \bar{\lambda}\boldsymbol{f}_1^e = \boldsymbol{0} & \text{in } \Omega_1^e\,, \forall\, e = 1, \cdots, ne_1 \\ (\boldsymbol{\sigma}_1^{e,LB} - \boldsymbol{\sigma}_1^{e',LB}) \cdot \boldsymbol{n}_1^{\xi_e^{e'}} = \boldsymbol{0}, & \forall\, \xi_e^{e'} \in \xi_1^O \\ \boldsymbol{\sigma}_1^{e,LB} \cdot \boldsymbol{n}_1^{\xi_e^N} = \bar{\lambda}\boldsymbol{g}^{\xi_e^N} & \forall\, \xi_e^N \in \xi_1^N \\ \boldsymbol{\sigma}_1^{e,LB} \cdot \boldsymbol{n}_1^{\xi_e^N} = \boldsymbol{t}_n + \boldsymbol{d}^1 & \forall\, \xi_e^N \in \xi_1^{\bar{N}} \\ \boldsymbol{\sigma}_1^{e,LB} \in \mathcal{B} & \text{in } \Omega_1^e,\, \forall\, e = 1, \cdots, ne_1. \end{cases} \tag{4.4.3}$$

Subproblem 2

$$\|\boldsymbol{d}_n^2\| = \min_{\boldsymbol{\sigma},\boldsymbol{d}^2} \|\boldsymbol{d}^2\|$$

$$\begin{cases} \nabla \cdot \boldsymbol{\sigma}_2^{e,LB} + \bar{\lambda}\boldsymbol{f}_2^e = \boldsymbol{0} & \text{in } \Omega_2^e\,, \forall\, e = 1, \cdots, ne_2 \\ (\boldsymbol{\sigma}_2^{e,LB} - \boldsymbol{\sigma}_2^{e',LB}) \cdot \boldsymbol{n}_2^{\xi_e^{e'}} = \boldsymbol{0}, & \forall\, \xi_e^{e'} \in \xi_2^O \\ \boldsymbol{\sigma}_2^{e,LB} \cdot \boldsymbol{n}_2^{\xi_e^N} = \bar{\lambda}\boldsymbol{g}^{\xi_e^N} & \forall\, \xi_e^N \in \xi_2^N \\ \boldsymbol{\sigma}_2^{e,LB} \cdot \boldsymbol{n}_2^{\xi_e^N} = -\boldsymbol{t}_n - 2\boldsymbol{d}_n^1 - \boldsymbol{d}^2 & \forall\, \xi_e^N \in \xi_2^{\bar{N}} \\ \boldsymbol{\sigma}_2^{e,LB} \in \mathcal{B} & \text{in } \Omega_2^e,\ \forall\, e = 1, \cdots, ne_2. \end{cases} \tag{4.4.4}$$

Update $\boldsymbol{t}$:

$$\boldsymbol{t}_{n+1} = \boldsymbol{t}_n + \boldsymbol{d}_n^1 + \boldsymbol{d}_n^2.$$

If $\boldsymbol{t}_{n+1} = \boldsymbol{t}_n$ i.e. $||\boldsymbol{d}_n^1 - \boldsymbol{d}_n^2|| = 0$, then a common traction field $\boldsymbol{t} = \boldsymbol{t}_n$ that equilibrates the two subdomains has been found and $\lambda^k \leq \lambda^*$. Otherwise, if for all n, $||\boldsymbol{d}_n^1 + \boldsymbol{d}_n^2|| > 0$, no common value of $\boldsymbol{t}$ can be found for the chosen load factor λ^k and therefore $\lambda^* < \lambda^k$.

Note: The algorithm that computes the approximate value of λ^* has been implemented both in *MATLAB* and *FORTRAN90*.

In the next section the AAR-based decomposition method is illustrated and is applied for problems in limit analysis.

4.5. Numerical Results

4.5.1. Illustrative Example

We illustrate the AAR-based decomposition technique explained in previous section with a toy nonlinear convex problem given by,

$$\begin{aligned} \lambda^* = \max_{\lambda,\boldsymbol{x}_1,\boldsymbol{x}_2}\ & \lambda \\ & \mathbf{A}_1\boldsymbol{x}_1 + \lambda\boldsymbol{F}_1 = \boldsymbol{b}_1 \\ & \mathbf{A}_2\boldsymbol{x}_2 + \lambda\boldsymbol{F}_2 = \boldsymbol{b}_2 \\ & \mathbf{G}_1\boldsymbol{x}_1 + \mathbf{G}_2\boldsymbol{x}_2 = \boldsymbol{b} \\ & \boldsymbol{x}_1 \in K_1,\ \boldsymbol{x}_2 \in K_2,\ \lambda \in \Re, \end{aligned} \tag{4.5.1}$$

where K_1 and K_2 are second-order cones and $\boldsymbol{x}_1 \in \Re^4$, $\boldsymbol{x}_2 \in \Re^4$. The values of matrix $\mathbf{A}_i, \mathbf{G}_i$ and vectors $\boldsymbol{F}_i, \boldsymbol{b}_i$ and $\boldsymbol{b}$ are given next, and the optimal

value of this problem is $\lambda^* = 1.1965$.

$$\max_{\lambda,\boldsymbol{x}_1,\boldsymbol{x}_2} \ \lambda$$

$$\overbrace{\begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}}^{\mathbf{A}_1} \boldsymbol{x}_1 + \lambda \overbrace{\begin{Bmatrix} 1 \\ 1 \end{Bmatrix}}^{\boldsymbol{F}_1} = \overbrace{\begin{Bmatrix} 1.2 \\ 2.4 \end{Bmatrix}}^{\boldsymbol{b}_1}$$

$$\overbrace{\begin{bmatrix} -1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}}^{\mathbf{A}_2} \boldsymbol{x}_2 + \lambda \overbrace{\begin{Bmatrix} 1 \\ 1 \end{Bmatrix}}^{\boldsymbol{F}_2} = \overbrace{\begin{Bmatrix} 3.2 \\ 0.2 \end{Bmatrix}}^{\boldsymbol{b}_2}$$

$$\overbrace{\begin{bmatrix} 0 & 2 & 1 & 0 \\ 0 & 2 & 0 & 1 \end{bmatrix}}^{\mathbf{G}_1} \boldsymbol{x}_1 + \overbrace{\begin{bmatrix} 0 & 2 & 1 & 1 \\ 0 & 2 & 0 & 1 \end{bmatrix}}^{\mathbf{G}_2} \boldsymbol{x}_2 = \overbrace{\begin{Bmatrix} 1.2 \\ 6.2 \end{Bmatrix}}^{\boldsymbol{b}} = \overbrace{\begin{Bmatrix} -0.8 \\ 3.2 \end{Bmatrix}}^{\boldsymbol{b}_3} + \overbrace{\begin{Bmatrix} 2 \\ 3 \end{Bmatrix}}^{\boldsymbol{b}_4}$$

$$\boldsymbol{x}_1 \in K_1, \boldsymbol{x}_2 \in K_2, \lambda \in \Re$$

By selecting arbitrary vectors $\boldsymbol{b}_3$ and $\boldsymbol{b}_4$ such that $\boldsymbol{b} = \boldsymbol{b}_3 + \boldsymbol{b}_4$ and introducing a new variable $\boldsymbol{t}$, the problem (4.5.1) is written in the following form,

$$\begin{aligned} \max_{\lambda,\boldsymbol{x}_1,\boldsymbol{x}_2,\lambda,\boldsymbol{t}} \ & \lambda \\ & \mathbf{A}_1\boldsymbol{x}_1 + \lambda\boldsymbol{F}_1 = \boldsymbol{b}_1 \\ & \mathbf{G}_1\boldsymbol{x}_1 - \boldsymbol{b}_3 = \boldsymbol{t} \\ & \mathbf{A}_2\boldsymbol{x}_2 + \lambda\boldsymbol{F}_2 = \boldsymbol{b}_2 \\ & \mathbf{G}_2\boldsymbol{x}_2 - \boldsymbol{b}_4 = -\boldsymbol{t} \\ & \boldsymbol{x}_1 \in K_1, \ \boldsymbol{x}_2 \in K_2, \ \lambda \in \Re, \ \boldsymbol{t} \in \Re^m. \end{aligned} \tag{4.5.2}$$

In view of (4.5.2), for any given $\lambda = \bar{\lambda}$, we have the two feasible sets defined in (4.1.3), which now take the following form :

$$Z(\bar{\lambda}) = \{\boldsymbol{t} \ | \mathbf{G}_1\boldsymbol{x}_1 - \boldsymbol{b}_3 = \boldsymbol{t}, \ \mathbf{A}_1\boldsymbol{x}_1 + \bar{\lambda}\boldsymbol{F}_1 = \boldsymbol{b}_1 \text{ for some } \boldsymbol{x}_1 \in K_1\},$$
$$W(\bar{\lambda}) = \{\boldsymbol{t} \ | \mathbf{G}_2\boldsymbol{x}_2 - \boldsymbol{b}_4 = -\boldsymbol{t}, \ \mathbf{A}_2\boldsymbol{x}_2 + \bar{\lambda}\boldsymbol{F}_2 = \boldsymbol{b}_2 \text{ for some } \boldsymbol{x}_2 \in K_2\}.$$

For solving this problem we first take $(\lambda^0_{lb}, \boldsymbol{t}) = (0, \mathbf{0})$ and, according to Proposition 4.2, after solving the two optimisation problem in (4.1.6), we obtain two upper bounds $(\lambda_{1ub}, \lambda_{2ub}) = (1.2000, 3.2000)$. Then we set $\lambda^0_{ub} = \min\{\lambda_{1ub}, \lambda_{2ub}\} = 1.2000$, and consequently $\lambda^* \in [\lambda^0_{lb}, \lambda^0_{ub}] = [0, 1.2000]$. The algorithm comes to an end when $|\Delta^k| = |\lambda^k_{up} - \lambda^k_{lb}| < \epsilon_\lambda = 10^{-3}$.

The numerical results of the algorithm are reported in Table 4.1. It can be observed that after a total of 24 subiterations, the gap between the λ^* and the approximate value of λ^* is 3×10^{-4}. Figure 4.7 shows the optimal solution of the toy problem and λ^k as a function of the number of subiterations.

Toy Problem						
$\lambda^* = \mathbf{1.1965}$						
k	λ_{lb}^{k-1}	λ_{ub}^{k-1}	λ^k	$\frac{\lvert\lambda*-\lambda^k\rvert}{\lvert\lambda*\rvert}$	n_k	Δ^{k-1}
1	0	1.2000	0.6000	0.4985	2	1.2000
2	0.6000	1.2000	0.9000	0.2478	2	0.6000
3	0.9000	1.2000	1.0500	0.1224	2	0.3000
4	1.0500	1.2000	1.1250	0.0597	2	0.1500
5	1.1250	1.2000	1.1625	0.0284	2	0.0750
6	1.1625	1.2000	1.1812	0.0127	3	0.0375
7	1.1812	1.2000	1.1906	0.0049	2	0.0187
8	1.1906	1.2000	1.1953	0.0010	2	0.0094
9	1.1953	1.2000	**1.1977**	0.0010	2	0.0047
10	1.1953	1.1977	1.1965	0.0000	2	0.0023
11	1.1965	1.1977	**1.1971**	0.0005	3	0.0012
12	1.1965	1.1971	$\lambda^* \simeq 1.1968$	0.0003	$\sum_{k=1}^{12} n_k = 24$	0.0007

Table 4.1.: Results of toy problem by using AAR-based decomposition method. Number in bold font indicate upper bounds of λ^*.

4.5.2. Example 2

We consider an optimisation problem with the same structure as in (4.5.1):

$$\begin{aligned} \lambda^* = \max_{\lambda,\boldsymbol{x}_1,\boldsymbol{x}_2} \ & \lambda \\ & \mathbf{A}_1\boldsymbol{x}_1 + \lambda\boldsymbol{F}_1 = \boldsymbol{b}_1 \\ & \mathbf{A}_2\boldsymbol{x}_2 + \lambda\boldsymbol{F}_2 = \boldsymbol{b}_2 \\ & \mathbf{G}_1\boldsymbol{x}_1 + \mathbf{G}_2\boldsymbol{x}_2 = \mathbf{0} \\ & \boldsymbol{x}_1 \in K_1,\ \boldsymbol{x}_2 \in K_2,\ \lambda \geq 0. \end{aligned} \tag{4.5.3}$$

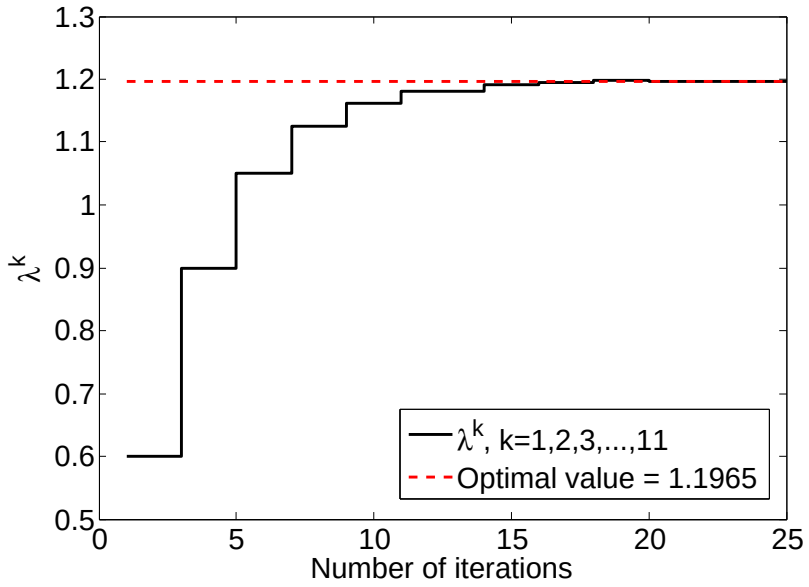


Figure 4.7.: Evolution of λ^k for the toy problem.

In this case, K_1 and K_2 are given by

$$K_1 = \Re^{n_1} \times K_{11} \times K_{12} \times \cdots \times K_{1p},\ p \geq 1,$$
$$K_2 = \Re^{n2} \times K_{21} \times K_{22} \times \cdots \times K_{2q},\ q \geq 1,$$

where K_{ij} are three-dimensional second-order cones, and matrices $\mathbf{A}_i$, and vectors $\boldsymbol{F}_i, \boldsymbol{b}_i$ are those resulting from a discretised limit analysis problem similar to those in [33, 37]. The problem (4.5.3) can be written in the general standard form:

$$\begin{aligned} \lambda^* = \max_{\boldsymbol{x}}\ & \boldsymbol{c} \cdot \boldsymbol{x} \\ & \mathbf{A}\boldsymbol{x} = \boldsymbol{b} \\ & \boldsymbol{x} \in K, \end{aligned} \tag{4.5.4}$$

where K is a convex cone and $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \lambda)$.

Now we apply the AAR-based decomposition method for problems with different size given in Table 4.2, where n, m denote the number of rows and columns of matrix $\mathbf{A}$ in (4.5.4), and $\sum_{k=1} n_k$ is the total number of subiterations.

The problems have been solved using the optimisation software Mosek [1] and SPT3 [47]. In all the cases, the former required less CPU time and yielded more accurate results (smaller gap between primal and dual

problems). For the larger problem 5, SDPT3 failed to give accurate results after the master iteration $k = 2$.

Problem	n	m	CPU Time [s]		$\sum_{k=1} n_k$	
			SDP3	MOSEK	SDP3	MOSEK
1	2240	2401	163	30	45	45
2	4368	4705	281	84	43	42
3	8880	9601	913	243	40	42
4	12768	13825	1754	543	47	49
5	14979	16225	*	707	*	48

Table 4.2.: Size, CPU time and total subiterations of each problem solved using Mosek [1] and SDPT3 [47].

Like in the toy problem, we have used Proposition 4.2 in order to obtain an upper bound of λ^* at the master iteration zero λ^0_{ub}. The numerical results of problems 1-5 are reported in Tables 4.3-4.7, respectively. Figure 4.8 shows λ^*, λ^k and β_n related to Problem 3 as a function of number of subiterations. For the other problems, similar trends of these variables have been obtained.

Figure 4.9 shows the error of the objective λ^k as a function of the master iterations. The linear convergence trend is characteristic of the bisection method employed for the update of λ^k in step M1, Box 1. Other more sophisticated updates have been tested (secant method for instance), but their efficient implementation required far more subiterations. Despite this more sophisticated updates gave rise to fewer master iterations, the total number of iteration, (master and subiterations) was much larger than those reported in the examples. We note that the use of Benders decomposition, and for the same convergence tolerance, these nonlinear problems required more than 200 iterations in all cases, for example, Problem 3 needed more than 800 iterations. Furthermore, in contrast to our method, the number of iterations of the Benders implementation scaled with the problem size.

We finally highlight that although the detection of upper bounds required in general more than four iterations, the detection of lower bounds only

Problem 1 $\lambda^* = \mathbf{0.5008}$							
k	λ_{lb}^{k-1}	λ_{ub}^{k-1}	λ^k	$\frac{\lvert\lambda*-\lambda^k\rvert}{\lvert\lambda*\rvert}$	n_k		Δ^{k-1}
					SDPT3	MOSEK	
1	0	0.7071	0.3536	0.2940	2	2	0.7071
2	0.3536	0.7071	**0.5303**	0.0589	13	13	0.3536
3	0.3536	0.5303	0.4419	0.1176	2	2	0.1768
4	0.4419	0.5303	0.4861	0.0293	2	2	0.0884
5	0.4861	0.5303	**0.5082**	0.0148	5	4	0.0442
6	0.4861	0.5082	0.4972	0.0073	2	2	0.0221
7	0.4972	0.5082	**0.5027**	0.0038	10	10	0.0110
8	0.4972	0.5027	0.4999	0.0017	2	2	0.0055
9	0.4999	0.5027	**0.5013**	0.0010	5	6	0.0028
10	0.4999	0.5013	0.5006	0.0004	2	2	0.0014
	0.5006	0.5013	$\lambda^* \simeq 0.5010$	0.0003	45	45	0.0007

Table 4.3.: Numerical results of Problem 1

Problem 2 $\lambda^* = \mathbf{0.5044}$							
k	λ_{lb}^{k-1}	λ_{ub}^{k-1}	λ^k	$\frac{\lvert\lambda*-\lambda^k\rvert}{\lvert\lambda*\rvert}$	n_k		Δ^{k-1}
					SDPT3	MOSEK	
1	0	0.7071	0.3536	0.2990	2	2	0.7071
2	0.3536	0.7071	**0.5303**	0.0514	14	14	0.3536
3	0.3536	0.5303	0.4419	0.1238	2	2	0.1768
4	0.4419	0.5303	0.4861	0.0362	2	2	0.0884
5	0.4861	0.5303	**0.5082**	0.0076	6	6	0.0442
6	0.4861	0.5082	0.4972	0.0143	2	2	0.0221
7	0.4972	0.5082	0.5027	0.0033	2	2	0.0110
8	0.5027	0.5082	**0.5055**	0.0021	5	5	0.0055
9	0.5027	0.5055	0.5041	0.0006	2	2	0.0028
10	0.5041	0.5055	**0.5048**	0.0008	6	5	0.0014
	0.5041	0.5048	$\lambda^* \simeq 0.5044$	0.0001	43	42	0.0007

Table 4.4.: Numerical results of Problem 2

Problem 3							
$\lambda^* = \mathbf{0.5063}$							
k	λ_{lb}^{k-1}	λ_{ub}^{k-1}	λ^k	$\frac{\lvert\lambda* - \lambda^k\rvert}{\lvert\lambda*\rvert}$	n_k		Δ^{k-1}
					SDPT3	MOSEK	
1	0	0.7071	0.3536	0.3017	2	2	0.7071
2	0.3536	0.7071	**0.5303**	0.0475	14	15	0.3536
3	0.3536	0.5303	0.4419	0.1271	2	2	0.1768
4	0.4419	0.5303	0.4861	0.0398	2	2	0.0884
5	0.4861	0.5303	**0.5082**	0.0039	8	7	0.0442
6	0.4861	0.5082	0.4972	0.0180	2	2	0.0221
7	0.4972	0.5082	0.5027	0.0071	2	2	0.0110
8	0.5027	0.5082	0.5055	0.0016	2	2	0.0055
9	0.5055	0.5082	**0.5069**	0.0011	4	6	0.0028
10	0.5055	0.50569	0.5062	0.0002	2	2	0.0014
	0.5062	0.5069	$\lambda^* \simeq 0.5065$	0.0004	40	42	0.0007

Table 4.5.: Numerical results of Problem 3

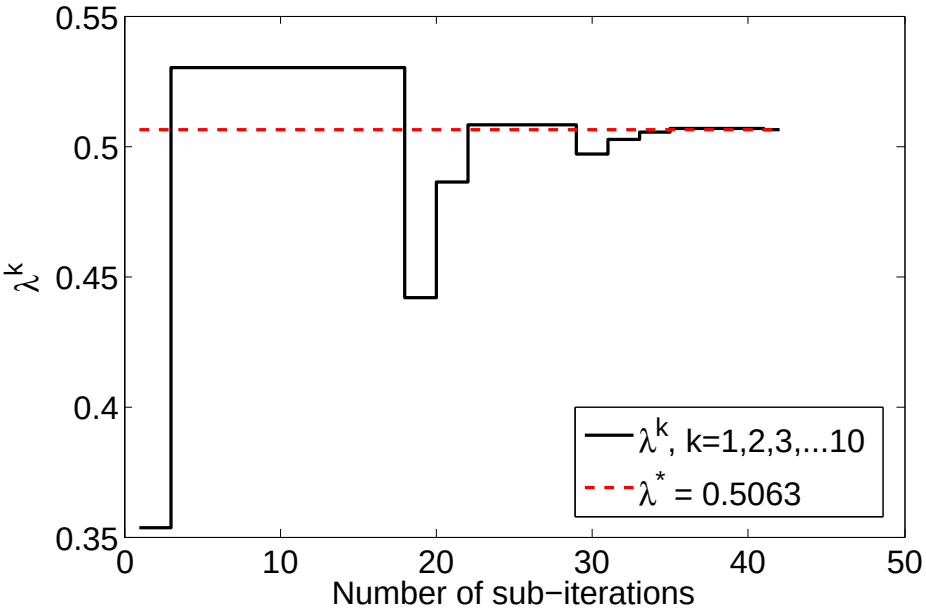
Problem 4							
$\lambda^* = \mathbf{0.5071}$							
k	λ_{lb}^{k-1}	λ_{ub}^{k-1}	λ^k	$\frac{\lvert\lambda* - \lambda^k\rvert}{\lvert\lambda*\rvert}$	n_k		Δ^{k-1}
					SDPT3	MOSEK	
1	0	0.7071	0.3536	0.3028	2	2	0.7071
2	0.3536	0.7071	**0.5303**	0.0458	14	15	0.3536
3	0.3536	0.5303	0.4419	0.1285	2	2	0.1768
4	0.4419	0.5303	0.4861	0.0414	2	2	0.0884
5	0.4861	0.5303	**0.5082**	0.0022	15	16	0.0442
6	0.4861	0.5082	0.4972	0.0196	2	2	0.0221
7	0.4972	0.5082	0.5027	0.0087	2	2	0.0110
8	0.5027	0.5082	0.5055	0.0032	2	2	0.0055
9	0.5055	0.5082	0.5069	0.0005	2	2	0.0028
10	0.5069	0.5082	**0.5075**	0.0008	4	4	0.0014
	0.5069	0.5075	$\lambda^* \simeq 0.5072$	0.0002	47	49	0.0007

Table 4.6.: Numerical results of Problem 4

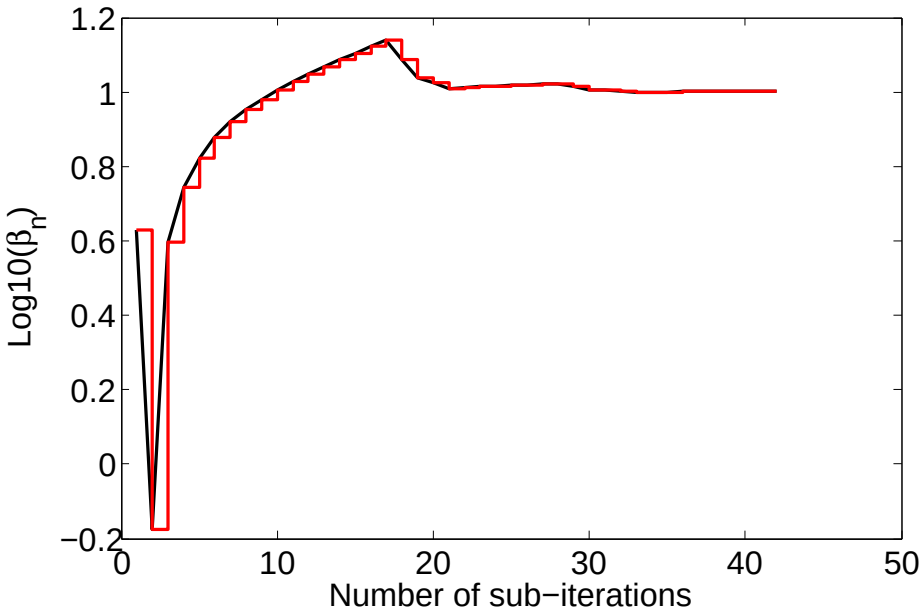
Problem 5						
$\lambda^* = \mathbf{0.5074}$						
k	λ_{lb}^{k-1}	λ_{ub}^{k-1}	λ^k	$\frac{\lvert\lambda* - \lambda^k\rvert}{\lvert\lambda*\rvert}$	n_k MOSEK	Δ^{k-1}
1	0	0.7071	0.3536	0.3032	2	0.7071
2	0.3536	0.7071	**0.5303**	0.0452	14	0.3536
3	0.3536	0.5303	0.4419	0.1290	2	0.1768
4	0.4419	0.5303	0.4861	0.0419	2	0.0884
5	0.4861	0.5303	**0.5082**	0.0017	16	0.0442
6	0.4861	0.5082	0.4972	0.0201	2	0.0221
7	0.4972	0.5082	0.5027	0.0092	2	0.0110
8	0.5027	0.5082	0.5055	0.0038	2	0.0055
9	0.5055	0.5082	0.5069	0.0011	2	0.0028
10	0.5069	0.5082	**0.5075**	0.0003	4	0.0014
	0.5069	0.5075	$\lambda^* \simeq 0.5072$	0.0004	48	0.0007

Table 4.7.: Numerical results of Problem 5

required two iterations in most of the case. This would justify the choice of low value of s in the update of λ^k in Box 1, step M1.



(a) Problem 3



(b) Problem 3

Figure 4.8.: (a) Evolution of λ^k for Problem 3, and (b) β_n as a functionofthe total numberof subiterations.

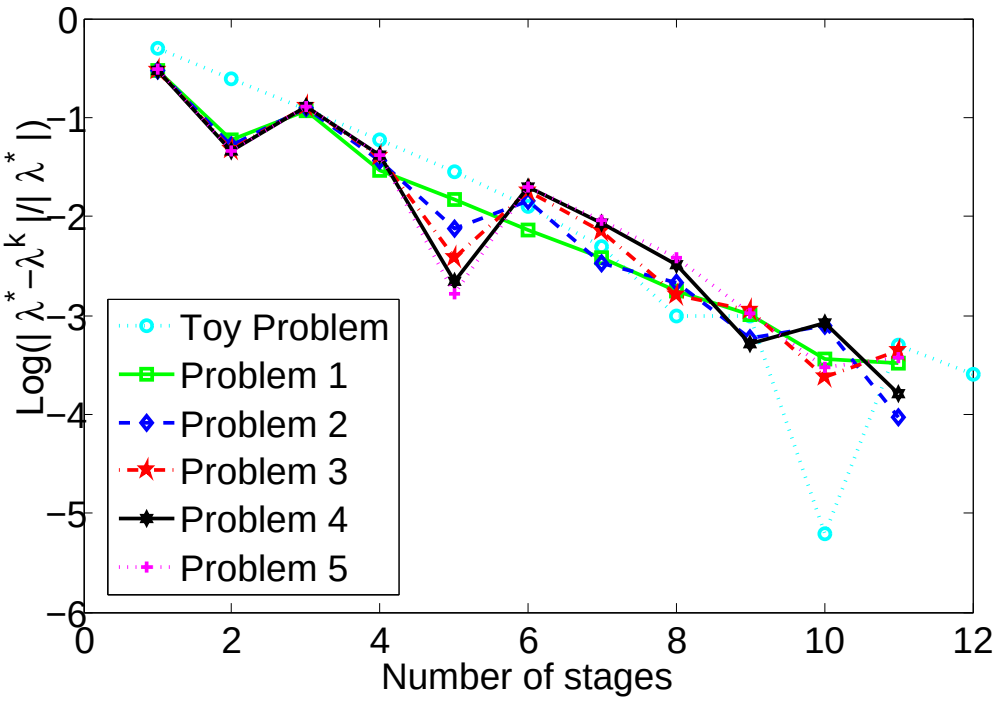


Figure 4.9.: Evolution of the relative error for each master iteration.

5

Conclusions and Future Research

5.1. Conclusions

This thesis has focused on the study of decomposition techniques for problems encountered in limit analysis. Some techniques have been introduced, such as primal, dual and Benders decomposition. We have applied and compared these techniques using a toy problem and optimisation problem in limit analysis.

We have also tested these techniques in the lower bound problem of limit analysis, with some reduced domains. The Benders decomposition can be easily adapted to limit analysis but, the number of iterations of Benders method appears to scale with the dimension of the interface.

In this thesis we have proposed a method to decompose convex nonlinear problems that contain only one complicating variable in the objective function. This type of problems includes many engineering applications in plastic structural analysis.

The method consists on interpreting the optimisation problem as the maximisation (or minimisation) of the variable subjected to a nonempty intersection set. The numerical results show that the total number of iterations does not scale with the number of variables. The extension of the method for a larger number of subproblems requires the application of the AAR method for a larger number of sets.

Working on this thesis gave rise to the flowing publications :

Articles

- N. Rabiei, J.J. Muñoz. AAR-based decomposition method for lower bound limit analysis. International Journal for Numerical Methods in Engineering (INT J NUMER METH ENG). (In preparation)

- J.J. Muñoz, N. Rabiei. Solution of lower bound limit analysis with AAR-based decomposition method. Engineering and Computational Mechanics (ICE). (Submitted)

- N. Rabiei, J.J. Muñoz. AAR-based decomposition algorithm for non-linear convex optimisation. Computational Optimization and Applications.(Springer). (Submitted)

Book Chapters

- J.J. Muñoz, N. Rabiei, A. Lyamin and A. Huerta. Computation of bounds for anchor problems in limit analysis and decomposition techniques. Book title: Direct Methods for Limit States in Structures and Materials. Spiliopoulos, Konstantinos; Weichert, Dieter (Eds). Springer , 2014.

Presentations

- N. Rabiei, J.J. Muñoz. AAR-based decomposition method for limit analysis. 11th World Congress on Computational Mechanics (WCCM XI), Barcelona, Spain, July 2014.

- N. Rabiei, J.J. Muñoz. AAR-based decomposition algorithm for non-linear convex optimisation. 20th Conference of the international federational of operational research societies (IFORS), Barselona,Spain,13th-18th july 2014.

- N. Rabiei, A. Huerta and J.J. Muñoz. Decomposition techniques in computational limit analysis. (COMPLAS XII), 3-5 Sept 2013, Barcelona, Spain.

- N. Rabiei,J.J. Muñoz. Decomposition techniques for computational limit analysis. Conference on numerical methods in engineering (CNM), Bilbao,spain,Jun.2013.

- J.J. Muñoz, N. Rabie. A. Lyamin, A. Huerta. Computation of bounds for anchor problems in limit analysis and decomposition techniques.

Third International workshop on Direct Methods . Feb. 2012, Athens, Greece.

- N. Rabiei, J.J. Muñoz. Decomposition techniques for computational limit analysis. 12th conference on Numerical Methods in Applied Mathematics, Science and Engineering (NMASE), Barcelona, Spain, 2013.

- N. Rabiei, J.J. Muñoz. Computation of bounds for anchor problems in limit analysis and decomposition techniques. 11th conference on Numerical Methods in Applied Mathematics, Science and Engineering (NMASE), Barcelona, Spain, 2012

5.2. Future Work

We next discuss some open ideas for future research derived form the work performed:

Extension the algorithm to three dimensions.

In order to apply the AAR-based decomposition technique in real three-dimensional cases, we propose the following stages:

- Implementation of the AAR-based decomposition technique to an existent three-dimensional lower problem.
- Apply the technique to simple three-dimensional cases such as the plane vertical cut.
- Consider more realistic and complex three-dimensional problems already analysed in the literature [21, 22].

We note that the extension of the algorithm does not require any substantial modifications, since the structure of the optimisation problem remains unaltered.

Finding more robust stopping criteria.

According to Proposition 4.3, when the distance between λ_{lb} and λ_{ub} is small, depending on the tolerances employed, the method may give us wrong lower bounds. In other words, it is possible that the method considers an upper bound λ^k as a lower bound (false lower bound candidate for some values of the parameters). For this reason, we aim

to obtaining robust stopping criteria, which will result in an improved accuracy.

Obtaining optimal dual variables for global problem.

In order to apply the adaptive remeshing strategy described in [16, 37], the stress and velocity fields $(\boldsymbol{\sigma}, \boldsymbol{u})$ are both required, which correspond to the primal and dual variables, respectively. Thus using AAR-based decomposition method, could not shed information about optimal dual variables which correspond to velocity field. Possible solutions to this is to consider the primal solution in order to build a reduced global problem, which does not contain the inactive constraints, or to employ the AAR-based technique presented here, but updated to the dual problem.

Early detection of upper bound λ_{ub}

From the trends of the convergence plots in Figure 4.8, it seems desirable to either reduce the number of iterations when $\lambda^* < \lambda^k$, or to approach λ^* from below, that is, to obtain as many lower bounds as possible. This is equivalent to using small values of s in the update process of λ. Further study of optimal value of s are therefore necessary.

A

Deduction of Lower Bound Discrete Problem

A.1. Equilibrium Constraint

Since we are in a two-dimensional problem, in each point of the domain the stress tensor is defined by 3 components: $\boldsymbol{\sigma}_{11}, \boldsymbol{\sigma}_{22}, \boldsymbol{\sigma}_{12}$. to simplify future expressions,we will use the following convention $\boldsymbol{\sigma}_1 = \boldsymbol{\sigma}_{11}, \boldsymbol{\sigma}_2 = \boldsymbol{\sigma}_{22}, \boldsymbol{\sigma}_3 = \boldsymbol{\sigma}_{12}$. within each triangle Ω^e, a local expression of the interpolation X^{LB} is given by:

$$\boldsymbol{\sigma}^e = \sum_{i=1}^{3} \boldsymbol{\sigma}^{i,e} N_i^e, \; \forall\, e = 1, \cdots, ne \tag{A.1.1}$$

with ne the number of elements, and

$$\boldsymbol{\sigma}^{i,e} = \begin{bmatrix} \sigma_1^{i,e} & \sigma_3^{i,e} \\ \sigma_3^{i,e} & \sigma_2^{i,e} \end{bmatrix}, \quad \boldsymbol{\sigma}^e = \begin{bmatrix} \sigma_1^e & \sigma_3^e \\ \sigma_3^e & \sigma_2^e \end{bmatrix} \qquad \forall\, e = 1, \cdots, ne\,, (i = 1, 2, 3).$$

Henceforth we will use the alternative form of the stress tensors, i.e. $\boldsymbol{\sigma}^{i,e} = (\sigma_1^{i,e}, \sigma_2^{i,e}, \sigma_3^{i,e})^T$. The equation $\nabla \cdot \boldsymbol{\sigma}^e + \lambda \boldsymbol{f}^e = 0$ results in the following equation:

$$\begin{Bmatrix} \frac{\partial \boldsymbol{\sigma}_1^e(X)}{\partial x_1} + \frac{\partial \boldsymbol{\sigma}_3^e(X)}{\partial x_2} \\ \frac{\partial \boldsymbol{\sigma}_3^e(X)}{\partial x_1} + \frac{\partial \boldsymbol{\sigma}_2^e(X)}{\partial x_2} \end{Bmatrix} + \lambda \begin{Bmatrix} f_1^e \\ f_2^e \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}. \tag{A.1.2}$$

Using the linear interpolation in (A.1.1), we obtain:

$$\begin{Bmatrix} \sum_{a=1}^{3} (\boldsymbol{\sigma}_1^{a,e} \frac{\partial N_a^e(X)}{\partial x_1} + \sigma_3^{a,e} \frac{\partial N_a^e(X)}{\partial x_2}) \\ \sum_{a=1}^{3} (\sigma_3^{a,e} \frac{\partial N_a^e(X)}{\partial x_1} + \sigma_2^{a,e} \frac{\partial N_a^e(X)}{\partial x_2}) \end{Bmatrix} + \lambda \begin{Bmatrix} f_1^e \\ f_2^e \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix}. \tag{A.1.3}$$

In order to simplify this expression, we use the following notation $N^a_{a,i} = \frac{\partial N^e_a(X)}{\partial x_i}, i = 1, 2$. Note that, since the shape functions are linear, their derivatives are constant on each element. In matrix form, and grouping all the nodal stress components in a single vector, the equation in (A.1.3) reads:

$$\begin{bmatrix} N^e_{1,1} & 0 & N^e_{1,2} & N^e_{2,1} & 0 & N^e_{2,2} & N^e_{3,1} & 0 & N^e_{3,2} \\ 0 & N^e_{1,2} & N^e_{1,1} & 0 & N^e_{2,2} & N^e_{2,1} & 0 & N^e_{3,2} & N^e_{3,1} \end{bmatrix} \begin{Bmatrix} \boldsymbol{\sigma}^{1,e} \\ \boldsymbol{\sigma}^{2,e} \\ \boldsymbol{\sigma}^{3,e} \end{Bmatrix} + \lambda \begin{Bmatrix} f^e_1 \\ f^e_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix},$$

or equivalently,

$$\mathbf{B}^e \boldsymbol{\sigma}^e + \lambda \boldsymbol{F}^{eq1,e},$$

with $\boldsymbol{\sigma}^e = (\boldsymbol{\sigma}^{1,e}, \boldsymbol{\sigma}^{2,e}, \boldsymbol{\sigma}^{3,e})^T$.

Let $\boldsymbol{\sigma}$ denote the vector collecting the nodal stress components for all the elements in the mesh and $\boldsymbol{F}^{eq1}$ be a global volume force vector. The vector $\boldsymbol{\sigma}$ has $9 \times ne$ dimensions and $\boldsymbol{F}^{eq1}$ is a $2 \times ne$ dimensional vector. Likewise, we construct a global matrix $\mathbf{A}^{eq1}$, with dimensions $(2 \times ne, 9 \times ne)$, that consists of the elemental matrices $\mathbf{B}^e$. The assembly process is straightforward since the equation for the elements are uncoupled. Consequently, $\mathbf{A}^{eq1}$ results in a very sparse block diagonal matrix, where 0 is a zero matrix of dimensions $(2, 9)$.

$$\mathbf{A}^{eq1} = \begin{bmatrix} \mathbf{B}^1 & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^2 & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \mathbf{0} & \cdots & \cdots & \cdots & \mathbf{B}^{ne} \end{bmatrix} \tag{A.1.4}$$

Then the first constraint in equation (1.2.4) is given by:

$$\mathbf{A}^{eq1} \boldsymbol{\sigma} + \mathbf{F}^{eq1} \lambda = \mathbf{0}. \tag{A.1.5}$$

A.2. Inter-element Equilibrium Constraints

In this section we address the practical implementation of equation $(\boldsymbol{\sigma}^e - \boldsymbol{\sigma}^{e'}) \cdot \boldsymbol{n}^{\xi^{e'}_e} = \mathbf{0}, \ \forall \xi^{e'}_e \in \xi^0$. Note that the equations must hold for all the

points in the edge ξ under consideration. The stress at each side of the the edge is given by,

$$\begin{aligned} \boldsymbol{\sigma}^{e} &= \boldsymbol{\sigma}^{e,1} N_1^e + \boldsymbol{\sigma}^{e,2} N_2^e, \\ \boldsymbol{\sigma}^{e'} &= \boldsymbol{\sigma}^{e',3} N_1^{e'} + \boldsymbol{\sigma}^{e',4} N_2^{e'}. \end{aligned} \tag{A.2.1}$$

Inserting this interpolation into the equation $(\boldsymbol{\sigma}^e - \boldsymbol{\sigma}^{e'}) \cdot \boldsymbol{n}^{\xi_e^{e'}} = \mathbf{0}$ for a given $\xi_e^{e'} \in \xi^O$, we obtain the following two scalar equations:

$$[(\sigma_1^{e,1} - \sigma_1^{e',3})n_1 + (\sigma_3^{e,1} - \sigma_3^{e',3})n_2]N_1^e + [(\sigma_1^{e,2} - \sigma_1^{e',4})n_1 + (\sigma_3^{e,2} - \sigma_3^{e',4})n_2]N_2^e = 0$$
$$[(\sigma_3^{e,1} - \sigma_3^{e',3})n_1 + (\sigma_2^{e,1} - \sigma_2^{e',3})n_2]N_1^{e'} + [(\sigma_3^{e,2} - \sigma_3^{e',4})n_1 + (\sigma_2^{e,2} - \sigma_2^{e',4})n_2]N_2^{e'} = 0$$

Clearly, a sufficient condition for the above equation to hold over all the points in the edge $\xi_e^{e'}$ is to nullify the 4 coefficients of N_i^e and $N_i^{e'}$:

$$\begin{aligned} (\sigma_1^{e,1} - \sigma_1^{e',3})n_1 + (\sigma_3^{e,1} - \sigma_3^{e',3})n_2 = 0, \\ (\sigma_1^{e,2} - \sigma_1^{e',4})n_1 + (\sigma_3^{e,2} - \sigma_3^{e',4})n_2 = 0, \\ (\sigma_3^{e,1} - \sigma_3^{e',3})n_1 + (\sigma_2^{e,1} - \sigma_2^{e',3})n_2 = 0, \\ (\sigma_3^{e,2} - \sigma_3^{e',4})n_1 + (\sigma_2^{e,2} - \sigma_2^{e',4})n_2 = 0, \end{aligned}$$

or in the matrix form:

$$\begin{bmatrix} n_1 & 0 & n_2 & 0 & 0 & 0 & -n_1 & 0 & -n_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & n_1 & 0 & n_2 & 0 & 0 & 0 & -n_1 & 0 & -n_2 \\ 0 & n_2 & n_1 & 0 & 0 & 0 & 0 & -n_2 & -n_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & n_2 & n_1 & 0 & 0 & 0 & 0 & -n_2 & -n_1 \end{bmatrix} \begin{Bmatrix} \boldsymbol{\sigma}^{e,1} \\ \boldsymbol{\sigma}^{e,2} \\ \boldsymbol{\sigma}^{e',3} \\ \boldsymbol{\sigma}^{e',4} \end{Bmatrix} = \mathbf{0},$$

If we permute row 2 and row 3 and as well we define

$$\mathbf{N} = \begin{bmatrix} n_1 & 0 & n_2 \\ 0 & n_2 & n_1 \end{bmatrix}, \quad \mathbf{B}_e^{e'} = \begin{bmatrix} \mathbf{N} & \mathbf{0} \\ \mathbf{0} & \mathbf{N} \end{bmatrix}, \tag{A.2.2}$$

then we have

$$\begin{aligned} &\begin{bmatrix} \mathbf{N} & \mathbf{0} & -\mathbf{N} & \mathbf{0} \\ \mathbf{0} & \mathbf{N} & \mathbf{0} & -\mathbf{N} \end{bmatrix} \begin{Bmatrix} \boldsymbol{\sigma}^{e,1} \\ \boldsymbol{\sigma}^{e,2} \\ \boldsymbol{\sigma}^{e',3} \\ \boldsymbol{\sigma}^{e',4} \end{Bmatrix} = \mathbf{0}, \\ &\mathbf{B}_e^{e'} \begin{Bmatrix} \boldsymbol{\sigma}^{e,1} \\ \boldsymbol{\sigma}^{e,2} \end{Bmatrix} - \mathbf{B}_e^{e'} \begin{Bmatrix} \boldsymbol{\sigma}^{e',3} \\ \boldsymbol{\sigma}^{e',4} \end{Bmatrix} = \mathbf{0} \qquad \forall \xi_e^{e'} \in \xi^O. \end{aligned} \tag{A.2.3}$$

Therefore the second constraint in equation (1.2.4) is given by

$$\mathbf{A}^{eq2}\boldsymbol{\sigma} + \mathbf{0}\lambda = \mathbf{0}. \tag{A.2.4}$$

A.3. Boundary Element Equilibrium Constraints

Exactly the same procedure is followed for the third equation $\boldsymbol{\sigma}^e \cdot \boldsymbol{n}^{\xi_e^N} = \lambda \boldsymbol{g}^{\xi_e^N}$, $\forall \xi_e^N \in \xi^N$, which after using discretisation in (A.2.1) reads

$$\begin{aligned}(\sigma_1^{e,1} n_1 + \sigma_3^{e,1} n_2) N_1^e + (\sigma_1^{e,2} n_1 + \sigma_3^{e,2} n_2) N_2^e = \lambda g_1^e,\\ (\sigma_3^{e,1} n_1 + \sigma_2^{e,1} n_2) N_1^e + (\sigma_3^{e,2} n_1 + \sigma_2^{e,2} n_2) N_2^e = \lambda g_2^e.\end{aligned}$$

Now, in order to guarantee the satisfaction of the equations above at all points of the edge, it is sufficient to force the four coefficients to be equal to $\lambda \boldsymbol{g}^e$. That is:

$$\begin{aligned}\sigma_1^{e,1} n_1 + \sigma_3^{e,1} = \lambda g_1^e,\\ \sigma_1^{e,2} n_1 + \sigma_3^{e,2} = \lambda g_1^e,\\ \sigma_3^{e,1} n_1 + \sigma_2^{e,1} = \lambda g_2^e,\\ \sigma_3^{e,2} n_1 + \sigma_2^{e,2} = \lambda g_2^e.\end{aligned}$$

In matrix form, and using the previous definitions of $\mathbf{N}$ and $\mathbf{B}$ in (A.2.2),

$$\begin{bmatrix}\mathbf{N} & \mathbf{0}\\ \mathbf{0} & \mathbf{N}\end{bmatrix}\begin{Bmatrix}\boldsymbol{\sigma}^{e,1}\\ \boldsymbol{\sigma}^{e,2}\end{Bmatrix} + \lambda \begin{Bmatrix}-g_1^e\\ -g_2^e\\ -g_1^e\\ -g_2^e\end{Bmatrix} = \mathbf{0},$$

$$\mathbf{B}_e \begin{Bmatrix}\boldsymbol{\sigma}^{e,1}\\ \boldsymbol{\sigma}^{e,2}\end{Bmatrix} + \lambda \begin{Bmatrix}-g_1^e\\ -g_2^e\\ -g_1^e\\ -g_2^e\end{Bmatrix} = \mathbf{0},\ \forall\, \xi_e^N \in \xi^N.$$

Consequently the third equilibrium constraint in equation (1.2.4) may be expressed as,

$$\mathbf{A}^{eq3}\boldsymbol{\sigma} + \lambda \boldsymbol{F}^{eq3} = \mathbf{0}. \tag{A.3.1}$$

A.4. Membership Constraints

In this work we consider von Mises type in plane strain, which is given by the following membership constraint:

$$(\sigma_1 - \sigma_2)^2 + 4\sigma_3^2 \leq \frac{4}{3}\sigma_y^2,$$

where σ_y is a material parameter. We use this constraint for each node, i.e.

$$\boldsymbol{\sigma}^{e,i,LB} \in \mathcal{B} \text{ in } \Omega^e, \quad (\forall\, e = 1, \cdots, ne,\ i = 1, 2, 3),$$

with $\mathcal{B} = \{\boldsymbol{\sigma} | (\sigma_2 - \sigma_2)^2 + 4\sigma_3^2 \leq \frac{4}{3}\sigma_y^2\}$.

On the whole we have $3 \times ne$ inequalities. A convenient way to impose the inequality above is to force the vector $(\frac{2}{\sqrt{3}}\sigma_y, 2\sigma_3^{e,i}, \sigma_1^{e,i} - \sigma_2^{e,i})$ to belong to the Lorentz cone L^n with $n = 3$. Since second-order cone constraints are directly imposed through the decision variables, we can introduce a vector $\boldsymbol{x}^{e,i}$ of additional variables as follows:

$$\begin{cases} x_1^{e,i} = \dfrac{2}{\sqrt{3}}\sigma_y \\ -2\sigma_3^{e,i} + x_2^{e,i} = 0 \\ -\sigma_1^{e,i} + \sigma_2^{e,i} + x_3^{e,i} = 0 \end{cases} \tag{A.4.1}$$

We will force each vector $\boldsymbol{x}^{e,i}$ to belong to L^n such that L^n is:

$$L^n = \left\{ x \in \Re^n | x_1 \geq \sqrt{\sum_{j=2}^{n} x_j^2} \right\}.$$

The imposition of (A.4.1) over all the mesh requires $9 \times ne$ equations. In matrix notation, for each node we have:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -2 \\ -1 & 1 & 0 \end{bmatrix} \begin{Bmatrix} \sigma_1^{e,i} \\ \sigma_2^{e,i} \\ \sigma_3^{e,i} \end{Bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x_1^{e,i} \\ x_2^{e,i} \\ x_3^{e,i} \end{Bmatrix} = \begin{Bmatrix} \sqrt{2}\sigma_y \\ 0 \\ 0 \end{Bmatrix},$$

then the global system can be written as follows:

$$\mathbf{A}^{soc}\boldsymbol{\sigma} + \mathbf{I}\boldsymbol{x}_{1:3} = \boldsymbol{b}^{soc},$$

where $\boldsymbol{\sigma}$ is the usual vector of known nodal stresses, $\mathbf{I}$ is a $(9 \times ne, 9 \times ne)$ identity matrix, $\boldsymbol{x}_{1:3}$ is a vector of $9 \times ne$ additional variables ordered in the same way as $\boldsymbol{\sigma}$. The $9 \times ne$ dimensional vector $\boldsymbol{b}^{soc}$ and $(9 \times ne, 9 \times ne)$ block diagonal matrix $\mathbf{A}^{soc}$ have the following forms:

$$\boldsymbol{b}^{soc} = \begin{Bmatrix} b \\ b \\ \vdots \\ \vdots \\ b \end{Bmatrix}, \ \mathbf{A}^{soc} = \begin{bmatrix} \mathbf{M} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{M} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \mathbf{0} & \cdots & \cdots & \cdots & \mathbf{M} \end{bmatrix},$$

where

$$\boldsymbol{b} = \begin{Bmatrix} \sqrt{2}\sigma_y \\ 0 \\ 0 \end{Bmatrix}, \ \mathbf{M} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -2 \\ -1 & 1 & 0 \end{bmatrix}. \tag{A.4.2}$$

By recasting all the previous matrix constraints in a single matrix equation, we obtain the discrete lower bound problem:

$$\lambda^{LB} = \max \lambda$$
$$\begin{cases} \begin{bmatrix} \mathbf{A}^{eq1} & \vdots & \boldsymbol{F}^{eq1} & \vdots & \mathbf{0} \\ \mathbf{A}^{eq2} & \vdots & \mathbf{0} & \vdots & \mathbf{0} \\ \mathbf{A}^{eq3} & \vdots & \boldsymbol{F}^{eq3} & \vdots & \mathbf{0} \\ \mathbf{A}^{soc} & \vdots & \mathbf{0} & \vdots & I \end{bmatrix} \begin{Bmatrix} \boldsymbol{\sigma} \\ \lambda \\ \boldsymbol{x}_{1:3} \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{b}^{soc} \end{Bmatrix}, \\ \boldsymbol{\sigma} \text{ free } \lambda \geq 0, \ \boldsymbol{x}_{1:3} \in K \end{cases} \tag{A.4.3}$$

where $K = L^n \times \cdots \times L^n$. The solution of (A.4.3) corresponds to the desired lower bound λ^{*LB}. Note that, this problem is a conic program and has the standard form required by most optimisation packages.

The global lower bound matrix defined in (A.4.3) has dimensions $2ne + 4|\xi^0| + 4|\xi^N| + 9ne; 9ne + 1 + 9ne$, and involves the vector of nodal stresses $\boldsymbol{\sigma}$, the collapse multiplier λ and the vector of additional variables $\boldsymbol{x}^{soc}$. While it was natural and easy to build the linear system of equation using those variable, they are not optimal for solving the lower bound problem since they lead to matrices that are much larger than strictly required. This increases unnecessarily the computational time and memory requirements involved in the solution process. with the purpose of optimisation the computational

cost od solving the problem, we have introduced a change of variables that enables us to reduce substantially the number of equations and variables involved. Note that $\boldsymbol{\sigma}$ in (A.4.3) is a free vector, that is, nothing is imposed directly on the nodal stresses, which are not required to belong to any particular cone. On the other hand, the imposition of the yield condition requires that, for each node, some affine combinations of the stresses belongs to Lorentz cone L^n. This is the reason why we introduced the additional variables $\boldsymbol{x}^{soc}$. So we get rid of the nodal stress variables $\boldsymbol{\sigma}$ by directly formulating the equilibrium equations in terms of the additional variables $\boldsymbol{x}^{soc}$. For plane strain our goal can be achieved as follows:

$$\begin{cases} x_1^{e,i} = \dfrac{2}{\sqrt{3}} \\ -2\sigma_3^{e,i} + x_2^{e,i} = 0 \\ -\sigma_1^{e,i} + \sigma_2^{e,i} + x_3^{e,i} = 0 \\ -\sigma_1^{e,i} + x_4^{e,i} = 0 \\ \forall\ e = 1, \cdots, ne,\ i = 1, 2, 3 \end{cases} \tag{A.4.4}$$

Considering (A.4.4), we can write an equivalent expression for plane strain:

$$\begin{aligned} &\begin{Bmatrix} \sigma_1^{e,i} \\ \sigma_2^{e,i} \\ \sigma_3^{e,i} \end{Bmatrix} = \begin{Bmatrix} 1 \\ 1 \\ 0 \end{Bmatrix} x_4^{e,i} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & \frac{1}{2} & 0 \end{bmatrix} \begin{Bmatrix} x_1^{e,i} \\ x_2^{e,i} \\ x_3^{e,i} \end{Bmatrix} \\ &\boldsymbol{\sigma}^{e,i} = \boldsymbol{p} x_4^{e,i} + \mathbf{Q} \boldsymbol{x}_{1:3}^{e,i}, \end{aligned} \tag{A.4.5}$$

and write $\boldsymbol{\sigma} = \mathbf{P}\boldsymbol{x}_4 + \mathbf{Q}\boldsymbol{x}_{1:3}$ where:

$$\mathbf{P} = \begin{bmatrix} \boldsymbol{p} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{p} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \mathbf{0} & \cdots & \cdots & \cdots & \boldsymbol{p} \end{bmatrix}, \ \boldsymbol{x}_4 = (\boldsymbol{x}_4^{e,i})$$

$(e = 1, \cdots, ne; i = 1, 2, 3)$

$$\mathbf{Q} = \begin{bmatrix} \boldsymbol{Q} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{Q} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \mathbf{0} & \cdots & \cdots & \cdots & \boldsymbol{Q} \end{bmatrix}, \ \boldsymbol{x}_{1:3} = (\boldsymbol{x}_{1:3}^{e,i}), \ \boldsymbol{x}_{1:3}^{e,i} = \begin{Bmatrix} x_1^{e,i} \\ x_2^{e,i} \\ x_3^{e,i} \end{Bmatrix},$$

$(e = 1, \cdots, ne; \ i = 1, 2, 3).$

Using the matrix notation, the equilibrium constraints can be written as:

$$\begin{aligned} &\mathbf{A}^{eq1}\mathbf{P}\boldsymbol{x}_4 + \mathbf{A}^{eq1}\mathbf{Q}\boldsymbol{x}_{1:3} + \lambda \boldsymbol{F}^{eq1} = \mathbf{0}, \\ &\mathbf{A}^{eq2}\mathbf{P}\boldsymbol{x}_4 + \mathbf{A}^{eq2}\mathbf{Q}\boldsymbol{x}_{1:3} = \mathbf{0}, \\ &\mathbf{A}^{eq3}\mathbf{P}\boldsymbol{x}_4 + A^{eq3}\mathbf{Q}\boldsymbol{x}_{1:3} + \lambda \boldsymbol{F}^{3} = \mathbf{0}. \end{aligned} \tag{A.4.6}$$

To impose the membership constraints in the three nodes of all the elements, it is still necessary to add the remaining nodal equation in (A.4.4) for plane strain. This is accomplished as follows:

$$\begin{aligned} &x_1^{e,i} = \frac{2}{\sqrt{3}}\sigma_y \implies \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{Bmatrix} x_1^{e,i} \\ x_2^{e,i} \\ x_3^{e,i} \end{Bmatrix} = \frac{2}{\sqrt{3}}, \\ &\mathbf{R}x_{1:3}^{e,i} = \frac{2}{\sqrt{3}}, \text{ where } \boldsymbol{R} = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix}, \ \boldsymbol{x}_{1:3}^{e,i} = \begin{Bmatrix} x_1^{e,i} \\ x_2^{e,i} \\ x_3^{e,i} \end{Bmatrix}, \end{aligned} \tag{A.4.7}$$

which results in the following global matrix equation:

$$\mathbf{R}\boldsymbol{x}_{1:3} = \boldsymbol{b},$$

where

$$\mathbf{R} = \begin{bmatrix} \mathbf{R} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{R} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \mathbf{0} & \cdots & \cdots & \cdots & \mathbf{R} \end{bmatrix}, \ \boldsymbol{x}_{1:3} = \left\{ \boldsymbol{x}_{1:3}^{e,i} \right\}, \ \boldsymbol{b} = \begin{Bmatrix} \frac{2}{\sqrt{3}} \\ \frac{2}{\sqrt{3}} \\ \vdots \\ \vdots \\ \frac{2}{\sqrt{3}} \end{Bmatrix}. \tag{A.4.8}$$

Finally, the transformed version of (A.4.3) results in the following optimisation problem, which is the one we will actually solve. For plane strain:

$$\begin{aligned} &\lambda^{LB} = \max \lambda \\ &\quad \begin{bmatrix} \mathbf{A}^{eq1}\mathbf{P} & \boldsymbol{F}^{1} & \mathbf{A}^{eq1}\mathbf{P} \\ \mathbf{A}^{eq2}\mathbf{P} & \mathbf{0} & \mathbf{A}^{eq2}\mathbf{P} \\ \mathbf{A}^{eq3}\mathbf{P} & \boldsymbol{F}^{3} & \mathbf{A}^{eq3}\mathbf{P} \\ \mathbf{0} & \mathbf{0} & \mathbf{R} \end{bmatrix} \begin{Bmatrix} \boldsymbol{x}_4 \\ \lambda \\ \boldsymbol{x}_{1:3} \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \\ \boldsymbol{b} \end{Bmatrix} \\ &\quad \boldsymbol{x}_4 \text{ is free}, \ \lambda \geq 0, \ \boldsymbol{x}_{1:3} \in K \\ &\quad |\boldsymbol{x}_4| = 3\text{ne}, \ |\lambda| = 1, \ |\boldsymbol{x}_{1:3}| = 9\text{ne}. \end{aligned} \tag{A.4.9}$$

As the reader will observe, the size of problem above depends on number of elements and dimension of space. By increasing the number of elements or dimension of space, the size of the problem to become greater.

B

Background on Convex Sets

B.1. Sets in $\Re^n$

Throughout this thesis, $\Re^n$ is a real linear space with scalar (or inner) product. The associated norm is denoted by $\|.\|$ and the associated distance by d, i.e.,

$$(\forall \boldsymbol{x} \in \Re^n)(\forall \boldsymbol{y} \in \Re^n) \quad \|\boldsymbol{x}\| = (\boldsymbol{x} \cdot \boldsymbol{x})^{\frac{1}{2}} \quad \text{and} \quad d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\|. \quad \text{(B.1.1)}$$

The identity operator on $\Re^n$ is denoted by I.

Let C and D be subsets of $\Re^n$, and let $\boldsymbol{z} \in \Re^n$. Then $C+D = \{\boldsymbol{x}+\boldsymbol{y} | \boldsymbol{x} \in \Re^n, \boldsymbol{y} \in \Re^n\}$, $\boldsymbol{z} + C = \{\boldsymbol{z}\} + C$, $C - \boldsymbol{z} = C - \{\boldsymbol{z}\}$, and, for every $\lambda \in \Re$, $\lambda C = \{\lambda \boldsymbol{x} | \boldsymbol{x} \in C\}$. If Λ is a nonempty subset of $\Re$, then $\Lambda C = \bigcup_{\lambda \in \Lambda} \lambda C$ and $\Lambda \boldsymbol{z} = \Lambda\{\boldsymbol{z}\} = \{\lambda \boldsymbol{z} | \lambda \in \Lambda\}$. In particular, C is a affine subspace if

$$C \neq \emptyset \quad \text{and} \quad (\forall \lambda \in \Re), \quad C = \lambda C + (1 - \lambda) C. \quad \text{(B.1.2)}$$

The four types of line segments between two points $\boldsymbol{x}$ and $\boldsymbol{y}$ in $\Re^n$ are

$$\begin{aligned} [\boldsymbol{x}, \boldsymbol{y}] &= \{(1-\alpha)\boldsymbol{x} + \alpha \boldsymbol{y} | 0 \leq \alpha \leq 1\}, \\ (\boldsymbol{x}, \boldsymbol{y}] &= \{(1-\alpha)\boldsymbol{x} + \alpha \boldsymbol{y} | 0 < \alpha \leq 1\}, \\ [\boldsymbol{x}, \boldsymbol{y}) &= \{(1-\alpha)\boldsymbol{x} + \alpha \boldsymbol{y} | 0 \leq \alpha < 1\}, \\ (\boldsymbol{x}, \boldsymbol{y}) &= \{(1-\alpha)\boldsymbol{x} + \alpha \boldsymbol{y} | 0 < \alpha < 1\}, \end{aligned} \quad \text{(B.1.3)}$$

and $(\boldsymbol{x}, \boldsymbol{y}] = [\boldsymbol{y}, \boldsymbol{x})$.

B.2. The Extended Real Line

One obtains the extended real line $[-\infty, +\infty] = \Re \cup \{-\infty\} \cup \{+\infty\}$ by adjoining the elements $-\infty$ and $+\infty$ to the real line $\Re$ and extending the order via $(\forall\, \xi \in \Re)\ -\infty < \xi < +\infty$. Given $\xi \in \Re$, $(\xi, +\infty] = (\xi, +\infty) \cup \{+\infty\}$; the other extended intervals are defined similarly.

Throughout this thesis, for extended real numbers, positive means ≥ 0, strictly positive means > 0, negative means ≤ 0, and strictly negative means < 0. Moreover $\Re_+ = [0, +\infty) = \{\xi \in \Re | \xi \geq 0\}$ and $\Re_{++} = (0, +\infty) = \{\xi \in \Re | \xi > 0\}$. Likewise, if n is a strictly positive integer, the positive orthant is $\Re^n_+ = [0, +\infty)^n$ and the strictly positive orthant is $\Re^n_{++} = (0, +\infty)^n$. The sets $\Re_-$ and $\Re_{--}$, as well as the negative orthants, $\Re^n_-$ and $\Re^n_{--}$, are defined similarly.

B.3. Convex Sets and Cones

Definition B.1 *A subset C of $\Re^n$ is convex if $\forall \alpha \in (0,1)$, $\alpha C + (1-\alpha)C = C$ or, equivalently, if*

$$(\forall\, \boldsymbol{x} \in C)\,(\forall\, \boldsymbol{y} \in C) \quad (\boldsymbol{x}, \boldsymbol{y}) \in C, \tag{B.3.1}$$

where $(\boldsymbol{x}, \boldsymbol{y})$ is the line segments between two points $\boldsymbol{x}$ and $\boldsymbol{y}$. In particular, $\Re^n$ and $\emptyset$ are convex.

Example B.1 *In each of the following cases, C is a convex subset of $\Re^n$.*

(i) *C is a ball.*

(ii) *C is an affine subspace.*

(iii) *$C = \bigcap_{j \in J} C_j$, where $(C_j)_{j \in J}$ is a family of convex subsets of $\Re^n$.*

Definition B.2 *A subset K of $\Re^n$ is called a cone if it is closed under strictly positive scalar multiplication, i.e.*

$$K = \Re_{++} K. \tag{B.3.2}$$

Hence, $\Re^n$ is a cone and the intersection of family of cones is a cone. A convex cone is a cone which is a convex set.

Fact B.1 *Let $K \subset \Re^n$, $L \subset \Re^m$ be closed convex cones. Then*

$$K \oplus L := \{(\boldsymbol{x}, \boldsymbol{y}) \in \Re^n \oplus \Re^m | \boldsymbol{x} \in K, \boldsymbol{y} \in L\},$$

is again a closed convex cone, the direct sum of K and L.

Remark B.1 *If not stated otherwise,$(\boldsymbol{x}, \boldsymbol{y})$ denotes the direct sum of spaces, and not the segment.*

Let us remind the reader that $\Re^n \oplus \Re^m$ is the set $\Re^n \times \Re^m$, turned into a Hilbert space via

$$\begin{aligned}
&(\boldsymbol{x}, \boldsymbol{y}) + (\acute{\boldsymbol{x}}, \acute{\boldsymbol{y}}) := (\boldsymbol{x} + \acute{\boldsymbol{x}}, \boldsymbol{y} + \acute{\boldsymbol{y}}),\\
&\lambda(\boldsymbol{x}, \boldsymbol{y}) := (\lambda\boldsymbol{x}, \lambda\boldsymbol{y}),\\
&(\boldsymbol{x}, \boldsymbol{y}) \cdot (\acute{\boldsymbol{x}}, \acute{\boldsymbol{y}}) := \boldsymbol{x} \cdot \acute{\boldsymbol{y}} + \boldsymbol{y} \cdot \acute{\boldsymbol{y}}.
\end{aligned}$$

Two of the most important convex cones are the positive orthant and the strictly positive orthant of $\Re^n$. These cones are useful in the theory of inequalities.

In the following we describe two concrete closed convex cone.

B.3.1. The Ice Cream Cone in $\Re^n$(The Quadratic Cone)

This cone is defined as

$$C_q = \{(x_1, \boldsymbol{x}) \in \Re \times \Re^{n-1} : \|\boldsymbol{x}\| \leq x_1\}, \tag{B.3.3}$$

and is also known as the quadratic cone, Lorenz cone, and the second-order cone. See Figure B.1 for an illustration in $\Re^3$ that (hopefully) explains the name. It is closed because of the $\leq$, and its convexity follows from the the triangle inequality $\|\boldsymbol{x} + \boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\|$.

B.3.2. The Rotated Quadratic Cone

Another important closed convex cone that appears in conic optimisation formulation is the rotated quadratic cone, which is defined as follows:

$$C_q^r = \{(x_1, x_2, \boldsymbol{x}) \in \Re \times \Re \times \Re^{n-2} | 2x_1x_2 \geq \sum_{j=3}^{n} x_j^2, x_1, x_2 \geq 0\} \tag{B.3.4}$$

Remark B.2 *$\boldsymbol{x} = (x_1, x_2, x_3, \cdots, x_n) \in C_q^r$ if and only if $\boldsymbol{y} = (y_1, y_2.y_3, \cdots, y_n) \in C_q$ where $y_1 = \frac{x_1+x_2}{\sqrt{2}}$, $y_2 = \frac{x_1-x_2}{\sqrt{2}}$, and $y_i = x_i, j = 3, \cdots, n$. In other words, the rotated quadratic cone is identical to the quadratic cone under a linear transformation.*

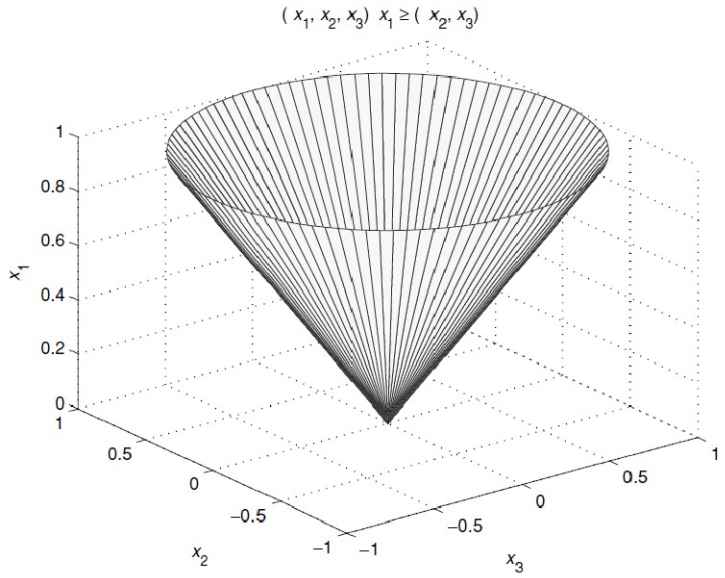


Figure B.1.: The ice cream cone [20]

B.3.3. Polar and Dual Cones

Definition B.3 *Let C be a subset of $\Re^n$. The polar cone of C is*

$$C^\circ = C^\ominus = \{\boldsymbol{u} \in \Re^n | \boldsymbol{x} \cdot \boldsymbol{u} \leq 0 \quad \forall \boldsymbol{x} \in C\}, \tag{B.3.5}$$

and the dual cone of C is $C^\oplus = -C^\ominus$, i.e.

$$C^* = C^\oplus = \{\boldsymbol{u} \in \Re^n | \boldsymbol{x} \cdot \boldsymbol{u} \geq 0 \quad \forall \boldsymbol{x} \in C\}. \tag{B.3.6}$$

If C is a nonempty convex cone, then C is self-dual if $C = C^*$. See Figure B.2 for an illustration of dual and polar cone of a set.

Let us illustrate this notion on the examples that we have seen earlier. What is the dual of the positive orthant $\Re^n_+$? This is the set of $\boldsymbol{u}$ such that

$$\boldsymbol{x} \cdot \boldsymbol{u} \geq 0 \quad \forall \boldsymbol{x} \geq 0.$$

This set certainly contains the positive orthant $\{\boldsymbol{u} \in \Re^n \, | \boldsymbol{u} \geq 0\}$ itself, but not more: Given $\boldsymbol{u} \in \Re^n$ with $u_i < 0$, we have $\boldsymbol{u} \cdot \boldsymbol{e}_i < 0$, where $\boldsymbol{e}_i$ is the i-th unit vector (a member of $\Re^n_+$), and this proves that $\boldsymbol{u}$ is not a member of the dual cone of $(\Re^n_+)^*$. It follows that the dual cone of $\Re^n_+$ is $\Re^n_+$: the positive orthant is self-dual.

For the even more trivial cones, the situation is as follows.

$$C = \{\mathbf{0}\} \Rightarrow C^* = \Re^n,$$
$$C = \Re^n \Rightarrow C^* = \{\mathbf{0}\}.$$

Proposition B.1 *The quadratic and rotated quadratic cone are self-dual i.e.*

(i) $(C_q)^* = C_q$.

(ii) $(C_q^r)^* = C_q^r$.

Proof 12 *We first will show that the quadratic code is contained in the dual cone. Since all vectors in the quadratic cone have first component nonnegative we have:*

$$\boldsymbol{z} \cdot \boldsymbol{x} \geq 0 \Leftrightarrow z_1 x_1 \geq -z_2 x_2 - \cdots - z_n x_n$$

$$\begin{aligned} -z_2 x_2 - \cdots - z_n x_n &\leq |(z_2, \cdots, z_n) \cdot (x_2, \cdots, x_n)| \\ &= \|(z_2, \cdots, z_n)\| \|(x_2, \cdots, x_n)\| |cos(\theta)| \\ &\leq z_1 x_1, \end{aligned}$$

thus $C_q \subset (C_q)^*$. *Next we show that the dual cone is contained in the quadratic cone. If* $\boldsymbol{z} \in (C_q)^*$ *then for all* $\boldsymbol{x} \in C$*:*

$$\begin{aligned} z_1 x_1 &\geq -z_2 x_2 - \cdots - z_n x_n \\ &= -(z_2, \cdots, z_n) \cdot (x_2, \cdots, x_n) cos(\theta) \\ &= -\|(z_2, \cdots, z_n)\| \|(x_2, \cdots, x_n)\| cos(\theta), \end{aligned}$$

now choose the member $\boldsymbol{x}^*$ *of* C *so that* $|x_1^*| = 1$, $sign(x_1^*) = sign(z_1)$, $\|(x_2^*, \cdots, x_n^*)\| = 1$ *and* $cos(\theta) = -1$ *then* $z_1 x_1 = |z_1|$ *and we obtain:*

$$|z_1| \geq \|(z_2, \cdots, z_n)\|,$$

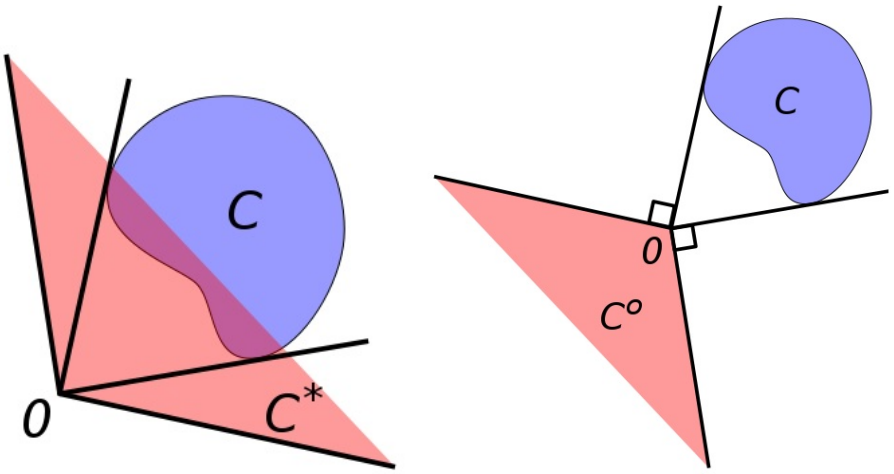
thus $\boldsymbol{z} \in C_q$ *and* $(C_q)^* \subset C_q$, *Therefore* $(C_q)^* = C_q$.

(ii). In view of Remark B.2, since the rotated quadratic cone is identical to the quadratic cone under a linear operator then is self-dual. □

We conclude this section with the following intuitive fact: the dual of a direct sum of cones is the direct sum of the dual cones. This fact is easy but not entirely trivial. It actually requires a small proof, see [27].

Proposition B.2 *Let* $K \subset \Re^n$, $L \subset \Re^m$. *Then*

$$(K \oplus L)^* = K^* \oplus L^*.$$



(a) (b)

Figure B.2.: (a) A set C and its dual cone C^*, (b) A set C and its polar cone C°

B.4. Farkas Lemma, Cone Version

Under any meaningful notion of duality, you expect the dual of the dual to be the primal (original) object. For cone duality (Definition B.3), this indeed works.

Proposition B.3 *Let $K \subset \Re^n$ be a closed convex cone. Then $(K^*)^* = K$.*

Maybe surprisingly, the proof of this innocent-looking fact already requires the machinery of separation theorems that will also be essential for cone programming duality below. Separation theorems generally assert that disjoint convex sets can be separated by a hyperplane.

B.4.1. A Separation Theorem for Closed Convex Cones

Theorem B.1 *Let $K \subset \Re^n$ be a closed convex cone, $\boldsymbol{b} \in \Re^n \backslash K$. Then there exists a vector $\boldsymbol{y} \in \Re^n$ such that*

$$\boldsymbol{y} \cdot \boldsymbol{x} \geq 0, \forall \boldsymbol{x} \in K, \quad \boldsymbol{b} \cdot \boldsymbol{y} < 0.$$

The statement is illustrated in Figure B.3 (left) for $\Re^2$. The hyperplane $h = \{\boldsymbol{x} \in \Re^2 | \boldsymbol{y} \cdot \boldsymbol{x} = 0\}$ (that passes through the origin) strictly separates $\boldsymbol{b}$ from K. We also say that $\boldsymbol{y}$ is a witness for $\boldsymbol{b} \notin K$.

Proof 13 *See [27].* □

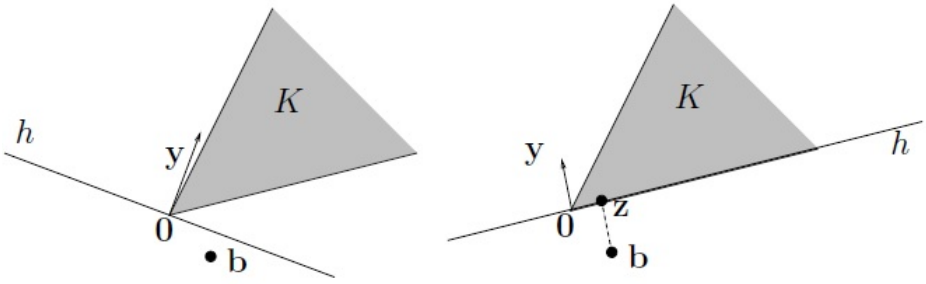


Figure B.3.: A point $\boldsymbol{b}$ not contained in a closed convex cone $K \subset \Re^2$ can be separated from K by a hyperplane $h = \{\boldsymbol{x} \in \Re^2 | \boldsymbol{y} \cdot \boldsymbol{x} = 0\}$ through the origin (left). The separating hyperplane resulting from the proof of Theorem B.1(right).

Using this result, we can now show that $(K^*)^* = K$ for any closed convex cone.

Proof of Proposition B.3.

For the direction$K \subset (K^*)^*$, we just need to apply the definition of duality: Let us choose $\boldsymbol{b} \in K$. By definition of K^*, $\boldsymbol{y} \cdot \boldsymbol{b} = \boldsymbol{b} \cdot \boldsymbol{y}, \forall \boldsymbol{y} \in K^*$; this in turn shows that $\boldsymbol{b} \in (K^*)^*$. For the other direction, we let $\boldsymbol{b} \notin K$. According to Theorem B.1, we find a vector $\boldsymbol{y}$ such $\boldsymbol{y} \cdot \boldsymbol{x} \geq 0, \forall \boldsymbol{x} \in K$ and $\boldsymbol{b} \cdot \boldsymbol{y} < 0$. The former inequality shows that $\boldsymbol{y} \in K^*$, but then the latter inequality witnesses that $\boldsymbol{b} \notin (K^*)^*$.

B.4.2. Adjoint Operators

We will now bring a linear operator into the game. Let V and W be Hilbert spaces and let $\mathbf{A} : V \to W$ be a linear operator from V into W. If V and W are finite dimension, then $\mathbf{A}$ can be represented as a matrix. But even in the general case, $\mathbf{A}$ behaves like a matrix for our purposes, and we will write $\mathbf{A}\boldsymbol{x}$ instead of $\mathbf{A}(\boldsymbol{x})$ for $\boldsymbol{x} \in V$. Here is the generalization of the transpose of a matrix.

Definition B.4 *Let* $\mathbf{A} : V \to W$ *be a linear operator. A linear operator* $\mathbf{A}^T : W \to V$ *is called an adjoint of* $\mathbf{A}$ *if*

$$\boldsymbol{y} \cdot \mathbf{A}\boldsymbol{x} = \mathbf{A}^T\boldsymbol{y} \cdot \boldsymbol{x} \quad \forall \boldsymbol{x} \in V, \boldsymbol{y} \in W$$

If V and W are finite dimension, there is an adjoint $\mathbf{A}^T$ of $\mathbf{A}$. In general, if there is an adjoint, then it is easy to see that it is unique which justifies the notation $\mathbf{A}^T$.

In order to stay as close as possible to the familiar matrix terminology, we will also introduce the following notation. If $V_1, V_2, \cdots, V_n$ and $W_1, W_2, \cdots, W_m$ are Hilbert spaces with linear operators $\mathbf{A}_{ij} : V_j \to W_i$, then we write the matrix

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1n} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{m1} & \mathbf{A}_{m2} & \cdots & \mathbf{A}_{mn} \end{bmatrix} \tag{B.4.1}$$

for the linear operator $\bar{\mathbf{A}} : V_1 \oplus V_2 \oplus \cdots \oplus V_n \to W_1 \oplus W_2 \oplus \cdots \oplus W_m$ defined by

$$\bar{\mathbf{A}}(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n) = (\sum_{j=1}^{n} \mathbf{A}_{1j}\boldsymbol{x}_j, \sum_{j=1}^{n} \mathbf{A}_{2j}\boldsymbol{x}_j, \cdots, \sum_{j=1}^{n} \mathbf{A}_{mj}\boldsymbol{x}_j).$$

A simple calculation then shows that

$$\mathbf{A}^T = \begin{bmatrix} \mathbf{A}_{11}^T & \mathbf{A}_{21}^T & \cdots & \mathbf{A}_{m1}^T \\ \mathbf{A}_{12}^T & \mathbf{A}_{22}^T & \cdots & \mathbf{A}_{m2}^T \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_{1n}^T & \mathbf{A}_{2n}^T & \cdots & \mathbf{A}_{mn}^T \end{bmatrix} \tag{B.4.2}$$

just as with matrices.

Proposition B.4 *Let* $\mathbf{A}$ *be a linear operator from* $\Re^n$ *into* $\Re^m$*, and let* C *be a nonempty closed convex cone in* $\Re^m$*. Then the following hold :*

(i) $(\mathbf{A}^{-1}(C))^{\ominus} = \overline{\mathbf{A}^T(C^{\ominus})}$.

(ii) $(\mathbf{A}^T)^{-1}(C) = (\mathbf{A}(C^{\ominus}))^{\ominus}$.

Proof 14 *See [6].*

B.4.3. Farkas Lemma

Proposition B.5 *Let $K \subset \Re^n$ be a closed convex cone, and $C = \{\mathbf{A}\boldsymbol{x} | \boldsymbol{x} \in K\}$. Then $\bar{C}$, the closure of C, is a closed convex cone.*

Proof 15 *By definition, the closure of C is the set of all limit points of C. Formally, $\boldsymbol{b} \in \bar{C}$ if and only if there exists a sequence $(\boldsymbol{y}_k)_k \in N$ such that $\boldsymbol{y}_k \in C$ for all k and $\lim_{k\to\infty} \boldsymbol{y}_k = \boldsymbol{b}$. This yields that $\bar{C}$ is a convex cone, using that C is a convex cone. In addition, $\bar{C}$ is closed.* □

The fact $\boldsymbol{b} \in \bar{C}$ can be formulated without reference to the cone C, and this will be more convenient in what follows.

Definition B.5 *Let $K \subset \Re^n$ be a closed convex cone. The system*

$$\mathbf{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \in K,$$

is called subfeasible if there exists a sequence $(\boldsymbol{x}_k)_{k\in N}$ such that $\boldsymbol{x}_k \in K$ for all $k \in N$ and

$$\lim_{k\to\infty} \mathbf{A}\boldsymbol{x} = \boldsymbol{b}.$$

Here is Farkas lemma for equation systems over closed convex cones.

Proposition B.6 *Let $K \subset \Re^n$ be a closed convex cone, and $\boldsymbol{b} \in \Re^m$. The system $\mathbf{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \in K$ is subfeasible if and only if every $\boldsymbol{y} \in \Re^m$ with $\mathbf{A}^T\boldsymbol{y} \in K^*$ also satisfies $\boldsymbol{b} \cdot \boldsymbol{y} \geq 0$.*

Proof 16 *See [27].*

Bibliography

[1] E.D. Andersen, C. Roos, and T. Terlaky. *On implementing a primal-dual interior-point method for conic quadratic optimization*, volume 95:249–277. 2003.

[2] MOSEK ApS. The mosek optimization tools version 3.2 (revision 8). Avail. http://www.mosek.com, 2005.

[3] J.B. Baillon, R.E. Bruck, and S. Reich. On the asymptotic behavior of nonexpansive mappings and semigroups in Banach spaces. *Houston J. Math.*, 4:1–9, 1978.

[4] H.H. Bauschke and J.M. Borwein. On the convergence of von Neumanns alternating projection algorithm for two sets. *Set-Valued. Anal.*, 1:185–212, 1993.

[5] H.H. Bauschke and J.M. Borwein. On projection algorithms for solving convex feasibility problems. *SIAM Review.*, 38:367–426, 1996.

[6] H.H. Bauschke and P.L. Combettes. *Convex analysis and monotone operator theory in Hilbert spaces*. Springer, 2010.

[7] H.H. Bauschke, P.L. Combettes, and D. Russel Lukel. Finding best approximation pairs relative to two closed convex sets in Hilbert spaces. *J. Approx. Theory.*, 127:178–192, 2004.

[8] D.P. Bertsekas. *Nonlinear programming.* Athena Scientific, second edition, 1999.

[9] D.P. Bertsekas. *Convex analysis and optimization.* Athena Scientific, 2003.

[10] D.P. Bertsekas and Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Prentice Hall, New York, 1989.

[11] S. Boyd and L. Vandenberghe. *Convex optimisation.* Cambridge University Press, 2004.

[12] R.E. Bruck and S. Reich. Nonexpansive projections and resolvents of accretive operators in Banach spaces. *Houston J. Math.*, 3:459–470, 1977.

[13] G. Chen and M. Teboulle. A proximal-based decomposition method for convex minimization problems. *Math. Progr. Ser. A*, 64:81–101, 1994.

[14] W.F. Chen. *Limit analysis in soil mechanics.* Elsevier, 1990.

[15] E. Christiansen. *Handbook of Numerical Analysis*, volume IV, chapter Limit Analysis of Collapse States. North Holland Amsterdam, 1996.

[16] H. Ciria. Computational of upper and lower bounds in limit analysis using second cone programming and mesh adaptivity, May 14 2004. MSc thesis.

[17] P.L. Combettes. Inconsistent signal feasibility problems: least-squares solutions in a product space. *IEEE Trans. Signal Process.*, 42:2955–2966, 1994.

[18] P.L. Combettes. Hilbertian convex feasibility problem: convergence of projection methods. *Appl. Mathl. Optim*, 35:311–330, 1997.

[19] A.J. Conejo, E. Castillo, R.Mínguez, and R. García-Bertrand. *Decomposition Techniques in Mathematical Programming.* Springer, 2006.

[20] G. Cornuejols and R. Tutuncu. *Optimisation methods in finace.* Cambridge, 2007.

[21] M. Vicente da Silva and A.N. Antão. A non-linear programming method approch for upper bound limit analysis. *Int. J. Numer. Meth. Eng.*, 72:1192–1218, 2007.

[22] M. Vicente da Silva and A.N. Antão. Upper bound limit analysis with a parallel mixed fnite element formulation. *Int. J. Solids Struct.*, 45:5788–5804, 2008.

[23] G.B. Dantzig and P. Wolfe. Decomposition principle for linear programming. *Math. Oper. Res.*, 8:101–111, 1960.

[24] F. Deutsch. *Best Approximation in Inner Product Spaces*. Springer, 2001.

[25] Q.T. Dinh, C. Savorgnan, and M. Diehl. Combining Lagrangian decomposition and excessive gap smoothing technique for solving larg-escale separable convex optimization problems. *Comput. Optim. Appl.*, 55:75–111, 2013.

[26] S.A. Gabriel, Y. Shim, A.J. Conejo, S. de la Torre, and R. Garca-Bertrand. A Benders decomposition method for discretely-constrained mathematical programs with equilibrium constraints. *J. Oper. Res. Soc.* , 61(9):1404–1419, 2010.

[27] B. Gartner and J. Matousek. *Approximation Algorithms and Semidefinite Programming*. Springer, 2012.

[28] M. Goldburg and R.J. Marks II. Signal synthesis in the presence of an inconsistent set of constraints. *IEEE Trans. Circuits Syst.*, 32:647–663, 1985.

[29] J. Huang, W. Xu, P. Thomson, and S. Di. A general rigid-plastic/rigid-viscoplastic FEM for metal-forming processes based on the potential reduction interior point method. *Int. J. Mach. Tool. Manuf.*, 43:379–389, 2003.

[30] I. Kaneko. A decomposition procedure for large-scale optimum plastic design problems. *Int. J. Numer. Meth. Eng.*, 19:873–889, 1983.

[31] K. Krabbenhøft, A.V. Lyamin, and S.W. Sloan. Formulation and solution of some plasticity problems as conic programs. *Int. J. Solids Struct.*, 44:1533–1549, 2007.

[32] A.V. Lyamin. Sonic. Solver for second order conic programing. 2004.

[33] A.V. Lyamin and S.W. Sloan. Lower bound limit analysis using non-linear programming. *Int. J. Numer. Meth. Eng.*, 55:576–611, 2002.

[34] A.V. Lyamin, S.W. Sloan, K. Krabbenhft K, and M. Hjiaj. Lower bound limit analysis with adaptive remeshing. *Int. J. Numer. Meth. Eng.*, 63:1961–1974, 2005.

[35] A. Makrodimopoulos and C.M. Martin. Lower bound limit analysis of cohesive-frictional materials using second-order cone programming. *Int. J. Numer. Meth. Eng.*, 66:604–634, 2006.

[36] R.D.C. Monteiro, C. Ortiz, and B.F. Svaiter. Implementation of a block-decomposition algorithm for solving large-scale conic semidefinite programming problems. *Comput. Optim. Appl.*, 2013.

[37] J.J. Muñoz, J. Bonet, A. Huerta, and J. Peraire. Upper and lower bounds in limit analysis: adaptive meshing strategies and discontinuous loading. *Int. J. Numer. Meth. Eng.*, 77:471–501, 2009.

[38] I. Necoara and J.A.K. Suyken. Application of a smoothing technique to decomposition in convex optimization. *IEEE. T. Automat. Contr.*, pages 2674–3679, 2008.

[39] Y. Nesterov. Smooth minimization of non-smooth functions. *Math. Progr. Ser. A*, 103(1):127–152, 2005.

[40] Z. Opial. Weak convergence of the sequence of successive approximations for nonexpansive mappings. *Bull. Amer. Math. Soc.*, 73:591–597, 1967.

[41] F. Pastor, E. Loute, J. Pastor, and M. Trillat. Mixed method and convex optimization for limit analysis of homogeneous Gurson materials: a kinematical approach. *Eur. J. Mech. A/Solids*, (28):25–35, 2009.

[42] A. Pazy. Asymptotic behavior of contractions in Hilbert space. *Israel. J. Math.*, 9:235–240, 1971.

[43] J.C. Pesquet and P.L. Combettes. Wavelet synthesis by alternating projections. *IEEE Trans. Signal Process.*, 44:728–732, 1996.

[44] R.L. Raffard, C.J. Tomlin, and S.P. Boyd. Distributed optimization for cooperative agents, application to formation flight. 43rd IEEE Conference on Decision and Contro, December 2004.

[45] J. F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optim. Method. Softw*, 11–12:625–653, 1999. Version 1.05 available from `http://http://sedumi.ie.lehigh.edu`.

[46] K.C. Toh, M.J. Todd, and R.H. Tutuncu. On the implementation and usage of SDPT3 a Matlab software package for semidefinite-quadratic linear programming. Version 4.0. Technical report,National University of Singapore, july 2006.

[47] R.H. Tütüncü, K.C. Toh, and M.J. Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *MPB*, 95:189–217, 2003. Avail. http://www.math.nus.edu.sg/ mattohkc/sdpt3.html.