AAR-based decomposition method for lower bound limit analysis

J. J. Muñoz, **N. Rabiei**
Department of Applied Mathematics III, Laboratory of Numerical Analysis (LaCàN), Universitat Politècnica de Catalunya (UPC), Urgell 187, 08036 Barcelona, Spain

Despite recent progress in optimisation techniques, finite element stability analysis of realistic three-dimensional (3D) problems is still hampered by the size of the resulting optimisation problem. Current solvers may take a prohibitive computational time, if they give a solution at all. Possible remedies to this are the design of adaptive de-remeshing techniques, decomposition of the system of equations, or the decomposition of the optimisation problem. This paper concentrates on the last approach, and presents an algorithm especially suited for limit analysis.

Optimisation problems in limit analysis are in general convex but non-linear. This fact renders the design of decomposition techniques specially challenging. The efficiency of general approaches such as Benders or Dantzig-Wolfe is not always satisfactory, and strongly depends on the structure of the optimisation problem. This work presents a new method that is based on rewriting the feasibility region of the global optimisation problem as the intersection of two subsets. By resorting to the Averaged Alternating Reflections (AAR) method in order to find the distance between the sets, the optimisation problem is successfully solved in a decomposed manner. Some representative examples illustrate the application of the method and its efficiency with respect to other well-known decomposition algorithms.

1. Introduction

Computational limit analysis aims to accurately compute the bearing capacity of structures. Mathematically, this can be stated as numerically solving the following maximisation (static) problem:

$$
\begin{aligned}
\lambda_{opt} = \max_{\lambda,\boldsymbol{\sigma}} & \lambda \\
s.t. & \nabla\cdot\boldsymbol{\sigma} + \lambda\boldsymbol{f} = \mathbf{0}, && \forall \boldsymbol{x}\in\Omega \\
& \boldsymbol{\sigma}\boldsymbol{n} = \lambda\boldsymbol{g}, && \forall \boldsymbol{x}\in\Gamma_n \\
& [\![\boldsymbol{\sigma}\boldsymbol{n}]\!] = \mathbf{0}, && \forall \boldsymbol{x}\in\Gamma_i \\
& \boldsymbol{\sigma}\in\mathcal{B}
\end{aligned}
\tag{1}
$$

Here, Ω is the domain of the body, while $\boldsymbol{\sigma}$, $\boldsymbol{f}$ and $\boldsymbol{g}$ are respectively the stress tensor, the volumetric loads, and the boundary loads. The conditions in the optimisation problem (1) correspond to the equilibrium conditions of a domain Ω, with applied boundary loads $\boldsymbol{g}$ on the boundary $\Gamma_n \subseteq \partial\Omega$, and with some potential discontinuities Γ_i. The set $\mathcal{B}$ represents the admissible domain for the plasticity criteria of the material.

Different discretisations of the optimisation problem in (1) yield different static and kinematic formulations that give respectively lower (Lyamin and Sloan, 2002; Lyamin *et al.*, 2005) or upper bounds (Krabbenhøft *et al.*, 2007) of the exact optimal load factor λ_{opt}, and the two type of solutions may be in turn combined for designing remeshing strategies (Muñoz *et al.*, 2009). The method has been well studied and applied for instance in the analysis of anchors (Merifield and Smith, 2010; Muñoz *et al.*, 2013a), masonry structures (Gilbert *et al.*, 2010) or inhomogeneous materials (Bleyer and de Buhan, 2014). This article focuses on the lower bound optimisation problem, although the ideas described below can be also applied to other formulations. By using a piecewise linear finite element discretisation of the stress variable $\boldsymbol{\sigma}$, and after using a linear transformation of the stresses, the analytical problem in (1) can be turned into the following finite optimisation problem (Lyamin and Sloan, 2002; Muñoz *et al.*, 2009):

$$
\begin{aligned}
\lambda^* = \max_{\lambda,\mathbf{x}} & \lambda \\
s.t. & \mathbf{A}\mathbf{x} + \lambda\mathbf{f} = \mathbf{b} \\
& \mathbf{x}\in\mathcal{K}
\end{aligned}
\tag{2}
$$

The vector $\mathbf{x}$ includes all the nodal components of the stress-like variable $\mathbf{x}$, which is a linear transformation of the stresses $\boldsymbol{\sigma}$, in such a manner that the new admissible set $\mathcal{K}$ is formed by the products of second order cones (SOCs), that is $\mathcal{K} = \mathcal{K}_1 \times \ldots \mathcal{K}_N$, with $\mathcal{K}_i =$

$\{y \in \mathbb{R}^n | y_1 \geq \sqrt{y_2^2 + \ldots + y_n^2}\}$. The membership constraint $x \in \mathcal{K}$ can be then easily dealt with by using standard optimisation software such as SDPT3 (Toh *et al.*, 2006), Mosek (MOSEK ApS., 2005) or Sonic (Lyamin, 2004). The static formulations solved here are such that the optimum value of the optimisation problem in (2), denoted by λ^*, is a lower bound of the exact solution in (1), i.e. $\lambda^* \leq \lambda_{opt}$. Matrix $\mathbf{A}$ is the result of discretisating the terms multiplying σ in the three sets of equilibrium equations in (1), while vector $\mathbf{f}$ contains those factors that are multiplied by λ in the same equations, and vector $\mathbf{b}$ stems from the discretisation and transformation of σ into the stress-like variable $\mathbf{x}$. Their specific form is not detailed here but may be found elsewhere (Lyamin and Sloan, 2002; Muñoz *et al.*, 2009).

Due to the size of the resulting optimisation problem in (2), limit analyses on three dimensional domains are scarce. Although optimisation solvers have increased their efficiency and reduced their computational cost, it is still very much desirable to design new methods that reduce the cost of the solution process. One of the possible remedies is to decompose the global problem in (2) into smaller sub-problems, which can be solved at a much lower cost. This idea is not new, and has already been used by Chen and Teboulle (1994); Kaneko (1983) using proximal-point decomposition, and by Pastor *et al.* (2009) and Kammoun *et al.* (2010) using overlapping domains. However, no optimal decomposition strategy exists in the optimisation literature, and that the standard techniques such as Benders (Benders, 1962; Geoffrion, 1972), Dantzig-Wolfe Conejo *et al.* (2006); Dantzig and Wolfe (1960), or primal and dual decomposition (Boyd *et al.*, 2007) require a very large number of iterations for the non-linear problem in (2) (Muñoz *et al.*, 2013b).

The proposed algorithm is based on computing the distance between two feasibility sets, which is found by using the method of Averaged Alternating Reflections (AAR). The analysis of the method and its converged properties can be found in (Rabiei and Muñoz, 2015). The method is here summarised and interpreted in mechanical terms, and it is applied to domains with an irregular subdivision, and to problems with and without boundary or body loads (gravity). The method is first presented and related to our decomposition algorithm. Then the performance of the method is illustrated with two representative examples. Finally the main contributions are commented in the last Section.

2. AAR-based decomposition algorithm

2.1. AAR algorithm

The AAR algorithm consists of finding the distance between two sets Z and W (Bauschke and Combettes, 2010). It is clear that if the distance between the sets W and Z, denoted by $d(W, Z)$, is equal to zero, and the sets are compact, there is a common element $\bar{\mathbf{t}} \in Z \cap W$. The AAR algorithm searches for such element $\bar{\mathbf{t}}$ by

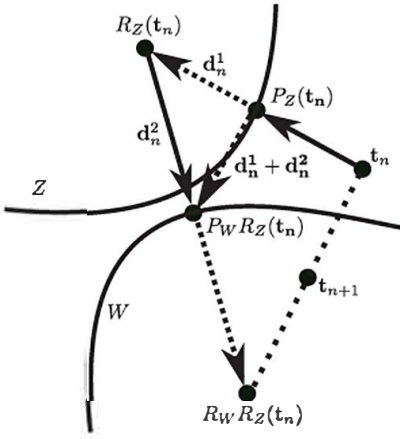

Figure 1. Illustration of the AAR algorithm for finding the distance between the two sets Z and W.

computing the fixed points of the following iterative process:

$$\mathbf{t}_{n+1} = \mathbf{T}(\mathbf{t}_n), \quad \text{with} \quad \mathbf{T} = \frac{\mathbf{R}_W \mathbf{R}_Z + \mathbf{I}}{2}. \tag{3}$$

The transformations $\mathbf{R}_W = 2\mathbf{P}_W - \mathbf{I}$ and $\mathbf{R}_Z = 2\mathbf{P}_Z - \mathbf{I}$ are the reflections on the sets W and Z, with $\mathbf{P}_W$ and $\mathbf{P}_Z$ the projections onto the same sets, respectively. Figure 1 illustrates the meaning of the transformations $\mathbf{T}$, $\mathbf{R}$ and $\mathbf{P}$.

Given a point $\mathbf{t}_n$, the AAR algorithm first searches for a point $\mathbf{R}_Z(\mathbf{t}_n)$ corresponding to the reflection of $\mathbf{t}_n$ into set Z. Then, it searches the reflection of such point $\mathbf{R}_Z(\mathbf{t}_n)$ into set W, denoted by $\mathbf{R}_W \mathbf{R}_Z(\mathbf{t}_n) = \mathbf{R}_W(\mathbf{R}_Z(\mathbf{t}_n))$. The AAR algorithm returns the average between this last point and the initial point $\mathbf{t}_n$, that is $\mathbf{t}_{n+1} = (\mathbf{t}_n + \mathbf{R}_W \mathbf{R}_W(\mathbf{t}_n))/2$. If $\mathbf{t}_n \in Z \cap W$, then $\mathbf{P}_W(\mathbf{t}_n) = \mathbf{P}_Z(\mathbf{t}_n) = \mathbf{R}_W(\mathbf{t}_n) = \mathbf{R}_Z(\mathbf{t}_n) = \mathbf{t}_n$, which means that $\mathbf{T}(\mathbf{t}_n) = \mathbf{t}_n$. Therefore, the vector $\mathbf{P}_W \mathbf{R}_Z(\mathbf{t}_n) - \mathbf{P}_Z(\mathbf{t}_n)$, denoted by $\mathbf{d}_n^1 + \mathbf{d}_n^2$ in Figure 1, is a measure of the distance between sets Z and W.

It is demonstrated in (Bauschke and Combettes, 2010), that when $d(W, Z) = 0$, the iterative process in (3) converges towards a fixed point such that $\bar{\mathbf{t}} = \mathbf{T}(\bar{\mathbf{t}})$ and $\bar{\mathbf{t}} \in Z \cap W$. And conversely, when $d(W, Z) > 0$, the algorithm gives a series of increasing values $||\mathbf{t}_n||$, but a converging series of distances $||\mathbf{d}_n^1 + \mathbf{d}_n^2||$.

2.2. Decomposition of lower bound optimisation problem

A decomposition method which splits the domain Ω into two non-overlapping domains is here proposed, Ω_1 and Ω_2, with $\Omega = \Omega_1 \cup \Omega_2$. The global optimisation problem in (2) is also rewritten into the

following partitioned form:

$$\begin{aligned}\lambda^* = \max_{\lambda,\mathbf{x}_1,\mathbf{x}_2}\ & \lambda \\ s.t. & \mathbf{A}_1\mathbf{x}_1 + \lambda\mathbf{f}_1 = \mathbf{b}_1 \\ & \mathbf{A}_2\mathbf{x}_2 + \lambda\mathbf{f}_2 = \mathbf{b}_2 \\ & \mathbf{B}_1\mathbf{x}_1 + \mathbf{B}_2\mathbf{x}_2 = 0 \\ & \mathbf{x}_1 \in \mathcal{K}_1, \mathbf{x}_2 \in \mathcal{K}_2\end{aligned}$$

where $\mathbf{x}_1 \in \Omega_1$ and $\mathbf{x}_2 \in \Omega_2$ are the stress-like nodal variables in each domain. It will be convenient to rewrite the complicating constraint $\mathbf{B}_1\mathbf{x}_1 + \mathbf{B}_2\mathbf{x}_2 = 0$ into two constraints, and use a new complicating variable $\mathbf{t}$ in such a manner that the optimisation problem above now reads

$$\begin{aligned}\lambda^* = \max_{\lambda,\mathbf{x}_1,\mathbf{x}_2,\mathbf{t}}\ & \lambda \\ s.t. & \mathbf{A}_1\mathbf{x}_1 + \lambda\mathbf{f}_1 = \mathbf{b}_1 \\ & \mathbf{A}_2\mathbf{x}_2 + \lambda\mathbf{f}_2 = \mathbf{b}_2 \\ & \mathbf{B}_1\mathbf{x}_1 = \mathbf{t} \\ & \mathbf{B}_2\mathbf{x}_2 = -\mathbf{t} \\ & \mathbf{x}_1 \in \mathcal{K}_1, \mathbf{x}_2 \in \mathcal{K}_2\end{aligned} \tag{4}$$

The new variable $\mathbf{t}$ corresponds to the nodal tractions at the common boundary $\Omega_1 \cap \Omega_2$, as illustrated in Figure 2. This variable allows us to decompose in turn the optimisation problem in (4) into the following master problem

$$\begin{aligned}\lambda^* = \max_{\lambda}\ & \lambda \\ s.t. \quad & d(Z(\lambda), W(\lambda)) = 0\end{aligned} \tag{5}$$

where $Z(\lambda)$ and $W(\lambda)$ are the following feasible sets:

$$\begin{aligned}W(\lambda) &= \{\mathbf{t} | \mathbf{A}_1\mathbf{x}_1 + \lambda\mathbf{f}_1 = \mathbf{b}_1, \mathbf{B}_1\mathbf{x}_1 = \mathbf{t}, \mathbf{x}_1 \in \mathcal{K}_1\} \\ Z(\lambda) &= \{\mathbf{t} | \mathbf{A}_2\mathbf{x}_2 + \lambda\mathbf{f}_2 = \mathbf{b}_2, \mathbf{B}_2\mathbf{x}_2 = -\mathbf{t}, \mathbf{x}_2 \in \mathcal{K}_2\}\end{aligned} \tag{6}$$

In mechanical terms, $W(\lambda)$ and $Z(\lambda)$ represent the sets of tractions at the boundary $\Omega_1 \cap \Omega_2$ that are in equilibrium with the given load factor λ, and with admissible stresses $\mathbf{x}_1 \in \mathcal{K}_1$ and $\mathbf{x}_2 \in \mathcal{K}_2$, respectively.

In order to explain the application of the AAR algorithm, let us denote by $\bar{\lambda}$ and $\underline{\lambda}$ strict upper and lower bounds respectively of λ^*, i.e. $\lambda^* < \bar{\lambda}$ and $\lambda^* > \underline{\lambda}$. Then, since $\bar{\lambda}$ is not globally feasible, no common traction field $\mathbf{t}$ at the boundary can be found that is in equilibrium with admissible stresses $\mathbf{x}_1$ and $\mathbf{x}_2$, and with the load factor $\bar{\lambda}$. Therefore $d(Z(\bar{\lambda}), W(\bar{\lambda})) > 0$. Furthermore, it may occur that at least one of the sets $W(\bar{\lambda})$ or $Z(\bar{\lambda})$ is empty since no equilibrated traction field can be found for the load factor $\bar{\lambda}$ in one of the sub-domains. On the other hand, since the value $\underline{\lambda}$ is globally feasible, the intersection $W(\underline{\lambda}) \cap Z(\underline{\lambda})$ is non-empty, and thus $d(Z(\underline{\lambda}), W(\underline{\lambda})) = 0$.

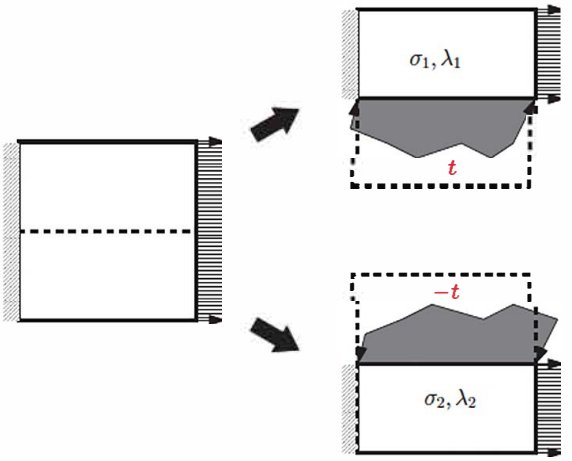

Figure 2. Partitioning of the domain Ω (left) into domains Ω_1 and Ω_2 (right). Variable t are the tractions at the common boundary $\Omega_1 \cap \Omega_2$.

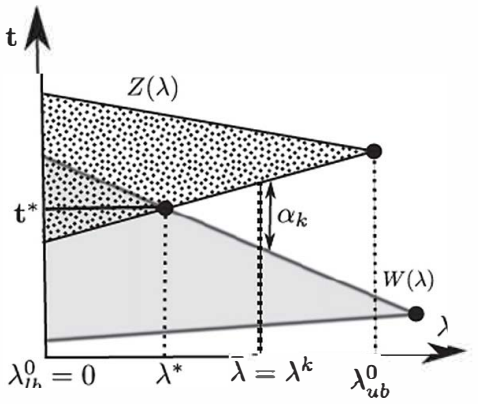

Figure 3. Illustration of the feasibility sets $Z(\lambda)$ and $W(\lambda)$ as a function of λ, and the optimal value λ^*.

Figure 3 shows schematically the sets $W(\lambda)$ and $Z(\lambda)$ on the $(\lambda, \mathbf{t})$-plane, and the situations when $\lambda < \lambda^*$ and $\lambda > \lambda^*$. In each case it can be observed that:

$$\begin{cases}\lambda \leq \lambda^* \Leftrightarrow W(\lambda) \cap Z(\lambda) \neq \emptyset \Leftrightarrow d(W(\lambda), Z(\lambda)) = 0 \\ \lambda > \lambda^* \Leftrightarrow W(\lambda) \cap Z(\lambda) = \emptyset \Leftrightarrow d(W(\lambda), Z(\lambda)) > 0\end{cases} \tag{7}$$

These implications justify the form of the optimisation problem given in (5), which in mechanical terms read: find the maximum load factor λ^* such that the two domains Ω_1 and Ω_2 are in equilibrium with a common traction field $\mathbf{t}^*$, and such that the stress variables $\mathbf{x}_1$ and $\mathbf{x}_2$ are plastically admissible and in equilibrium with $\mathbf{t}^*$.

2.3. Master problem and sub-problems

In view of the previous results, the following master problem for solving the optimal problem in (5) is proposed:

- Initialise: find $\lambda_{lb}^0 < \lambda^*$ and $\lambda_{ub}^0 > \lambda^*$. Set $\lambda^0 = \frac{1}{2}(\lambda_{lb}^0 + \lambda_{ub}^0)$ and $k = 0$.
- Step 1: Find the distance $\alpha^k = d(W(\lambda^k), Z(\lambda^k))$.
- Step 2:
  - 2.1 If $\alpha^k = 0$: Set $\lambda_{lb}^{k+1} = \lambda^k$ and $\lambda_{ub}^{k+1} = \lambda_{ub}^k$.
  - 2.2 If $\alpha^k > 0$: Set $\lambda_{lb}^{k+1} = \lambda_{lb}^k$ and $\lambda_{ub}^{k+1} = \lambda^k$.
- Step 3: Compute $\epsilon_\lambda = \lambda_{lb}^{k+1} - \lambda_{lb}^{k+1}$.
  - 3.1 If $\epsilon_\lambda < tol$: STOP.
  - 3.2 If $\epsilon_\lambda \geq tol$: $\lambda^{k+1} = \left(\lambda_{lb}^{k+1} + \lambda_{ub}^{k+1}\right)/2$, set $k = k + 1$ and GO TO Step 1.

Step 1 requires the computation of the distance $\alpha^k = d(W(\lambda^k), Z(\lambda^k))$. This step will be completed by resorting to the AAR algorithm presented in Section 2.1 with $W(\lambda^k)$ and $Z(\lambda^k)$ the feasibility sets of each sub-domain for a constant value of the load factor λ^k. In fact, since the algorithm does not require the computation of the actual value of the distance, but just in detecting whether α^k is positive, the following sub-problem will be followed:

- Step 1.1. Set $\lambda = \lambda^k$, $\mathbf{t}_0^k = \mathbf{t}^{k-1}$, $n = 0$
- Step 1.2. Solve Sub-problem 1 in (8). Obtain $\mathbf{d}_n^1$ and set $\tilde{\mathbf{t}}_n^k = \mathbf{R}_Z(\mathbf{t}_n^k) = \mathbf{t}_n^k + 2\mathbf{d}_n^1$.
- Step 1.3. Solve Sub-problem 2 in (9). Obtain $\mathbf{d}_n^2$ and set $\hat{\mathbf{t}}_n^k = \mathbf{R}_W\mathbf{R}_Z(\mathbf{t}_n^k) = \tilde{\mathbf{t}}_n^k + 2\mathbf{d}_n^2$.
- Step 1.4. Set $\beta_n = ||\mathbf{d}_n^1|| + ||\mathbf{d}_n^2||$ and $\alpha_n = ||\mathbf{d}_n^1 + \mathbf{d}_n^2||$.
  If $\beta_{n-1} > \beta_n$ or $\alpha_n < \epsilon_{1\alpha}$
  - 1.4.1. Set $\alpha^k = 0$. STOP

  elseif $\beta_{n-1} < \beta_n$ and $\Delta\alpha_n < \epsilon_{2\alpha}$
  - 1.4.2. Set $\alpha^k > 0$. STOP

  else
  - 1.4.3. Set $\mathbf{t}_{n+1}^k = \mathbf{T}(\mathbf{t}_n) = \left(\hat{\mathbf{t}}_n^k + \mathbf{t}_n^k\right)/2$, $n = n + 1$, GO TO Step 1.2

The n-iterations in the Steps 1.2-1.4 will be called *sub-iterations*, while the k-iterations in steps 1-3 will be called master-iterations. Also, α^k denotes the distance between the feasibility sets $W(\lambda^k)$ and $Z(\lambda^k)$, while α_n denotes the iterative values of these measures within the sub-iterations in Steps 1.2-1.4, and in fact $\alpha^k = \lim_{n\to\infty} \alpha_n$. The sub-iterations 1.2-1.4 aim precisely to detect whether $\alpha^k > 0$ or $\alpha^k = 0$ for a number of sub-iterations as small as possible.

Step 1.1. consists on the initialisation of the algorithm. The initial lower and upper bounds, λ_{lb}^0 and λ_{ub}^0, can be computed by setting $\lambda_{lb}^0 = 0$, and $\lambda_{ub}^0 = \min(\lambda_1^0, \lambda_2^0)$, with λ_1^0 and λ_2^0 the optimal values of λ when the global problem is solved with only the constraints of domain Ω_1 and Ω_2, respectively. Since the initial value of the traction vector $\mathbf{t}$ does not need to belong to $Z(\lambda^k)$ or $W(\lambda^k)$, the value $\mathbf{t}_0^0 = \mathbf{0}$ is set.

The sub-problems 1 and 2, in Steps 1.2 and 1.3, correspond respectively to compute the transformations $\mathbf{R}_Z(\mathbf{t}_n^k)$ and $\mathbf{R}_W\mathbf{R}_Z(\mathbf{t}_n^k)$ shown in Figure 1, necessary at each iteration of the AAR algorithm in order to obtain the new point $\mathbf{t}_{n+1}^k = \mathbf{T}(\mathbf{t}_n^k)$. This transformation requires the vectors $\mathbf{d}_n^1$ and $\mathbf{d}_n^2$ drawn in Figure 1, which are computed by solving the following two sub-problems,

$$
\begin{aligned}
&\min_{\mathbf{x}_1, \mathbf{d}^1} ||\mathbf{d}^1|| \\
&s.t. \mathbf{A}_1\mathbf{x}_1 + \lambda\mathbf{f}_1 = \mathbf{b}_1 \\
&\quad \mathbf{B}_1\mathbf{x}_1 - \mathbf{d}^1 = \mathbf{t}_n \\
&\quad \mathbf{x}_1 \in \mathcal{K}_1
\end{aligned} \tag{8}
$$

and

$$
\begin{aligned}
&\min_{\mathbf{x}_2, \mathbf{d}^2} ||\mathbf{d}^2|| \\
&s.t. \mathbf{A}_2\mathbf{x}_1 + \lambda\mathbf{f}_2 = \mathbf{b}_2 \\
&\quad \mathbf{B}_2\mathbf{x}_2 + \mathbf{d}^2 = -\mathbf{t}_n - 2\mathbf{d}_n^1 \\
&\quad \mathbf{x}_2 \in \mathcal{K}_2
\end{aligned} \tag{9}
$$

Note that according to the forms in (8) and (9), the vectors $\mathbf{d}^1 = \mathbf{B}_1\mathbf{x}_1 - \mathbf{t}_n$ and $\mathbf{d}^2 = -(\mathbf{B}_2\mathbf{x}_2 + \mathbf{R}_Z(\mathbf{t}_n))$ measure respectively the distance between $\mathbf{t}_n$ and $Z(\lambda)$, and between $\mathbf{R}_Z(\mathbf{t}_n) = \mathbf{t}_n + 2\mathbf{d}_n^1$ and $W(\lambda)$, as the AAR-algorithm in Figure 1 requires. The problems in (8)-(9) give the optimal solutions $\mathbf{d}_n^1$ and $\mathbf{d}_n^2$, which are thus used to measure $d(W(\lambda^k), Z(\lambda^k))$.

It has been numerically found that in fact, the value of $\beta_n = ||\mathbf{d}_n^1|| + ||\mathbf{d}_n^2||$ is a better indicator of the convergence trend of the AAR algorithm in (8)-(9) than the parameter $\alpha_n = ||\mathbf{d}_n^1 + \mathbf{d}_n^2||$. The stopping criteria in Step 1.4 uses the two parameters, α_n and β_n. Actually, the criteria in step 1.4. aims to detect, before an accurate value of α^k is computed, whether the AAR method is converging towards a fixed value (and α_n tends to zero), or diverging (β_n is increasing and α_n converges towards a non-zero value). In step 1.4.1. the value of λ^k is detected as a lower bound, while in 1.4.2. λ^k is detected as an upper bound. In Step 1.4.3 no identification can be said yet, and the iterative process in Steps 1.2-1.4 continues. The convergence of the algorithm is demonstrated in (Rabiei and Muñoz, 2015) for general non-linear problems. The next section shows its performance for some illustrative examples.

Each one of these problems has the structure of a second order cone program (SOCP), which can be solved by using standard optimisation packages.

3. Results

This section presents the numerical results of the proposed method in an example with non-zero boundary loads and no body loads, and a second example with only body loads. In the two examples

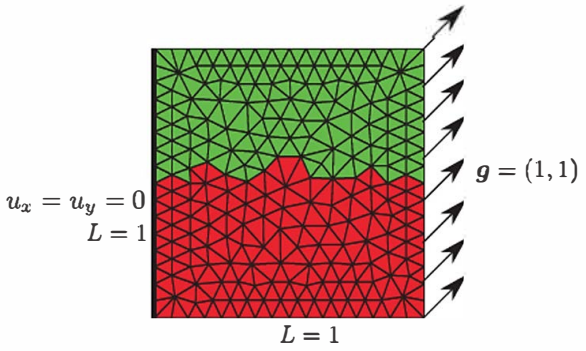

Figure 4. Unstructured mesh in Problem 1 (nelem=402) of the example in Section 3.1. The value of L denotes the length of the closest straight boundary

Problem	Nelem	$\sum_k n_k$
1	402	71
2	648	63
3	936	66

Table 1. Size and total number of sub-iterations (cumulative number of steps 1.2-1.3 during all the master-iterations) in each problem of the example in Section 3.1.

the domain is sub-divided with an irregular interface, and using the tolerances $(tol, \epsilon_{1\alpha}, \epsilon_{2\alpha}) = (5E-4, 1E-4, 1E-4)$.

3.1. Square domain with boundary loads

The AAR-based decomposition method is applied to the analysis of a square domain $\Omega = [0, 1] \times [0, 1]$ with a fixed left boundary, and the right boundary subjected to a nominal traction equal to $g = \{1, 1\}^T$. The geometry and boundary conditions are depicted in Figure 4. Three discretisations with three different number of elements (Nelem) have been used, which are indicated in Table 1. The third column in the table also shows the accumulated number of iterations in the sub-problems, that is, the sum of the n_k sub-iterations for each master iteration k. The domain has been partitioned horizontally, although it has been numerically tested that the results shown here do not qualitatively depend on the actual partitioning and regularity of the mesh.

The optimal solution $\lambda^\star$ for each problem has been computed solving the global problem in (2) with SONIC solver (Lyamin, 2004).

The numerical results of Problems 1-3 are reported in Table 2-4, respectively, where k indicates the number of master iterations, and n_k is the number of iterations taken by the sub-problem at each master iteration k. The second and third columns indicate

Problem 1					
$\lambda^\star = 0.50280$					
k	λ_{lb}^{k-1}	λ_{ub}^{k-1}	λ^k	n_k	$\Delta\lambda^{k-1}$
1	0.0	0.704068	0.352034	2	0.7041
2	0.352034	0.704068	**0.528051**	31	0.3520
3	0.352034	0.528051	0.440043	2	0.1760
4	0.440043	0.528051	0.484047	2	0.0880
5	0.484047	0.528051	**0.506049**	11	0.0440
6	0.484047	0.506049	0.495048	2	0.0220
7	0.495048	0.506049	0.500548	2	0.0110
8	0.495048	0.503299	**0.503299**	9	0.0083
9	0.501924	0.503299	0.501924	2	0.0014
10	0.502611	0.503299	0.502611	2	0.0007
11	0.502611	0.502955	**0.502955**	4	0.0003
12	0.502783	0.502955	0.502783	2	0.0002
–	0.502783	0.502955	0.502869	$\sum$=71	0.0002

Table 2. Numerical results of Problem 1 of the example in Section 3.1.

Problem 2					
$\lambda^\star = 0.50371$					
k	λ_{lb}^{k-1}	λ_{ub}^{k-1}	λ^k	n_k	$\Delta\lambda^{k-1}$
1	0.0	0.745779	0.372889	2	0.7458
2	0.372889	0.745779	**0.559334**	21	0.3729
3	0.372889	0.559334	0.466112	2	0.1864
4	0.466112	0.559334	**0.512723**	9	0.0932
5	0.466112	0.512723	0.489417	2	0.0466
6	0.489417	0.512723	0.501070	2	0.0233
7	0.489417	0.506897	**0.506897**	7	0.0175
8	0.489417	0.503983	**0.503983**	9	0.0146
9	0.502527	0.503983	0.502527	2	0.0015
10	0.503255	0.503983	0.503255	2	0.0007
11	0.503619	0.503983	0.503619	2	0.0004
12	0.503619	0.503801	**0.503801**	3	0.0002
–	0.503619	0.503801	0.503710	$\sum$=63	0.0002

Table 3. Numerical results of Problem 2 of the example in Section 3.1.

the highest lower bound and the lowest upper bound at each master iteration, in such a way that $\lambda^\star \in [\lambda_{lb}^{k-1}, \lambda_{ub}^{k-1}]$, and $\lambda^k = \frac{(\lambda_{lb}^{k-1} + \lambda_{ub}^{k-1})}{2}$. Numbers in bold font indicate that λ^k is an upper bound, and $\Delta\lambda^{k-1} = \lambda_{ub}^{k-1} - \lambda_{lb}^{k-1}$. It can be observed on the tables, and the plot in Figure 5, that whenever λ^k is an upper bound, the number of sub-iterations increases notoriously. In the next example alternative updates of the load factor in Step 3.2. are presented, that aim to amend this drawback.

Problem 3					
$\lambda^* = 0.50507$					
k	λ_{lb}^{k-1}	λ_{ub}^{k-1}	λ^k	n_k	$\Delta\lambda^{k-1}$
1	0.0	0.705498	0.352749	2	0.7055
2	0.352749	0.705498	0.529123	29	0.3527
3	0.352749	0.529123	0.440936	2	0.1764
4	0.440936	0.529123	0.485030	2	0.0882
5	0.485030	0.529123	0.507077	11	0.0441
6	0.485030	0.507077	0.496053	2	0.0220
7	0.496053	0.507077	0.501565	2	0.0110
8	0.501565	0.507077	0.504321	2	0.0055
9	0.504321	0.507077	0.505699	7	0.0028
10	0.504321	0.505699	0.505010	2	0.0014
11	0.505010	0.505699	0.505354	3	0.0007
12	0.505010	0.505354	0.505182	2	0.0003
–	0.505010	0.505182	0.505096	$\sum$=66	0.0002

Table 4. Numerical results of Problem 3 of the example in Section 3.1.

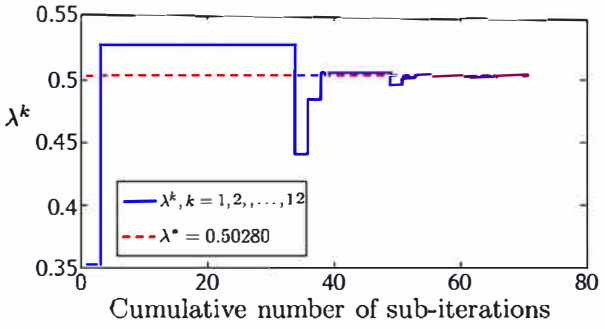

Figure 5. Evolution of the load factor for each sub-iteration of Problem 1 of the example in Section 3.1.

3.2. Vertical cut problem

This Section analyses the problem of a vertical cut problem, which has been also studied elsewhere (Lyamin and Sloan, 2002; Muñoz *et al.*, 2009). Figure 6 indicates the geometry, mesh and division employed. For a frictionless material ($\phi = 0$) and with $c = 1$, the global problem gives the optimum lower bound $\lambda^* = 2.66566$, while the decomposed problem gives the values indicated in the last row of Table 5.

Motivated by the large number of iterations that the algorithm takes when detecting an upper bound in the previous example, in this example alternative updates have been tested to the one indicated in Step 3.2. More specifically, the following two forms have been used:

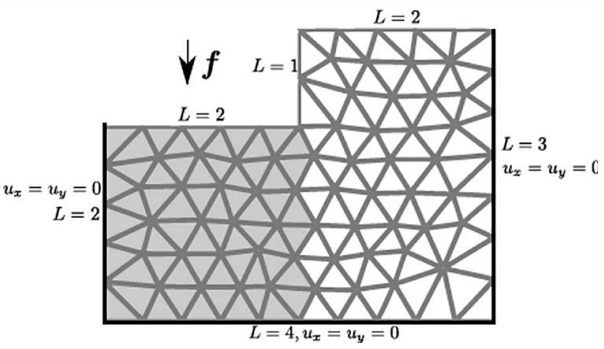

Figure 6. Geometry, mesh and boundary conditions of the vertical cut problem in Section 3.2. The value of L denotes the length of the closest straight boundary. Gray and white elements denote the two domains.

- Form 1:

$$\lambda^{k+1} = w_1 \lambda_{lb}^{k+1} + w_2 \lambda_{ub}^{k+1} \quad (10)$$

- Form 2:

$$\lambda^{k+1} = \lambda_{ub}^1 - \frac{\lambda_{ub}^1 - \lambda_{ub}^2}{\alpha^1 - \alpha^2}\alpha^1 \quad (11)$$

In the second option, the values $(\lambda_{ub}^1, \alpha^1)$ and $(\lambda_{ub}^2, \alpha^2)$ correspond to two pairs of previous upper bound load factors and the corresponding estimated values of α. This update is equivalent to estimating the value of λ^{k+1} from a linear approximation extrapolated from the previous two upper bounds (those that give $\alpha^k > 0$). The first update with $w_1 = w_2 = 1/2$ has been tested, as in the previous example, and also the choice $w_1 = 3w_2 = 3/4$. As it can be observed in Table 5, the choice in Form 2 is the one that gives the lowest number of iterations, and improves the convergence once two upper bounds have been detected. When the update in Form 2 was also tested in the first example in Section 3.1, the total number of iterations $\sum_k n_k$ was also reduced by 1, 7 and 1 iteration, for Problems 1, 2 and 3, respectively, while the update in Form 1 with $w_1 = 3w_2 = 3/4$ did not reduce the total number of iterations in two of the cases.

The computation of the global optimisation problem took 0.7s, while each sub-problem took between 0.2s and 0.3s. The difference between the global and decomposed problem of the tractions $\mathbf{t}$ at the interface between the two sub-domains depends obviously on the tolerance and number of sub-iterations. For the tolerances used, this difference was below $1E-4$ in all the cases tested.

In general, assuming that the cost of the optimisation problem increases approximately with $\mathcal{O}(nelem^2)$, the total number of iterations obtained using the decomposed algorithm is still not competitive. Nonetheless, the number of iterations that other

Vertical Cut problem, $\lambda^* = 2.66566$						
	$w_1 = w_2 = 1/2$		$w_1 = 3w_2 = 3/4$		Eqn. (11)	
k	λ^k	n_k	λ^k	n_k	λ^k	n_k
1	1.4469	2	1.4469	2	1.4469	2
2	2.1703	2	1.8086	2	2.1703	2
3	2.5321	2	22.0799	2	2.5321	2
4	**2.7129**	24	2.2834	2	**2.7129**	24
5	2.6225	2	2.4360	4	2.6225	2
6	**2.6677**	11	2.5504	2	**2.6677**	11
7	2.6451	2	2.6363	10	2.6642	2
8	2.6564	2	**2.7007**	27	**2.6659**	3
9	2.6621	2	2.6524	2	2.6651	2
10	2.6649	2	2.6644	2	**2.6655**	3
11	**2.6663**	3	**2.6735**	4	-	-
12	**2.6656**	3	**2.6667**	4	-	-
13	2.6652	2	2.6650	2	-	-
14	-	-	2.6654	2	-	-
15	-	-	**2.6658**	8	-	-
	2.66547	$\sum$=59	2.66556	$\sum$=74	2.66532	$\sum$=53

Table 5. Numerical results of the vertical cut problem in Section 3.2. Values in bold font denote upper bound load factors.

standard procedures would require, such as Benders or dual decomposition, has been reduced by one order of magnitude (Rabiei and Muñoz, 2015). Also, the number of sub-iterations does not scale with the problem size, and the memory requirements of the global problem have been reduced by a factor of two when using the decomposed algorithm.

4. Conclusions

This paper has presented and algorithm that solves the optimisation problem in limit analysis. The method exploits the structure of the optimisation problem: a single scalar on the objective function.

The proposed algorithm does not reduce the total computational cost, but halves the memory requirements, and does not scale with increasing number of elements. It has been shown that the use of irregular divisions of the domains does not alter the performance of the algorithm.

The reduction of the number of iterations when the estimate of the load factor λ^k is an upper bound is under investigation. Different choices have been already tested here. One of them uses a weighting that renders λ^{k+1} closer to λ^k_{lb}, while another extrapolates the value of λ^{k+1} from the previous upper bounds. Further alternative options are under study in order to reduce the total number of iterations.

5. Acknowledgements

The authors acknowledge the financial support of the Spanish Ministry of Economy and Competitiveness, through the research grant Nr. DPI2013-43727-R.

REFERENCES

Bauschke HH and Combettes PL (2010) *Convex analysis and monotone operator theory in Hilbert spaces*. CMS Books in Mathematics, Springer.

Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. *Numer. Math.* **4**: 238–252.

Bleyer J and de Buhan P (2014) A computational homogenization approach for the yield design of periodic thin plates. Part I: Construction of the macroscopic strength criterion. *Int. J. Solids Struct.* **51(13)**: 2448–2459.

Boyd S, Xiao L, Mutapcic A and Mattingley J (2007) *Notes on Decomposition Methods*. *Technical report*, Stanford University, http://stanford.edu/class/ee364b/lectures.html, Notes of EE364B course.

Chen G and Teboulle M (1994) A proximal-based decomposition method for convex minimization problems. *Math. Progr. Ser. A* **64**: 81–101.

Conejo A, Castillo E, Mínguez R and García-Bertrand R (2006) *Decomposition Techniques in Mathematical Programming*. Springer, The Netherlands.

Dantzig GB and Wolfe (1960) Decomposition principle for linear programs. *Oper. Res.* **8(1)**: 101–111.

Geoffrion AM (1972) Generalized Benders Decomposition. *J. Optim. Th. Appl.* **10(4)**: 238–252.

Gilbert M, Smith C and Pritchard T (2010) Masonry arch analysis using discontinuity layout optimisation. *Proc. Inst. Civil Eng. - Eng. Comp. Mech.* **163(3)**: 155–166.

Kammoun Z, Pastor F, Smaoui H and Pastor J (2010) Large static problem in numerical limit analysis: A decomposition approach. *Int. J. Num. Anal. Meth. Geomech.* **34**: 1960–1980.

Kaneko I (1983) A decomposition procedure for large-scale optimum plastic design problems. *Int. J. Num. Meth. Engng.* **19**: 873–889.

Krabbenhøft K, Lyamin AV and Sloan SW (2007) Formulation and solution of some plasticity problems as conic programs. *Int. J. Solids Struct.* **44**: 1533–1549.

Lyamin AV (2004) Sonic. Solver for second order conic programing.

Lyamin AV and Sloan SW (2002) Lower bound limit analysis using non-linear programming. *Int. J. Num. Meth. Engng.* **55**: 576–611.

Lyamin AV, Sloan SW, Krabbenhøft K and Hjiaj M (2005) Lower bound limit analysis with adaptive remeshing. *Int. J. Num. Meth. Engng.* **63**: 1961–1974.

Merifield R and Smith C (2010) The ultimate uplift capacity of multi-plate strip anchors in undrained clay. *Comp. Geotech.*

37(4): 504–514.

MOSEK ApS. (2005) The MOSEK optimization tools version 3.2 (Revision 8). *User's Manual and Reference* Avail. http://www.mosek.com.

Muñoz JJ, Bonet J, Huerta A and Peraire J (2009) Upper and lower bounds in limit analysis: adaptive meshing strategies and discontinuous loading. *Int. J. Num. Meth. Engng.* **77**: 471–501.

Muñoz JJ, Lyamin A and Huerta A (2013a) Stability of anchored sheet wall in cohesive-frictional soils by FE limit analysis. *Int. J. Num. Anal. Meth. Geomech.* **37(9)**: 1213–1230.

Muñoz JJ, Rabiei N, Lyamin A and Huerta A (2013b) *Direct Methods for Limit States in Structures and Materials*, chap. Computation of bounds for anchor problems in limit analysis and decomposition techniques. Springer Verlag, pp. 79–100.

Pastor F, Loute E and Pastor J (2009) Limit analysis and convex programming: A decomposition approach of the kinematic mixed method. *Int. J. Num. Meth. Engng.* **78**: 254–274.

Rabiei N and Muñoz JJ (2015) AAR-based decomposition method for non-linear convex optimization. *Comp. Opt. Appl.* On line.

Toh KC, Todd MJ and Tütüncü RHRH (2006) *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, http://www.math.nus.edu.sg/~mattohkc/sdpt3.html.