INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERING
Int. J. Numer. Meth. Engng 2002; **55**:573–611 (DOI: 10.1002/nme.511)

Lower bound limit analysis using non-linear programming

A. V. Lyamin and S. W. Sloan*,†

Department of Civil, Surveying and Environmental Engineering, University of Newcastle, NSW 2308, Australia

SUMMARY

This paper describes a new formulation, based on linear finite elements and non-linear programming, for computing rigorous lower bounds in 1, 2 and 3 dimensions. The resulting optimization problem is typically very large and highly sparse and is solved using a fast quasi-Newton method whose iteration count is largely independent of the mesh refinement. For two-dimensional applications, the new formulation is shown to be vastly superior to an equivalent formulation that is based on a linearized yield surface and linear programming. Although it has been developed primarily for geotechnical applications, the method can be used for a wide range of plasticity problems including those with inhomogeneous materials, complex loading, and complicated geometry. Copyright © 2002 John Wiley & Sons, Ltd.

KEY WORDS: static; lower bound; limit analysis; plasticity; finite element; formulation

1. INTRODUCTION

The static theorem of classical plasticity states that the limit load calculated from a statically admissible stress field is a lower bound on the true collapse load. In this context, a statically admissible stress field is one which satisfies equilibrium, the stress boundary conditions, and the yield condition. The lower bound theorem assumes a rigid plastic material model with an associated flow rule and is appealing because it provides a safe estimate of the load capacity of a structure.

Deriving statically admissible stress fields by hand is especially difficult for problems involving complicated geometries, inhomogeneous materials, or complex loading and numerical methods are usually called for. The most common numerical implementation of the lower bound theorem is based on a finite element discretization of the continuum and results in an optimization problem with large, sparse constraint matrices. By adopting linear finite elements and a linearized yield surface, the optimization problem is one which can be solved using classical linear programming techniques. This type of approach has been widely used and is described in detail, for example, in Bottero *et al.* [1], Sloan [2] and Pastor [3]. Although linear approximations of non-linear yield surfaces can be written easily for simple deformation

*Correspondence to: S. W. Sloan, Department of Civil, Surveying and Environmental Engineering, University of Newcastle, NSW 2308, Australia.
†E-mail: scott.sloan@newcastle.edu.au

Received 9 February 2001
Revised 18 October 2001

modes, such as those that arise in two-dimensional (2D) and axisymmetric loading, they are difficult to derive for three dimensional deformation. Moreover, since the linearization process may generate very large numbers of additional inequality constraints, the cost of solution is often excessive. Non-linear programming formulations, on the other hand, are more complex to solve but avoid the need to linearize the yield constraints and can be used for a wide range of yield functions under two- and three-dimensional loading.

This paper describes a discrete formulation of the lower bound theorem which is based on linear finite elements and non-linear programming. A fast, robust quasi-Newton algorithm for solving the resulting optimization problem is given and the performance of the new procedure is illustrated by applying it to a range of two- and three-dimensional examples. The results demonstrate its superiority over a conventional lower bound formulation that is based on linear programming.

2. DISCRETE FORMULATION OF LOWER BOUND THEOREM

Consider a body with a volume V and surface area A, as shown in Figure 1. Let $\mathbf{t}$ and $\mathbf{q}$ denote, respectively, a set of fixed tractions acting on the surface area A_t and a set of unknown tractions acting on the surface area A_q. Similarly, let $\mathbf{g}$ and $\mathbf{h}$ be a system of fixed and unknown body forces which act, respectively, on the volume V. Under these conditions, the objective of a lower bound calculation is to find a stress distribution which satisfies equilibrium throughout V, balances the prescribed tractions $\mathbf{t}$ on A_t, nowhere violates the yield criterion, and maximizes the integral

$$Q = \int_{A_q} \mathbf{q}\, \mathrm{d}A + \int_V \mathbf{h}\, \mathrm{d}V \tag{1}$$

Since this problem can be solved analytically for a few simple cases only, we seek a discrete numerical formulation which can model the stress field for problems with complex geometries, inhomogeneous material properties, and complicated loading patterns. The most appropriate method for this task is the finite element method.

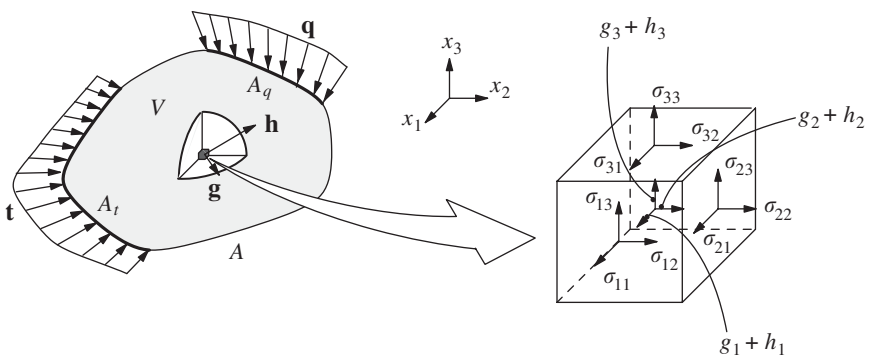


Figure 1. A body subject to a system of surface and body forces.

Disregarding, for the moment, the type of element that is used to approximate the stress field, any discrete formulation of the lower bound theorem leads to a constrained optimization problem of the form

$$\text{maximize } Q(x)$$

subject to

$$a_i(\mathbf{x}) = 0, \quad i \in I = \{1, \ldots, m\}$$
$$f_j(\mathbf{x}) \leqslant 0, \quad j \in J = \{1, \ldots, r\}$$
$$\mathbf{x} \in \mathrm{R}^n$$

where $\mathbf{x}$ is an n-dimensional vector of stress and body force variables and Q is the collapse load. The equalities defined by the functions a_i follow from the element equilibrium, discontinuity equilibrium, and boundary and loading conditions, while the inequalities defined by the functions f_j arise because of the yield constraints and the constraints on applied forces.

The following sections give a detailed description of the discretization procedure for the case of D-dimensional linear elements, where $D = 1, 2, 3$. As an illustration, the specific forms of the key equations for a two-dimensional formulation are given in Appendix A. There are strong reasons for choosing linear finite elements, and not higher order finite elements, for lower bound computations. The most important of these are:

- With a linear variation of the stresses, the yield condition is satisfied throughout an element if it is satisfied at all the element's nodes. This characteristic means that the yield constraints need be enforced only at each of the nodes, and thus gives rise to a minimum number of inequalities. Furthermore, for cone-like yield functions which have a linear envelope in the meridional plane, the strength parameters may vary linearly throughout the element and yet still preserve the lower bound property of the solution. This feature is invaluable for modelling the behaviour of inhomogeneous materials, such as undrained clays, where the strength properties may vary with the spatial co-ordinates.
- With linear elements it is simple to incorporate statically admissible stress discontinuities at interelement boundaries. These discontinuities permit large stress jumps over an infinitesimal distance and reduce the number of elements needed to predict the collapse load accurately. This feature, to a large extent, compensates for the low order of linear elements.
- Many problems in solid mechanics have boundaries that can be modelled, with sufficient accuracy, using a piecewise linear approximation. For curved boundaries which are subject to applied loading, a more accurate approximation of the geometry may be adopted when computing the equivalent nodal loads. This decreases the error introduced by a piecewise linear boundary, but preserves the lower bound nature of the solution.
- Linear finite elements generate only linear equalities in the final optimization problem. This feature allows us to develop algorithms for which the solution is kept within the feasible domain during every iteration. Maintaining feasibility during the iteration process is highly desirable when solving large optimization problems, as it limits error accumulation and permits 'backtracking' if numerical difficulties are encountered.
- All the mesh generation, assembly and solution software can be designed to be independent of the dimensionality of the problem. This means that the same code can deal

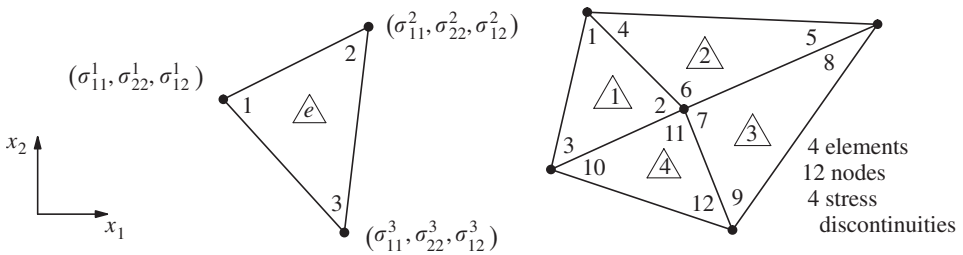


Figure 2. Linear stress triangle and finite element mesh for 2D lower bound analysis.

with one-, two- and three-dimensional geometries and greatly simplifies the software development process.

2.1. Linear finite elements

For the reasons discussed above, simplex finite elements are used to discretize the continuum. Unlike the usual form of the finite element method each node is unique to a particular element, multiple nodes can share the same co-ordinates, and statically admissible stress discontinuities are permitted at all interelement boundaries (Figure 2). If D is the dimensionality of the problem, then there are $D+1$ nodes in the element and each node is associated with $(D^2+D)/2$ independent components of the stress tensor σ_{ij}, $i=1,\ldots,D$; $j=i,\ldots,D$ (this set will be also denoted by the stress vector $\boldsymbol{\sigma}$). These stresses, together with the body force components h_i which act on a unit volume of material, are taken as the problem variables. The vector of unknowns for an element e is denoted by $\mathbf{x}^e$ and may be written as

$$\mathbf{x}^e = \{\{\sigma_{ij}^1\}^{\mathrm{T}}, \ldots, \{\sigma_{ij}^{D+1}\}^{\mathrm{T}}, \{h_i\}^{\mathrm{T}}\}^{\mathrm{T}}, \quad i=1,\ldots,D;\ j=i,\ldots,D$$

where $\{\sigma_{ij}^l\}$ are the stresses at node l and $\{h_i\}$ are the elemental body forces. For a D-dimensional mesh with N nodes and E elements, the total number of variables n is thus equal to $ND(D+1)/2 + DE$.

The variation of the stresses throughout each element may be written conveniently as

$$\boldsymbol{\sigma} = \sum_{l=1}^{D+1} N_l \boldsymbol{\sigma}^l \tag{2}$$

where N_l are linear shape functions. The latter can be expressed as

$$N_l = \frac{1}{|\mathbf{C}|} \sum_{k=0}^{D} (-1)^{l+k+1} |\mathbf{C}_{(l)(k)}| x_k \tag{3}$$

where x_k are the co-ordinates of the point at which the shape functions are to be computed (with the convention that $x_0 = 1$), $\mathbf{C}$ is a $(D+1)\times(D+1)$ matrix formed from the element

nodal co-ordinates according to

$$\mathbf{C}=\begin{vmatrix} 1 & x_1^1 & ,\ldots, & x_D^1 \\ 1 & x_1^2 & ,\ldots, & x_D^2 \\ \ldots & \ldots & ,\ldots, & \ldots \\ 1 & x_1^{D+1} & ,\ldots, & x_D^{D+1} \end{vmatrix} \quad \text{or} \quad \mathbf{C}=\begin{vmatrix} x_0^1 & x_1^1 & ,\ldots, & x_D^1 \\ x_0^2 & x_1^2 & ,\ldots, & x_D^2 \\ \ldots & \ldots & ,\ldots, & \ldots \\ x_0^{D+1} & x_1^{D+1} & ,\ldots, & x_D^{D+1} \end{vmatrix} \tag{4}$$

and $\mathbf{C}_{(l)(k)}$ is a $D\times D$ submatrix of $\mathbf{C}$ obtained by deleting the lth row and the kth column of $\mathbf{C}$ (the determinant $|\mathbf{C}_{(l)(k)}|$ is the *minor* of the entry c_{lk} of $\mathbf{C}$). In expressions (4), the superscripts are row numbers and correspond to the local node number of the element, while the subscripts are column numbers and designate the co-ordinate index. Elements in the first column of $\mathbf{C}$ are, by convention, allocated the subscript 0 and a value of unity. After grouping terms, Equation (3) can be written in the more compact form

$$N_l=\sum_{k=0}^{D} a_{lk}x_k, \quad \text{where } a_{lk}=a(l,k)=(-1)^{l+k+1}\frac{|\mathbf{C}_{(l)(k)}|}{|\mathbf{C}|} \tag{5}$$

2.2. Element equilibrium

To generate a statically admissible stress field, the stresses throughout each element must obey the equilibrium equations

$$\frac{\partial\sigma_{ij}}{\partial x_j}+h_i=-g_i, \quad i=1,\ldots,D \tag{6}$$

where σ_{ij} are Cartesian stress components, defined with respect to the axes x_j, and g_i and h_i are, respectively, prescribed and unknown body forces acting on a unit volume of material within the element.

Substituting Equations (2) and (5) into Equation (6) leads to linear equilibrium constraints of the form

$$\sum_{l=1}^{D+1}\frac{\partial N_l}{\partial x_j}\sigma_{ij}^l+h_i=\sum_{l=1}^{D+1}a_{lj}\sigma_{ij}^l+h_i=-g_i, \quad i=1,\ldots,D \tag{7}$$

In practice, however, we do not employ Equation (7) directly as this does not exploit the symmetry inherent in the stress tensor and results in too many variables. Instead we write the governing equations in terms of the stress vector, defined in Section 2.1, which reduces the number of problem unknowns by $D(D-1)/2$. Defining the two auxiliary index functions

$$\delta(i,k,m)=\delta_{ikm}=\begin{cases} 1 & \text{if } i=k \text{ or } i=m \\ 0 & \text{otherwise} \end{cases}$$

and

$$\phi(i,k,m)=\phi_{ikm}=\begin{cases} m & \text{if } i=k \\ k & \text{if } i=m \\ 1 & \text{otherwise} \end{cases}$$

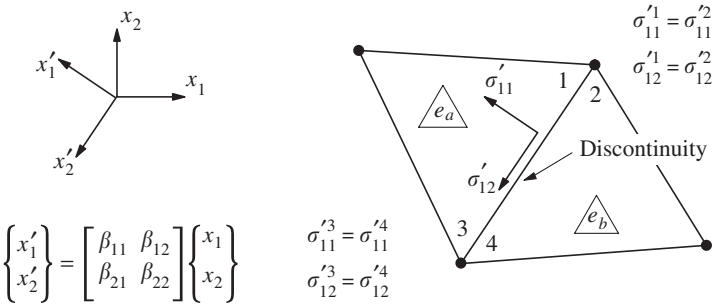


Figure 3. Statically admissible stress discontinuity in 2D.

we can rewrite (7) as

$$\sum_{l=1}^{D+1} \sum_{k=1}^{D} \sum_{m=k}^{D} \delta_{ikm} a(l, \phi_{ikm}) \sigma_{km}^{l} + h_i = -g_i, \quad i = 1, \ldots, D$$

In matrix notation, this becomes

$$\mathbf{A}^e_{\text{equil}} \mathbf{x}^e = \mathbf{b}^e_{\text{equil}} \tag{8}$$

where

$$\mathbf{A}^e_{\text{equil}} = [\mathbf{B}_1^{\mathrm{T}}, \ldots, \mathbf{B}_{D+1}^{\mathrm{T}} \ \mathbf{I}], \quad \mathbf{b}^e_{\text{equil}} = -\{g_1, \ldots, g_D\}^{\mathrm{T}}$$

and

$$\mathbf{B}_l^{\mathrm{T}} = \begin{bmatrix} \delta_{111} a(l, \phi_{111}) & \ldots, & \delta_{1DD} a(l, \phi_{1DD}) & \ldots, & \delta_{11D} a(l, \phi_{11D}) \\ \cdots & \ldots, & \cdots & \ldots, & \cdots \\ \delta_{D11} a(l, \phi_{D11}) & \ldots, & \delta_{DDD} a(l, \phi_{DDD}) & \ldots, & \delta_{D1D} a(l, \phi_{D1D}) \end{bmatrix}$$

Thus, in total, the equilibrium condition generates D equality constraints on the element's variables.

2.3. *Discontinuity equilibrium*

To incorporate statically admissible discontinuities at interelement boundaries, it is necessary to enforce additional constraints on the nodal stresses. A statically admissible discontinuity requires continuity of the shear and normal components but permits jumps in the tangential stress. Since the stresses vary linearly along each element side, static admissibility is guaranteed if the normal and shear stresses are forced to be equal at each pair of adjacent nodes on an interelement boundary.

In the previous section, the components of the stress tensor σ_{ij} are defined with respect to the rectangular Cartesian system with axes x_j, $j = 1, \ldots, D$. In addition to this global co-ordinate system, let us define a local system of Cartesian co-ordinates x'_k, $k = 1, \ldots, D$, with the same origin but oriented differently, and consider the stress components in this new reference

system (Figure 3). If we assume these two co-ordinate systems are related by the linear transformation

$$x'_k = \beta_{kj} x_j, \quad k = 1, \ldots, D$$

where β_{kj} are the direction cosines of the x'_k-axes with respect to the x_j-axes, then the tractions acting on a surface element, whose normal is parallel to one of the axes x'_k, are given by the vector $\mathbf{t}^k$ with components

$$t_i^k = \sigma_{ij} \beta_{jk}$$

The corresponding transformation law for the stress components is

$$\sigma'_{km} = \sigma_{ij} \beta_{ki} \beta_{mj}$$

If the normal to the discontinuity plane is chosen to be parallel to one of the axes x'_k, and l_{e_a}, l_{e_b} denotes the pair of adjacent nodes of neighbouring elements e_a and e_b, then the discontinuity equilibrium constraints for these nodes take the form

$$\sigma'^{l_{e_a}}_{km} = \sigma'^{l_{e_b}}_{km}, \quad m = 1, \ldots, D \tag{9}$$

Using the definition of the stress vector, and assuming that the normal to the discontinuity plane is parallel to the axis x'_1, (9) may be written as

$$\mathbf{A}^d_{\text{equil}} \boldsymbol{\sigma}^d = \mathbf{b}^d_{\text{equil}} \tag{10}$$

where

$$\mathbf{A}^d_{\text{equil}} = [\mathbf{S}^l \ \ -\mathbf{S}^l]$$

$$\boldsymbol{\sigma}^d = \{\{\boldsymbol{\sigma}^{l_{e_a}}\}^{\mathrm{T}}, \{\boldsymbol{\sigma}^{l_{e_b}}\}^{\mathrm{T}}\}^{\mathrm{T}}$$

$$\mathbf{b}^d_{\text{equil}} = \{\mathbf{0}\}$$

and

$$\mathbf{S}^l = \begin{bmatrix} \beta^l_{11}\beta^l_{11} & \ldots, & \beta^l_{1D}\beta^l_{D1} & \beta^l_{11}\beta^l_{21} + \beta^l_{12}\beta^l_{11} & \ldots, & \beta^l_{11}\beta^l_{D1} + \beta^l_{1D}\beta^l_{11} \\ \cdots & \ldots, & \cdots & \cdots & \ldots, & \cdots \\ \beta^l_{11}\beta^l_{1D} & \ldots, & \beta^l_{1D}\beta^l_{DD} & \beta^l_{11}\beta^l_{2D} + \beta^l_{12}\beta^l_{1D} & \ldots, & \beta^l_{11}\beta^l_{DD} + \beta^l_{1D}\beta^l_{1D} \end{bmatrix} \tag{11}$$

is a $D \times ((D^2 + D)/2)$ transformation matrix. Hence the equilibrium condition for each discontinuity generates D^2 equality constraints on the nodal stresses.

2.4. Boundary conditions

Consider a distribution of prescribed surface tractions t_p, $p \in P$, where P is a set of N_p prescribed components, which act over part of the boundary area A_t, as shown in Figure 4. For the case of a linear finite element, where the tractions are specified in terms of global

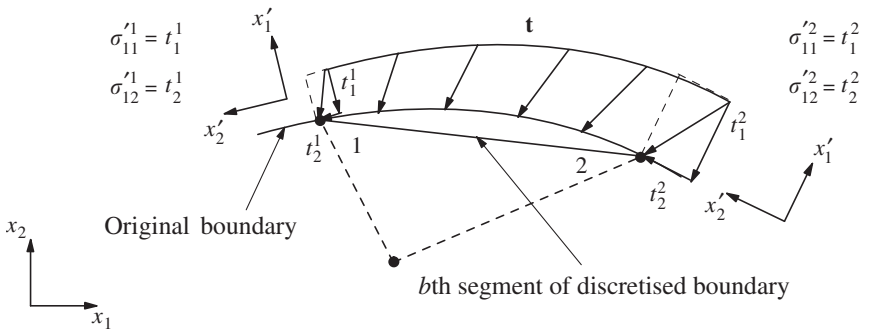


Figure 4. Stress boundary conditions in 2D.

co-ordinates over the linearized boundary area A_t^d, we can cast the stress boundary conditions for every node l as

$$\sigma_{ip}^l \beta_{ki}^l = t_p^l, \quad p \in P$$

Assuming the local co-ordinate system is chosen with x'_1 parallel to the surface normal at node l, this type of stress boundary condition gives rise to the constraints

$$\mathbf{A}^b_{\text{bound}} \boldsymbol{\sigma}^b = \mathbf{b}^b_{\text{bound}} \tag{12}$$

where

$$\begin{aligned} \boldsymbol{\sigma}^b &= \boldsymbol{\sigma}^l \\ \mathbf{A}^b_{\text{bound}} &= \left[\{\mathbf{R}^l_{p_1}\}^{\mathrm{T}} \dots \{\mathbf{R}^l_{p_{Np}}\}^{\mathrm{T}} \right]^{\mathrm{T}} \\ \mathbf{R}^l_p &= [\delta_{p11}\beta^l_{\phi_{p11},1}, \dots, \delta_{pDD}\beta^l_{\phi_{pDD},1} \ \delta_{p12}\beta^l_{\phi_{p12},1}, \dots, \delta_{p1D}\beta^l_{\phi_{p1D},1}] \\ \mathbf{b}^b_{\text{bound}} &= \{t^l_p\}^{\mathrm{T}}, \quad p \in P \end{aligned} \tag{13}$$

If the surface tractions are defined in terms of the local co-ordinate system, with the latter oriented so that one of the axes x'_k is normal to the boundary surface, then the stress boundary conditions may be written as

$$\sigma'^{l}_{kp} = \sigma^l_{ij}\beta^l_{ki}\beta^l_{pj} = t^l_p, \quad p \in P$$

In this case, which is the most common, Equation (12) still applies but

$$\begin{aligned} \mathbf{A}^b_{\text{bound}} &= \left[\{\mathbf{S}^l_{p_1}\}^{\mathrm{T}} \cdots \{\mathbf{S}^l_{p_{Np}}\}^{\mathrm{T}} \right]^{\mathrm{T}} \\ \mathbf{S}^l_p &= [\beta^l_{11}\beta^l_{1p}, \dots, \beta^l_{1D}\beta^l_{Dp} \ \beta^l_{11}\beta^l_{2p} + \beta^l_{12}\beta^l_{1p}, \dots, \beta^l_{11}\beta^l_{Dp} + \beta^l_{1D}\beta^l_{1p}] \\ \mathbf{b}^b_{\text{bound}} &= \{t^l_p\}^{\mathrm{T}}, \quad p \in P \end{aligned} \tag{14}$$

Thus every node which is subject to prescribed surface tractions generates a maximum of D equality constraints on the unknown stresses.

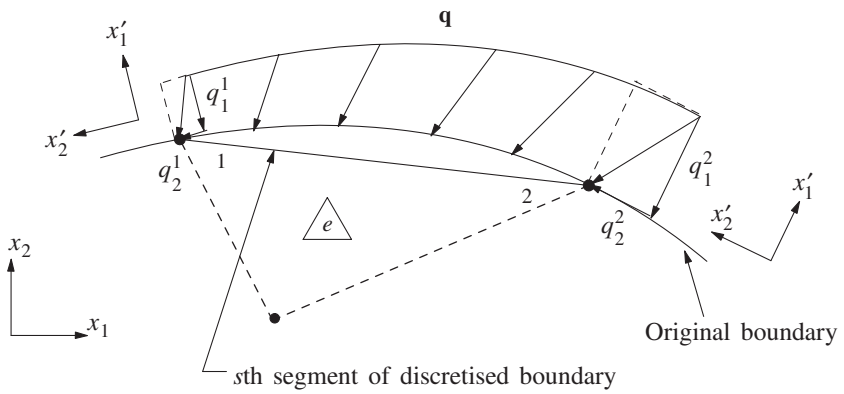


Figure 5. Surface loading in 2D.

2.5. *Objective function and loading constraints*

As indicated in Equation (1), the purpose of lower bound limit analysis is to find a statically admissible stress field which maximizes the load Q carried by a combination of surface tractions and body forces (Figure 5). The distribution of the latter may either be known or unknown, depending on the problem. In the terminology of mathematical programming, the expression for the collapse load Q is known as the objective function, since this is the quantity we want to maximize. In discrete form, integral (1) becomes

$$Q = \frac{1}{D} \sum_{s \in A_q^d} A_s \sum_{l=1}^{D} \sum_{o \in O_{\text{surf}}} \text{sign}(q_o^{ls}) \sigma_{oj}^{ls} \beta_{jk}^{ls} + \sum_{e \in V^d} V_e \sum_{o \in O_{\text{body}}} \text{sign}(h_o^e) h_o^e \tag{15}$$

where s denotes any side of a finite element, the superscript d signifies that the summations are taken over the discrete geometric approximation of the problem, A_s is the area of an element side, o denotes any component of a surface traction or body force that is to be optimized, $O_{\text{surf}} : O_{\text{surf}} \in \{1,2,3\}$ is the set of $N_{O_{\text{surf}}}$ surface stress components which are to be optimized, q_o^l is the oth component of the optimized surface traction at node l, the superscript ls denotes the lth node of the sth element side, V_e is the volume of an element, $O_{\text{body}} : O_{\text{body}} \in \{1,2,3\}$ is the set of $N_{O_{\text{body}}}$ element body force components which are to be optimized, h_o^e is the oth component of the optimized body force for element e, and $\text{sign}(q_o^{ls})$ and $\text{sign}(h_o^e)$ are signs of the oth surface traction and body force component distributions at node l and element e respectively.

Alternatively, in matrix notation, Equation (15) may be written in the form

$$Q = \frac{1}{D} \sum_{s \in A_q^d} A_s \sum_{l=1}^{D} \sum_{o \in O_{\text{surf}}} \text{sign}(q_o^{ls}) \mathbf{R}_o^{ls} \boldsymbol{\sigma}^{ls} + \sum_{e \in V^d} V_e \{\mathbf{i}_o\}^{\mathrm{T}} \mathbf{h}^e \tag{16}$$

where $\boldsymbol{\sigma}^{ls}$ is the unknown stress vector for node l of side s of an element, $\mathbf{h}^e$ is the unknown body force vector for element e, $\mathbf{R}_0$ is the matrix given by (13), and $\mathbf{i}_o$ is D-dimensional vector with entries of $\text{sign}(h_o^e)$ in its oth positions and zeros everywhere else.

When the tractions or body forces are to be maximized with respect to a local co-ordinate system, expression (15) transforms to

$$Q = \frac{1}{D} \sum_{s \in A_q^d} A_s \sum_{l=1}^{D} \sum_{o \in O_{\text{surf}}} \text{sign}(q_o^{ls}) \sigma_{ij}^{ls} \beta_{ki}^{ls} \beta_{jo}^{ls} + \sum_{e \in V^d} V_e \sum_{o \in O_{\text{body}}} \text{sign}(h_o^e) \beta_{oj}^e h_j^e \tag{17}$$

or

$$Q = \frac{1}{D} \sum_{s \in A_q^d} A_s \sum_{l=1}^{D} \sum_{o \in O_{\text{surf}}} \text{sign}(q_o^{ls}) \mathbf{S}_o^{ls} \boldsymbol{\sigma}^{ls} + \sum_{e \in V^d} V_e \{\mathbf{i}_o\}^{\mathrm{T}} \mathbf{B}^e \mathbf{h}^e \tag{18}$$

where $\mathbf{S}$ is given by Equation (14) and

$$\mathbf{B}^e = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1D} \\ \vdots & & \vdots \\ \beta_{D1} & \cdots & \beta_{DD} \end{bmatrix}$$

is a matrix of direction cosines which defines the orientation of the local element co-ordinate system relative to the global co-ordinate system.

In the more general case of combined global and local loading, the objective function incorporates parts of (15) and (17).

To define the orientation of the applied load, we can introduce additional loading constraints which specify the relative magnitude of the load components at all nodes on the loaded segments (for surface loading) or at all elements (for body force loading). Also, if the unknown load is applied with a known distribution (or profile) along a surface or throughout the body, then the loading constraints can be extended to include node-to-node or element-to-element relations for the appropriate load components. These types of constraints are linear and can be expressed in matrix form as

$$\mathbf{A}_{\text{load}} \mathbf{x} = \mathbf{b}_{\text{load}} \tag{19}$$

2.6. *Yield condition*

The general form of the yield condition for a perfectly plastic solid has the form

$$f(\sigma_{ij}) \leqslant 0$$

where f is a convex function of the stress components and material constants. The solution procedure presented later in this paper does not depend on a particular type of yield function, but does require it to be convex and smooth. Convexity is necessary to ensure the solution obtained from the optimization process is the global optimum, and is actually guaranteed by the assumptions of perfect plasticity. Smoothness is essential because the solution algorithm needs to compute first and second derivatives of the yield function with respect to the unknown stresses.

For yield functions which have singularities in their derivatives, such as the Tresca and Mohr–Coulomb criteria, it is necessary to adopt a smooth approximation of the original yield surface. A convenient hyperbolic approximation for smoothing the tip and corners of the Mohr–Coulomb model, which requires just two parameters, has been given by

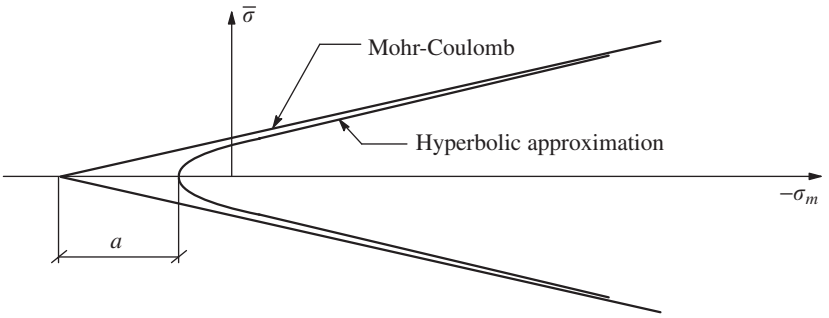


Figure 6. Hyperbolic approximation to Mohr–Coulomb yield function.

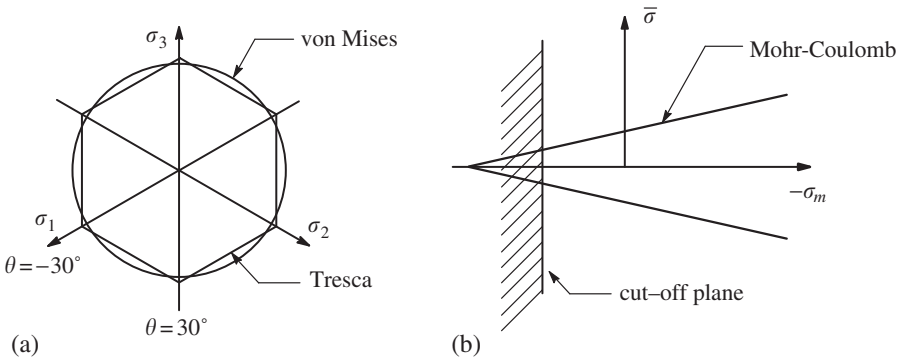


Figure 7. Composite yield criterion examples: (a) Tresca and von Mises; and (b) Mohr–Coulomb and cut-off plane.

Abbo and Sloan [4]. Defining $\sigma_m = \frac{1}{3}\sigma_{ij}$, $s_{ij} = \sigma_{ij} - \delta_{ij}\sigma_m$, and $\bar{\sigma} = (\frac{1}{2}s_{ij}s_{ij})^{1/2}$, a plot of this function in the meridional plane is shown in Figure 6.

Although the focus here is on simple yield functions, the method proposed can also be used for "composite" yield criteria where several surfaces are combined to constrain the stresses at each node of the mesh. These combinations can be different for different parts of the discretized body. Each of the surfaces must be convex and smooth but the resulting composite surface, though convex, is generally non-smooth. Figure 7(a) shows an example of a composite yield function where a conventional (non-smooth) Tresca criterion is combined with a von Mises cylinder to round the corners in the octahedral plane. Another example, Figure 7(b), shows the use of a simple plane to cut the apex off a cone-like yield surface. This type of cut-off is often used for modelling no-tension materials such as rock, and leads to a cup-shaped surface. The incorporation of composite yield functions affects the total computational effort only marginally (unless the number of component functions at each node exceeds about 10) as the matrix representing all the inequalities is still block diagonal and can be inverted quickly in a node-by-node manner.

Provided the stresses vary linearly, the yield condition is satisfied throughout an element if it is satisfied at all its nodes. This implies that the stresses at all N nodes in the finite element model must satisfy the following inequality:

$$f(\sigma_{ij}^l) \leqslant 0, \quad l = 1, \ldots, N$$

Thus, in total, the yield conditions give rise to N non-linear inequality constraints (considering composite yield criteria as one constraint) on the nodal stresses. Because each node is associated with a unique set of stress variables, it follows that each yield inequality is a function of an uncoupled set of stress variables σ_{ij}^l. This feature can be exploited to give a very efficient solution algorithm and will be discussed later in the paper.

2.7. Other inequality constraints

Some problems require additional inequalities to be applied to the unknowns. When optimizing unit weights, for example, it is necessary to impose non-negativity constraints on these variables so as to avoid physically unreasonable solutions. In most common cases these types of constraints are linear and can be included in the final set of inequalities, together with the yield constraints, to give

$$f_j(\mathbf{x}) \leqslant 0, \quad j \in J$$

where $\mathbf{x}$ is the total vector of unknowns and J is the set of all inequalities.

2.8. Assembly of constraint equations

All the steps necessary to formulate the lower bound theorem as an optimization problem have now been covered. The only step remaining is to assemble the constraint matrices and objective function coefficients for the overall mesh.

Using Equations (8), (10), (12) and (19) the various equality constraints may be assembled to give the overall equality constraint matrix according to

$$\mathbf{A} = \sum_{e=1}^{E} \mathbf{A}_{\text{equil}}^e + \sum_{d=1}^{Ds} \mathbf{A}_{\text{equil}}^d + \sum_{b=1}^{Bn} \mathbf{A}_{\text{bound}}^b + \sum_{l=1}^{Ld} \mathbf{A}_{\text{load}}^l$$

where E is the total number of elements, Ds is the total number of discontinuities, Bn is the total number of boundary nodes which are subject to prescribed surface tractions, Ld is the total number of loading constraints, and all coefficients are inserted into the appropriate rows and columns using the usual element assembly rules.

Similarly, the corresponding right-hand side vector $\mathbf{b}$ is assembled according to

$$\mathbf{b} = \sum_{e=1}^{E} \mathbf{b}_{\text{equil}}^e + \sum_{d=1}^{Ds} \mathbf{b}_{\text{equil}}^d + \sum_{b=1}^{Bn} \mathbf{b}_{\text{bound}}^b + \sum_{l=1}^{Ld} \mathbf{b}_{\text{load}}^l$$

Noting that the objective function is linear, the equality constraints are linear, and the yield inequalities are non-linear, the problem of finding a statically admissible stress field which maximizes the collapse load may be stated as

$$\text{maximize} \quad \mathbf{c}^{\mathrm{T}}\mathbf{x} \tag{20}$$

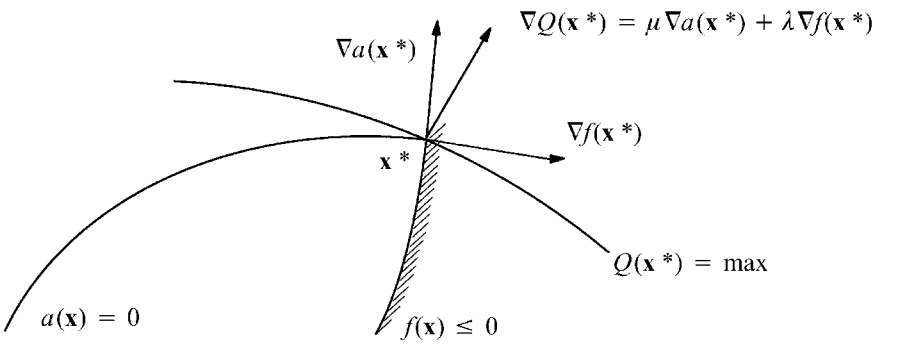


Figure 8. Geometrical interpretation of Kuhn–Tucker optimality conditions.

subject to

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{b} \\ f_j(\mathbf{x}) &\leqslant 0, \quad j \in J \\ \mathbf{x} &\in \mathrm{R}^n \end{aligned}$$

where $\mathbf{c}$ is an n-vector of objective function coefficients, $\mathbf{A}$ is an $m \times n$ matrix of equality constraint coefficients, $f_j(\mathbf{x})$ are yield functions and other inequality constraints, and $\mathbf{x}$ is an n-vector which is to be determined.

3. KUHN–TUCKER OPTIMALITY CONDITIONS

Because the objective function and equality constraints are linear, while the functions f_j are convex, problem (20) is equivalent to the system of Kuhn–Tucker optimality conditions:

$$\begin{aligned} -\mathbf{c} + \mathbf{A}^{\mathrm{T}}\boldsymbol{\mu} + \sum_{j \in J} \lambda_j \nabla f_j(\mathbf{x}^*) &= \mathbf{0} \\ \mathbf{A}\mathbf{x}^* &= \mathbf{b} \\ \lambda_j f_j(\mathbf{x}^*) &= 0, \quad j \in J \\ f_j(\mathbf{x}^*) &\leqslant 0, \quad j \in J \\ \lambda_j &\geqslant 0, \quad j \in J \\ \mathbf{x}^* &\in \mathrm{R}^n \end{aligned} \tag{21}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ are Lagrange multipliers for the equality and inequality constraints, respectively, and $\mathbf{x}^*$ is an optimum point for (20). Geometrically, the Kuhn–Tucker optimality conditions imply that, at an optimum point, the objective function gradient is a linear combination of all equality and active inequality constraint gradients (see Figure 8). Introducing the Lagrange function $L(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda})$ defined by

$$L(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = -\mathbf{c}^{\mathrm{T}}\mathbf{x} + \boldsymbol{\mu}^{\mathrm{T}}(\mathbf{A}\mathbf{x} - \mathbf{b}) + \sum_{j \in J} \lambda_j f_j(\mathbf{x})$$

we can cast the first equation of (21) in the form

$$\nabla_{\mathbf{x}} L(\mathbf{x}^*, \boldsymbol{\mu}, \boldsymbol{\lambda}) = \mathbf{0}$$

Zouain *et al.* [5] proposed a feasible point algorithm for solving limit analysis problems arising from a mixed formulation. This approach obviates the need to linearize the yield constraints and thus can be used for any type of yield function. The solution technique itself is based on the two-stage feasible point algorithm originating from the work of Herskovits [6, 7]. Although the optimization problem considered by these authors is different to the one considered here, their algorithms can be modified to suit our purposes. In brief, the algorithm of Zouain *et al.* [5] starts by using a quasi-Newton iteration formula for the set of all equalities in the optimality conditions. The search direction obtained from each Newton iteration is then deflected by a small amount to preserve feasibility with respect to the inequality constraints. To account for the nature of the optimization problem that is generated by the lower bound formulation, our algorithm uses a new deflection strategy and introduces some key modifications. The new procedure finds an initial feasible point, can deal with arbitrary loading, and permits optimization with respect to the body forces. A detailed description of the two stage quasi-Newton strategy for solving system (21) is given in the next section and Appendix B.

4. TWO STAGE QUASI-NEWTON ALGORITHM

In Newton's method, the non-linear equations at the current point k are linearized and the resulting system of linear equations is solved to obtain a new point $k+1$. The process is repeated until the governing system of non-linear equations is satisfied. Application of this procedure to the equalities of system (21) leads to

$$\nabla_{\mathbf{x}}^2 L(\mathbf{x}^k, \boldsymbol{\mu}^k, \boldsymbol{\lambda}^k)(\mathbf{x}^{k+1} - \mathbf{x}^k) + \mathbf{A}^{\mathrm{T}} \boldsymbol{\mu}^{k+1} + \sum_{j \in J} \lambda_j^{k+1} \nabla f_j(\mathbf{x}^k) = \mathbf{c} \tag{22}$$

$$\mathbf{A}(\mathbf{x}^{k+1} - \mathbf{x}^k) = \mathbf{0} \tag{23}$$

$$\lambda_j^k \nabla f_j(\mathbf{x}^k)(\mathbf{x}^{k+1} - \mathbf{x}^k) + \lambda_j^{k+1} f_j(\mathbf{x}^k) = 0, \quad j \in J \tag{24}$$

where the superscript k denotes the iteration number. Assuming that at point $\mathbf{x}^k$ the multiplier λ_j^k is positive for any $j \in J$ (as will be forced by the updating rule), it is possible to rewrite (22)–(24) in the more compact form

$$\mathbf{H}\mathbf{d} + \mathbf{A}^{\mathrm{T}} \boldsymbol{\mu} + \mathbf{G}^{\mathrm{T}} \boldsymbol{\lambda} = \mathbf{c} \tag{25}$$

$$\mathbf{A}\mathbf{d} = \mathbf{0} \tag{26}$$

$$\mathbf{G}\mathbf{d} + \mathbf{F}\boldsymbol{\lambda} = \mathbf{0} \tag{27}$$

where

$$\mathbf{H} = \nabla_{\mathbf{x}}^2 L(\mathbf{x}^k, \boldsymbol{\mu}^k, \boldsymbol{\lambda}^k)$$

$$\mathbf{A}^{\mathrm{T}} = [\nabla a_1(\mathbf{x}^k) \ldots \nabla a_m(\mathbf{x}^k)]$$
$$\mathbf{G}^{\mathrm{T}} = [\nabla f_1(\mathbf{x}^k) \ldots \nabla f_r(\mathbf{x}^k)]$$
$$\mathbf{F} = \mathrm{diag}(f_j(\mathbf{x}^k)/\lambda_j^k) \tag{28}$$

and

$$\mathbf{d} = \mathbf{x}^{k+1} - \mathbf{x}^k, \quad \boldsymbol{\mu} = \boldsymbol{\mu}^{k+1}, \quad \boldsymbol{\lambda} = \boldsymbol{\lambda}^{k+1}$$

If we denote

$$\mathbf{T} = \begin{bmatrix} \mathbf{H} & \mathbf{A}^{\mathrm{T}} & \mathbf{G}^{\mathrm{T}} \\ \mathbf{A} & \mathbf{0} & \mathbf{0} \\ \mathbf{G} & \mathbf{0} & \mathbf{F} \end{bmatrix}, \quad \mathbf{y} = \begin{Bmatrix} \mathbf{d} \\ \boldsymbol{\mu} \\ \boldsymbol{\lambda} \end{Bmatrix}, \quad \mathbf{v} = \begin{Bmatrix} \mathbf{c} \\ \mathbf{0} \\ \mathbf{0} \end{Bmatrix}$$

then the system defined by Equations (25)–(27) may be rewritten in the even more compact form

$$\mathbf{T}\mathbf{y} = \mathbf{v} \tag{29}$$

Now we have a highly sparse, symmetric system of $n + m + r$ linear equations that can be solved for the unknowns $\mathbf{y}$. For linear finite elements the only non-zero contributions to the matrix $\mathbf{H}$ are the Hessians of the yield constraints which are defined as

$$\mathbf{H} = \sum_{j \in J} \mathbf{H}_j = \sum_{j \in J} \lambda_j^k \nabla^2 f_j(\mathbf{x}^k) \tag{30}$$

If each plastic constraint depends on an uncoupled set of stress variables, then $\mathbf{H}$ is block diagonal and $\mathbf{G}$ also consists of disjoint blocks. The size of each block in $\mathbf{H}$ is equal to the number of stress variables in the corresponding yield constraint, with a maximum of six in the 3D case, and $\mathbf{H}^{-1}$ can be computed very quickly in a node-by-node manner.

For some yield functions, the Hessian of Equation (30) is not a positive definite matrix. In these cases, we can apply a small perturbation $\varepsilon\mathbf{I}$ to restore positive definiteness according to

$$\mathbf{H} = \sum_{j \in J} \lambda_j^k [\nabla^2 f_j(\mathbf{x}^k) + \varepsilon_j \mathbf{I}_j]$$

where $\varepsilon \in [10^{-6}, 10^{-4}]$.

4.1. Stage 1: increment estimate

In the first stage of the algorithm, estimates for $\mathbf{d}$, $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ are found by exploiting the above-mentioned features of the matrices $\mathbf{H}$ and $\mathbf{G}$. From (25) we have

$$\mathbf{d}_0 = \mathbf{H}^{-1}(\mathbf{c} - \mathbf{A}^{\mathrm{T}}\boldsymbol{\mu}_0 - \mathbf{G}^{\mathrm{T}}\boldsymbol{\lambda}_0) \tag{31}$$

where the subscript 0 is introduced to indicate the solution is from stage one of the algorithm. Substituting (31) into (27) gives

$$\boldsymbol{\lambda}_0 = \mathbf{W}^{-1}\mathbf{Q}^{\mathrm{T}}(\mathbf{c} - \mathbf{A}^{\mathrm{T}}\boldsymbol{\mu}_0) \tag{32}$$

where the diagonal matrix $\mathbf{W}$ and the matrix $\mathbf{Q}$ are computed from

$$\mathbf{W} = \mathbf{G}\mathbf{H}^{-1}\mathbf{G}^{\mathrm{T}} - \mathbf{F}$$
$$\mathbf{Q} = \mathbf{H}^{-1}\mathbf{G}^{\mathrm{T}}$$

Next we can eliminate $\boldsymbol{\lambda}_0$ from (31) using (32), which gives

$$\mathbf{d}_0 = \mathbf{D}(\mathbf{c} - \mathbf{A}^{\mathrm{T}}\boldsymbol{\mu}_0) \tag{33}$$

with

$$\mathbf{D} = \mathbf{H}^{-1} - \mathbf{Q}\mathbf{W}^{-1}\mathbf{Q}^{\mathrm{T}}$$

Multiplying both sides of (33) by $\mathbf{A}$ and taking into account (26) we obtain

$$\mathbf{K}\boldsymbol{\mu}_0 = \mathbf{A}\mathbf{D}\mathbf{c} \tag{34}$$

where

$$\mathbf{K} = \mathbf{A}\mathbf{D}\mathbf{A}^{\mathrm{T}}$$

is $m \times m$ symmetric positive semi-definite matrix [5] whose density is roughly double that of $\mathbf{A}$. To start the computational sequence, $\boldsymbol{\mu}_0$ is first found from (34). The quantities $\boldsymbol{\lambda}_0$ and $\mathbf{d}_0$ are then computed, respectively, using (32) and (33).

This computational strategy is, for large three-dimensional problems, up to 10 times faster than solving (29) by direct factorization of the matrix $\mathbf{T}$. It should be noted, however, that the condition number of the matrix $\mathbf{K}$ is equal to the square of the condition number of the matrix $\mathbf{T}$ as the optimum solution is approached. Fortunately, experience suggests that double precision arithmetic is sufficient to get an accurate solution using (34) before the matrix $\mathbf{K}$ becomes extremely ill-conditioned or numerically singular.

4.2. Stage 2: computation of a deflected feasible direction

It is clear from (27) that the vector $\mathbf{d}_0$ is tangential to the active constraints where $f_j(\mathbf{x}^k) = 0$, so we must deflect it to make the search direction feasible. This can be done by setting the right side of every row in (27) to some negative value which is known as the deflection factor. Zouain *et al.* [5] suggested that the same deflection factor, θ, should be used for each yield constraint. This strategy, however, was found to be ineffective when the curvatures of the active constraints differ greatly at the current iterate $\mathbf{x}^k$. With reference to Figure 9, let s_1 denote the step size along the deflected search direction $\mathbf{d}^1$ for a constraint with high curvature, and s_2 denote the step size along the deflected search direction $\mathbf{d}^2$ for a constraint with moderate curvature. Because of the relation $s_1/\|\mathbf{d}^1\| = s_2/\|\mathbf{d}^2\|$, it is clear the step size along $\mathbf{d}^2$ is unnecessarily small because of the constant θ rule. We now describe another technique that works well for all types of yield criteria and has a simple geometrical interpretation.

Consider the component-wise form of Equation (27). Letting $\mathbf{d}_j$ denote the part of the vector $\mathbf{d}$ which corresponds to the set of variables of the function f_j, and using the notation $\bar{\lambda}_j$ instead of λ_j^k, we can rewrite (27) in the form

$$\nabla f_j^{\mathrm{T}}\mathbf{d}_j + (f_j/\bar{\lambda}_j)\lambda_j = 0 \tag{35}$$

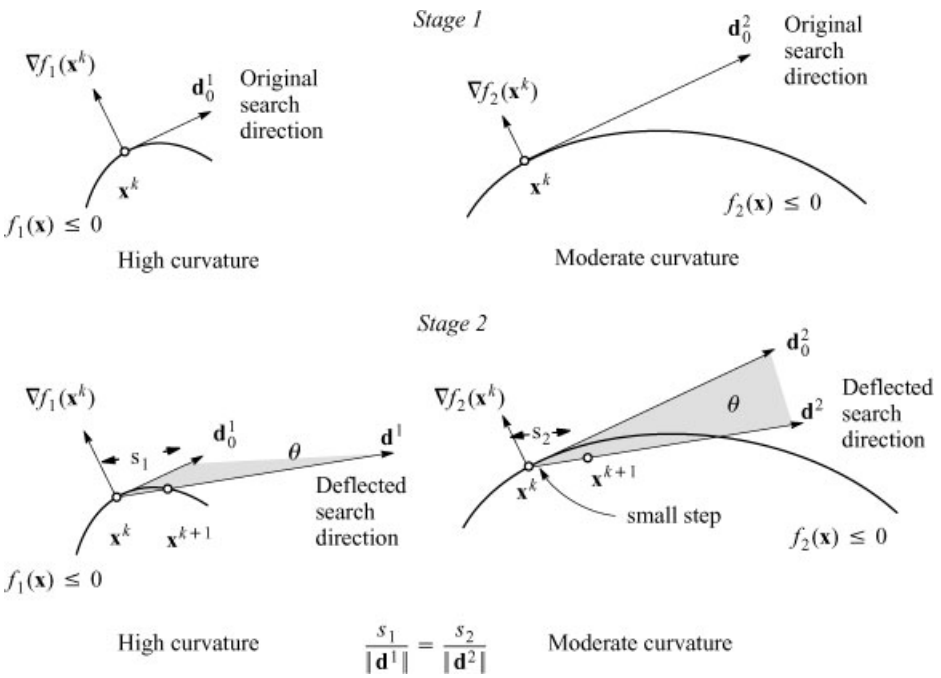


Figure 9. Inefficiency of equal deflection strategy.

Invoking the geometrical definition of the scalar product, this may also be expressed as

$$\|\nabla f_j\| \, \|\mathbf{d}_j\| \cos \phi_j + (f_j/\bar{\lambda}_j)\lambda_j = 0$$

where ϕ_j is the angle between the normal to the jth constraint and the vector $\mathbf{d}_j$. If we now introduce the deflection factor as the product $\|\nabla f_j\| \, \|\mathbf{d}_j\| \theta_{d_j}$, then θ_{d_j} is equal to $\cos \phi_j$ for all active constraints. Using an equal value of θ_d for these cases implies that the angle between the vector $\mathbf{d}_j$ and the tangential plane to the yield surface at the point $\mathbf{x}_j^k$, which we term the deflection angle θ, is identical for all active constraints. For some yield surfaces used in limit analysis, the curvature at a given point $\mathbf{x}_j^k$ is strongly dependent on the direction $\mathbf{d}_j$. Therefore, it is better to make θ proportional not only to the length of the vector $\mathbf{d}_j$, but also to the curvature $\varkappa_j$ of the function f_j in the direction $\mathbf{d}_j$. The latter may be estimated as the norm of the derivative of ∇f_j in the direction $\mathbf{d}_j$ multiplied by R_j, where R_j is the distance from the point $\mathbf{x}_j^k$ to the axis of the yield surface in the octahedral plane. Combining all the above factors leads us to replace (35) by

$$\nabla f_j^{\mathrm{T}} \mathbf{d}_j + (f_j/\bar{\lambda}_j)\lambda_j = -\theta_d \|\nabla f_j\| \, \|\mathbf{d}_{0j}\| \varkappa_j \tag{36}$$

where

$$\theta_d \in (0, 1)$$

$$\varkappa_j = \max(\varkappa_j^0/\varkappa_{\max}, \varkappa_{\min})$$

$$\varkappa_j^0 = \frac{\|\nabla^2 f_j(\mathbf{x}^k)\mathbf{d}_{0j}\|R_j}{\|\nabla f_j\|\,\|\mathbf{d}_{0j}\|}$$

$$\varkappa_{\max} = \max_{j \in J} \varkappa_j^0$$

$$\varkappa_{\min} \in (0, 1)$$

Note that θ_d is equal to the sine of the maximum deflection angle $\theta_{\max}$ and can be related analytically to the increment in the objective function $\mathbf{c}^{\mathrm{T}}\mathbf{d}_0$ (as will be shown later).

Replacing (27) by the matrix form of (36) furnishes the system of equations for computing the new feasible direction $\mathbf{d}$ as

$$\mathbf{H}\mathbf{d} + \mathbf{A}^{\mathrm{T}}\boldsymbol{\mu} + \mathbf{G}^{\mathrm{T}}\boldsymbol{\lambda} = \mathbf{c}$$

$$\mathbf{A}\mathbf{d} = \mathbf{0}$$

$$\mathbf{G}\mathbf{d} + \mathbf{F}\boldsymbol{\lambda} = -\theta_d\mathbf{u}$$

where $\mathbf{u}$ is an r-dimensional vector with components

$$u_j = \|\nabla f_j\|\,\|\mathbf{d}_{0j}\|\varkappa_j$$

Applying the same computational sequence as was implemented for solution of system (25)–(27), we finally have

$$\mathbf{d} = \mathbf{H}^{-1}(\mathbf{c} - \mathbf{A}^{\mathrm{T}}\boldsymbol{\mu} - \mathbf{G}^{\mathrm{T}}\boldsymbol{\lambda}) \tag{37}$$

$$\boldsymbol{\lambda} = \mathbf{W}^{-1}\mathbf{Q}^{\mathrm{T}}(\mathbf{c} - \mathbf{A}^{\mathrm{T}}\boldsymbol{\mu}) + \theta_d\mathbf{W}^{-1}\mathbf{u} \tag{38}$$

$$\mathbf{K}\boldsymbol{\mu} = \mathbf{A}\mathbf{D}\mathbf{c} - \theta_d\mathbf{A}\mathbf{Q}\mathbf{W}^{-1}\mathbf{u}$$

To derive an expression for θ_d, we substitute (38) into (37) to give

$$\mathbf{d} = \mathbf{D}(\mathbf{c} - \mathbf{A}^{\mathrm{T}}\boldsymbol{\mu}) - \theta_d\mathbf{Q}\mathbf{W}^{-1}\mathbf{u} \tag{39}$$

Multiplying both sides of (39) by $(\mathbf{c}^{\mathrm{T}} - \boldsymbol{\mu}_0^{\mathrm{T}}\mathbf{A})$ and using (26), (32) and (33) we obtain

$$\theta_d = (\mathbf{c}^{\mathrm{T}}\mathbf{d}_0 - \mathbf{c}^{\mathrm{T}}\mathbf{d})/\boldsymbol{\lambda}_0^{\mathrm{T}}\mathbf{u}$$

If we impose the condition that the *Stage 1* and *Stage 2* increments of the objective function are related by a fixed parameter $\beta \in (0, 1)$ such that $\mathbf{c}^{\mathrm{T}}\mathbf{d} = \beta\mathbf{c}^{\mathrm{T}}\mathbf{d}_0$, then θ_d can be computed as

$$\theta_d = \mathbf{c}^{\mathrm{T}}\mathbf{d}_0(1 - \beta)/\boldsymbol{\lambda}_0^{\mathrm{T}}\mathbf{u}$$

A typical value for β is 0.7. Another option for computing the feasible search direction in the second stage of the algorithm has been suggested by Borges *et al.* [8]. This is based on a step relaxation and stress scaling technique and is shown in Figure 10. When applied to the lower bound formulation this scheme should, more correctly, be called a step relaxation and variable scaling technique, since the vector of unknowns may include body forces. Inclusion

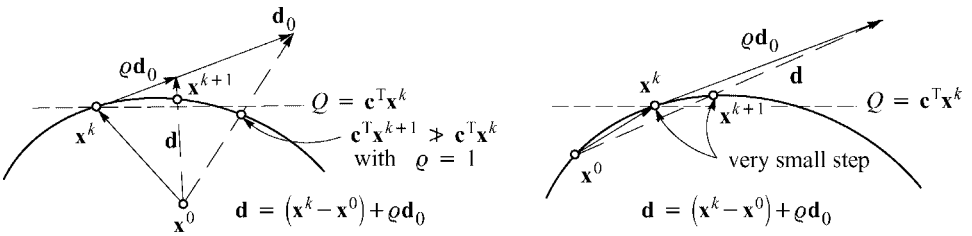


Figure 10. Effect of initial feasible point on efficiency of step relaxation and variable scaling technique.

of body forces as problem variables also means that the scaling should be performed with respect to some feasible point which is not the origin of the variable space. In practice, a good initial feasible point for this procedure can be found by using the solution to the phase 1 optimization problem, as discussed in the next section. As shown in Figure 10, a poor initial feasible point can lead to very small step sizes for some iterations.

The process is started by choosing a step relaxation factor, $\varrho \in (0, 1]$, for the initial search direction $\mathbf{d}_0$. Admissibility with respect to the inequality constraints is then imposed by scaling the vector of unknowns $\mathbf{x}$ according to the rule

$$\mathbf{x}^{k+1} = \mathbf{x}^0 + s((\mathbf{x}^k - \mathbf{x}^0) + \varrho\mathbf{d}_0)$$

where $\mathbf{x}^0$, $\mathbf{x}^k$, $\mathbf{x}^{k+1}$ and s are, respectively, the initial feasible point, the feasible point for the kth iteration, the feasible point for the $(k+1)$th iteration, and a scaling factor. The factor s is computed by taking each of the inequality constraints to be strictly negative according to

$$s = \min_{j \in J} s_j \mid f_j(\mathbf{x}^0 + s_j(\mathbf{x}^0 + \varrho\mathbf{d}_0)) = \delta_f f_j(\mathbf{x}^0) \tag{40}$$

where $\delta_f \in (0, 10^{-2}]$ is a given control parameter. Since each reduction of the relaxation factor ϱ forces the vector $(\mathbf{x}^{k+1} - \mathbf{x}^k)$ closer to the ascent direction $\mathbf{d}_0$, it is always possible to obtain ϱ small enough to guarantee that $\mathbf{c}^T\mathbf{x}^{k+1} > \mathbf{c}^T\mathbf{x}^k$. To determine the optimal value of the factor ϱ, we first start with $\varrho = 1$ and compute s using (40). If this estimate results in $\mathbf{c}^T\mathbf{x}^{k+1} < \mathbf{c}^T\mathbf{x}^k$, then the relaxation factor ϱ is replaced by $\delta_\varrho\varrho$, where $\delta_\varrho \in [0.7, 0.9]$ is a preset parameter, to give a new value of ϱ and the factor s is computed again. This procedure is repeated until we get $\mathbf{c}^T\mathbf{x}^{k+1} \geqslant \mathbf{c}^T\mathbf{x}^k$. Near the optimal solution, the values of ϱ and s will approach unity for all inequality constraints.

4.3. Initialization

So far it has been assumed that the current solution $\mathbf{x}^k$ is a feasible point. Therefore, to start the iterations, we need a procedure which will give us an initial feasible point $\mathbf{x}^0$. First of all $\mathbf{x}^0$ must satisfy

$$\mathbf{A}\mathbf{x}^0 = \mathbf{b} \tag{41}$$

where $\mathbf{A}$ is the matrix of equality constraints and $\mathbf{b}$ is the corresponding vector of right-hand sides. In order to convert (41) to a linear system with a square symmetric coefficient matrix,

it can be augmented by a set of dummy variables $\boldsymbol{\mu}$ to become

$$\begin{bmatrix} \mathbf{I} & \mathbf{A}^{\mathrm{T}} \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{x}^0 \\ \boldsymbol{\mu} \end{Bmatrix} = \begin{Bmatrix} \mathbf{0} \\ \mathbf{b} \end{Bmatrix}$$

Writing the equations in this form allows us to use the same solver as for the main system of Equations (25)–(27). Their solution can be readily found using the relations

$$\mathbf{x}_0 = -\mathbf{A}^{\mathrm{T}}\boldsymbol{\mu}$$

$$\mathbf{A}\mathbf{A}^{\mathrm{T}}\boldsymbol{\mu} = -\mathbf{b}$$

When factorizing the symmetric matrix $\mathbf{A}\mathbf{A}^{\mathrm{T}}$, however, we need to be aware of possible redundancies in the matrix of equalities $\mathbf{A}$. These are detected when a zero (or near zero) occurs on the diagonal of the factored form of $\mathbf{A}\mathbf{A}^{\mathrm{T}}$, and are marked in a single factorization pass. Redundant equalities are deleted from further consideration in the algorithm, except when checking the feasibility of the current solution. If redundancies in the matrix $\mathbf{A}$ are detected, then the offending constraints must be deleted and $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ factorized afresh to obtain values of $\boldsymbol{\mu}$ and the initial feasible point $\mathbf{x}^0$. Once $\mathbf{x}^0$ has been obtained, we then substitute it into the inequalities and compute the scalar

$$\alpha = \max_{j \in J} f_j(\mathbf{x}^0)$$

If $\alpha \leqslant 0$, $\mathbf{x}^0$ satisfies all the inequalities and the initial search direction $\mathbf{d}_0$ can be found by solving

$$\begin{bmatrix} \mathbf{I} & \mathbf{A}^{\mathrm{T}} \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{Bmatrix} \mathbf{d}^0 \\ \boldsymbol{\mu} \end{Bmatrix} = \begin{Bmatrix} \mathbf{c} \\ \mathbf{0} \end{Bmatrix}$$

where $\boldsymbol{\mu}$ are again a set of dummy variables that are introduced for computational convenience. We then set $\tilde{\lambda}_j = 1$ for all $j \in J$ and start the iteration process. If, on the other hand, $\alpha > 0$, this signifies that $\mathbf{x}^0$ does not satisfy all the inequalities and it is necessary to solve the so-called 'phase one' problem defined by

$$\text{minimize } \alpha_{r+1} \tag{42}$$

subject to

$$\begin{aligned} \mathbf{A}_\alpha \mathbf{x}_\alpha &= \mathbf{b} \\ \alpha_{j+1} - \alpha_j &= 0, \quad j \in J \\ f_j(\mathbf{x}_\alpha) - \alpha_j &\leqslant 0, \quad j \in J \\ -\alpha_{r+1} &\leqslant 0 \\ \mathbf{x}_\alpha &\in \mathrm{R}^{n+r+1} \end{aligned}$$

where $\mathbf{x}_\alpha^{\mathrm{T}} = \{\mathbf{x}^{\mathrm{T}}, \alpha_1, \ldots, \alpha_{r+1}\}$ and $\mathbf{A}_\alpha$ is the matrix $\mathbf{A}$ augmented by $r+1$ columns. Here, separate slack variables α_j have been used for each inequality constraint to avoid dense columns in the matrix $\mathbf{D}$. Because of the similarity of problem (42) to problem (20), it can

be solved using the same two-stage algorithm. If the optimal solution to the phase one problem (42) is such that $\alpha = \alpha_1 = \cdots = \alpha_{r+1} = 0$, then $\mathbf{x}^0$ is found by discarding the last $r+1$ entries in $\mathbf{x}_\alpha$. Otherwise, the original lower bound optimization problem has no feasible solution and the process is halted.

4.4. Convergence criterion

At the end of each Stage 1 iteration, various convergence tests are performed on the set of optimality conditions (21). Noting that all constraints of the original optimization problem (20) are satisfied at every step in the solution process, our dimensionless convergence checks are of the form

$$\|\mathbf{c} - \mathbf{A}^{\mathrm{T}}\boldsymbol{\mu}_0 - \mathbf{G}^{\mathrm{T}}\boldsymbol{\lambda}_0\| \leqslant \varepsilon_c \|\mathbf{c}\| \tag{43}$$

$$\lambda_{0j} \geqslant -\varepsilon_\lambda \lambda_{\max}, \quad j \in J \tag{44}$$

$$\lambda_{0j} \leqslant \varepsilon_\lambda \lambda_{\max} \quad \text{if } f_j(\mathbf{x}) < -\varepsilon_f, \quad j \in J \tag{45}$$

where $\lambda_{\max} = \max_{j \in J} \lambda_{0j} | \lambda_{0j} > 0$ and the three different tolerances are typically defined so that $\varepsilon_c, \varepsilon_\lambda, \varepsilon_f \in [10^{-3}, 10^{-2}]$. All conditions (43)–(45) must be satisfied before convergence is deemed to have occurred.

4.5. Line search

As the deflected search direction $\mathbf{d}$ automatically obeys all linear equality constraints, the only remaining requirement is that the resulting solution, $\mathbf{x}$, satisfies all inequality constraints. In our implementation, we prevent any inequality becoming active in a single iteration by using the following expression to determine the maximum step size s:

$$s = \min_{j \in J} s_j \mid f_j(\mathbf{x} + s_j \mathbf{d}) = \delta_f f_j(\mathbf{x})$$

In the above, the parameter δ_f is forced to converge to zero by means of the rule

$$\delta_f = \min[\delta_f^0, \|\mathbf{c}^{\mathrm{T}}\mathbf{d}_0\| / \|\mathbf{c}^{\mathrm{T}}\mathbf{x}\|]$$

where $\delta_f^0 \in (0, 1)$ is a given control parameter. Preventing inequality constraints becoming active in a single iteration gives smoother convergence since it avoids abrupt changes in the number of active constraints.

4.6. Updating

At the conclusion of each iteration, the solution vector $\mathbf{x}$ is updated using the computed search direction $\mathbf{d}$ and the maximum step size s according to

$$\mathbf{x} = \mathbf{x} + s\mathbf{d}$$

Next, the vector of Lagrange multipliers $\boldsymbol{\lambda}$ is updated so that each component is kept strictly positive. This restriction is necessary because each component is used as a denominator in expression (28), and we wish to maintain negative entries on the diagonal of $\mathbf{F}$ so that the

deflection strategy of (36) is able to maintain feasibility. The rule for updating the Lagrange multipliers is

$$\lambda_j = \max[\lambda_{0j}, \delta_\lambda \lambda_{\max}], \quad j \in J$$

with

$$\delta_\lambda = \min[\delta_\lambda^0, \|\mathbf{d}\|/\|\mathbf{x}\|]$$

where $\delta_\lambda^0 \in [10^{-4}, 10^{-2}]$ is a prescribed tolerance. This rule allows the λ_j to converge to a positive value if λ_{0j} is positive, but sets λ_j to a small positive value if λ_{0j} is tending to zero.

5. NUMERICAL RESULTS

We now analyse some classical soil stability problems to demonstrate the accuracy and efficiency of the new lower bound algorithm. The cases considered adopt the Tresca or Mohr–Coulomb yield criteria which simulate, respectively, the behaviour of undrained and drained clays. In the deviatoric plane, both of these models have corners where the gradient with respect to the stresses is singular. Further problems of this type also occur at the apex of the Mohr–Coulomb model. In the present work, all singularities in the Tresca and Mohr–Coulomb surfaces are smoothed using the approximations described in References [4, 9] (see Appendix A). These methods provide a close fit to the parent models, but generate sharp changes in the yield surface curvature which require the new deflection technique described in Section 4.2.

Three different meshes have been generated for each of the problems and these are classified as coarse, medium and fine. All runs incorporate stress discontinuities at every interelement boundary and, where applicable, special extension elements are used to extend the stress field in an infinite half space [10].

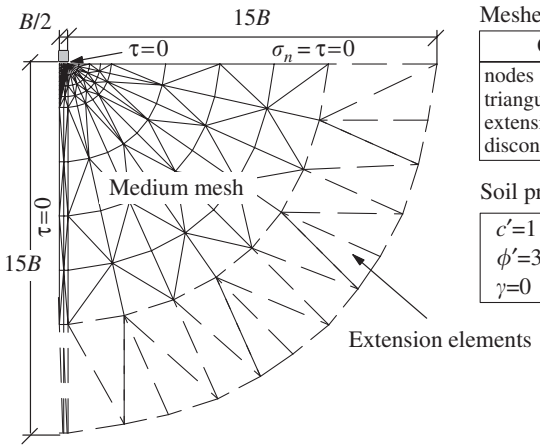
To solve the sparse symmetric system of Equations (34) efficiently, we use the MA27 multifrontal code developed by Duff and Reid [11]. This employs a direct solution method with threshold pivoting to maintain stability and sparsity in the factorization. The CPU times quoted in the tables for the first two problems are for a Dell Precision 220 workstation with a Pentium III 800EB MHz processor operating under Windows 98 second edition. The compiler used was Visual Fortran Standard Edition 6.5.0 with full optimization. The CPU times quoted in the tables for the last three problems are for a SUN Ultra Model 1 170 MHz operating under UNIX with SUN's optimizing FORTRAN compiler.

5.1. Rigid strip footing on cohesive-frictional soil

For a rigid strip footing on a weightless cohesive-frictional soil with no surcharge, the exact collapse pressure is given by the Prandtl [12] solution

$$q/c' = (\exp(\pi \tan \phi') \tan^2(45 + \phi'/2) - 1) \cot \phi'$$

where c' and ϕ' are, respectively, the effective cohesion and the effective friction angle. For a soil with a friction angle of $\phi' = 35^\circ$ this equation gives $q/c' = 46.14$. The stress boundary conditions and material properties used in the analysis, together with one of the meshes, are shown in Figure 11.



Meshes:

Quantity	Coarse	Medium	Fine
nodes	228	712	1452
triangular elements	66	220	452
extension elements	8	14	26
discontinuities	101	335	697

Soil properties:

c'=1
ϕ'=35°
γ=0

Figure 11. Lower bound mesh for strip footing problem.

Table I. Lower bounds for smooth strip footing on weightless cohesive-frictional soil ($c' = 1,\ \phi' = 35°,\ \gamma = 0$).

	Linear programming						Non-linear programming	
	Simplex (NSID = 24)		Active set (NSID = 24)		Interior-point (NSID = 24)		Present ($\varepsilon = 0.01$)	
Mesh	q/c' (error*%)	CPU (s) (iterations)	q/c' (error*%)	CPU (s) (iterations)	q/c' (error*%)	CPU (s) (iterations)	q/c' (error*%)	CPU (s) (iterations)
Coarse	37.791 (−18.1)	4.23 (926)	37.791 (−18.1)	1.71 (327)	37.791 (−18.1)	3.52 (27)	38.685 (−16.2)	0.60 (29)
Medium	43.032 (−6.7)	77.8 (4019)	43.032 (−6.7)	33.8 (1928)	43.032 (−6.7)	15.9 (42)	44.246 (−4.1)	2.3 (30)
Fine	44.064 (−4.5)	473 (10 234)	44.064 (−4.5)	283 (5554)	44.063 (−4.5)	25.2 (29)	45.568 (−1.2)	5.0 (29)

*With respect to exact Prandtl [12] solution of $q/c'_{\text{exact}} = 46.14$.

The results presented in Table I compare the performance of the non-linear two-stage algorithm (NLP) with that of Sloan [2], where the resulting linear program is solved using the active set method [13], the latest simplex method from the IBM optimization solutions library (OSL), and the latest interior-point method from the OSL. The new algorithm demonstrates fast convergence to the optimum solution and, crucially, the number of iterations required does not grow rapidly with the problem size. For the coarsest mesh, the non-linear formulation is nearly three times faster than the active set linear programming formulation and gives a lower bound limit pressure which is 2 per cent better. For the finest mesh the speed advantage of the non-linear procedure with respect to the active set technique is more dramatic, with a 57-fold reduction in the CPU time. In this case, the lower bound limit pressure is 1.2 per cent below the exact limit pressure and the analysis uses just 5 s of CPU time. Because the

Table II. Lower bounds for smooth strip footing problem with different convergence tolerances ($\varepsilon = \varepsilon_c = \varepsilon_\lambda = \varepsilon_f$).

Linear programming								Non-linear programming			
Active set				Interior-point				Present			
CPU (s)	Iter.	NSID	q/c'	CPU (s)	Iter.	NSID	q/c'	CPU (s)	Iter.	ε	q/c'
39	2140	28	43.280	17	39	28	43.280	1.87	20	0.1	43.347
127	4116	90	44.184	187	61	90	44.182*	2.21	27	0.05	44.185
536	7642	250	44.260	†	†	†	†	2.31	30	0.01	44.246
1040	10 278	370	44.265	†	†	†	†	3.02	36	0.005	44.266
2373	14 638	500	44.268	†	†	†	†	3.41	41	0.001	44.269

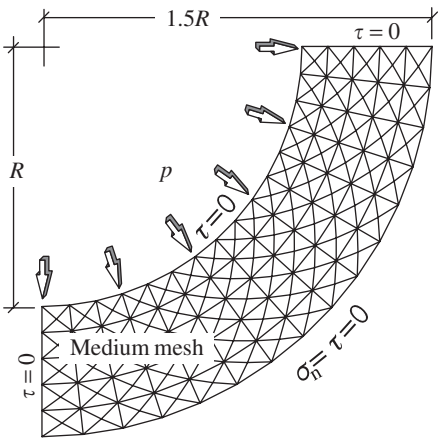
*Solution exceeds feasibility tolerance of 10^{-5}.
†Memory requirements exceed the RAM size of 256 Mb.

number of iterations with the new algorithm is essentially constant for all cases, its CPU time grows almost linearly with the problem size. This is in contrast to the simplex and active set linear programming formulations, whose iterations and CPU time grow at a much faster rate. Relative to these solvers, the CPU time savings from the non-linear method become larger as the problem size increases.

The performance of the OSL simplex method is considerably poorer than that of the active set technique, typically requiring about three times as many iterations and more than double the CPU time. Similar statistics have been reported by Sloan [13] for a simplex version of the active set method. The most competitive linear programming method uses the OSL interior-point solver. Like the proposed NLP approach, the iteration count for this scheme is largely independent of the problem size, but it is at least 5 times slower when used with a 24-sided yield surface linearization. Moreover, its accuracy is substantially less. A further disadvantage of the OSL interior-point solver is that it demands much more storage than any of the other codes tested. For the cases presented in Table I, it required at least four times the memory of the NLP formulation.

The tests in Table I were all run with the convergence tolerances (defined in Section 4.4) set to $\varepsilon_c = \varepsilon_\lambda = \varepsilon_f = \varepsilon = 0.01$. The influence of changing these values, for the 'medium' mesh illustrated in Figure 11, is shown in Table II. Clearly a uniform tolerance value of 0.01 is sufficiently accurate for practical calculations with the non-linear algorithm. Indeed, tightening the tolerances from 0.1 to 0.001 affects the collapse pressure error only marginally, reducing it from -6.1 to -4.1 per cent.

For the linear programming formulations, at least 24 sides (NSID) need to be used in the yield surface linearization, but this value may increase with increasing friction angles. Increasing NSID, however, dramatically increases the solution cost. To obtain a solution with the same accuracy as the non-linear programming method with $\varepsilon = 0.01$, the active set linear programming formulation requires 250 sides in the yield surface linearization and gives a 230-fold increase in CPU time. As expected, the linear and non-linear programming methods give the same collapse pressure when the former is used with an accurate linearization and the latter is used with stringent convergence tolerances. When it was used with a finer yield surface linearization to get the same accuracy as the NLP formulation, the OSL interior-point technique was found to require excessive amounts of memory. Indeed, runs with



Meshes:

Quantity	Coarse	Medium	Fine
nodes	36	900	3600
triangular elements	12	300	1200
discontinuities	14	430	1760

Soil properties:

c'=1
ϕ'=30°
γ=0
p_0=0

Figure 12. Lower bound mesh for expansion of thick cylinder.

more than 90 yield surface sides had to be abandoned as the system started to use virtual memory.

For this particular problem, the conventional linear programming formulations of the lower bound theorem cannot compete with the proposed NLP formulation in terms of efficiency. On a philosophical note, the use of yield surface linearization with interior-point solvers is unappealing as what we are doing is taking a non-linear problem, transforming it to a linear one, and then solving it using a non-linear method. It is conceptually and practically much simpler to solve the original non-linear problem directly.

5.2. *Thick cylinder expansion in cohesive-frictional soil*

The exact solution for a thick cylinder of cohesive-frictional soil subject to a uniform internal pressure has been obtained by Yu [14] and is given by the formula

$$p/c' = \frac{Y + (\alpha - 1)p_0}{c'(\alpha - 1)} \left((b/a)^{(\alpha-1)/\alpha} - 1 \right) + p_0/c'$$

where p is the collapse pressure, p_0 is the initial hydrostatic pressure acting throughout the soil, a and b are the inner and outer radii of the cylinder, and $Y = 2c' \cos\phi'/(1 - \sin\phi')$ and $\alpha = \tan^2(45 + \phi'/2)$ are material constants. For the example considered here with $p_0 = 0$, $b/a = 1.5$, $c' = 1$ and $\phi' = 30°$, the exact collapse pressure is $p = 0.5376$. One of the meshes used for the lower bound analyses is shown in Figure 12, together with the assumed loading and boundary conditions. Because of symmetry, only a 90° sector of the cylinder is discretized.

The results presented for the three different meshes, shown in Table III, demonstrate similar trends to the footing problem in regard to the performance of the non-linear and linear programming techniques.

Since the non-linear programming formulation needs, respectively, just 22 and 26 iterations to obtain the solution for the medium and fine meshes, this suggests that its iteration counts are again largely independent of the problem size (the low iteration counts for the coarse

Table III. Lower bounds for expansion of thick cylinder of cohesive-frictional soil ($p_0 = 0,\ b/a = 1.5,\ c' = 1,\ \phi' = 30^\circ$).

	Linear programming						Non-linear programming	
	Simplex (NSID = 36)		Active set (NSID = 36)		Interior-point (NSID = 36)		Present ($\varepsilon = 0.001$)	
Mesh	p/c' (error*%)	CPU (s) (iterations)	p/c' (error*%)	CPU (s) (iterations)	p/c' (error*%)	CPU (s) (iterations)	p/c' (error*%)	CPU (s) (iterations)
Coarse	0.3781 (−29.7)	0.17 (106)	0.3781 (−29.7)	0.06 (4)	0.3781 (−29.7)	1.15 (14)	0.5067 (−5.70)	0.10 (5)
Medium	0.5269 (−2.0)	148 (4031)	0.5269 (−2.0)	52.2 (1577)	0.5269 (−2.0)	20.6 (23)	0.5360 (−0.30)	2.47 (22)
Fine	0.5336 (−0.7)	5141 (22 755)	0.5336 (−0.7)	4878 (11 276)	0.5336 (−0.7)	98.9 (26)	0.5368 (−0.14)	11.3 (26)

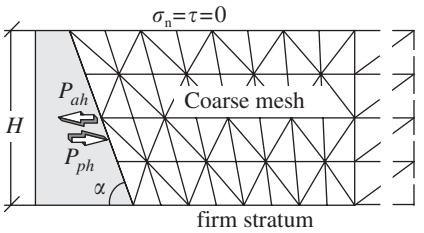
*With respect to exact Yu [14] solution of $p_{\text{exact}} = 0.5376$.

mesh are atypical). As a result of this characteristic, the non-linear programming procedure is very efficient and, for the fine mesh, takes only 11.3 s of CPU time to obtain a lower bound with −0.14 per cent error. In contrast, the iterations required by the active set and simplex linear programming formulations grow rapidly as the mesh is refined. Indeed, for the fine mesh, these procedures require more than 11 000 iterations and a CPU time which is more than 450 times greater than that of the non-linear programming method. The performance of the interior-point scheme is similar to that observed for the strip footing case, but it is now 9 times slower than the NLP method as the number of sides in the linearized surface has been increased to 36.

In this example, the linear and non-linear programming procedures do not give the same result when the yield surface is linearized accurately with the former. This is because the new lower bound formulation permits the orientation of the surface tractions to vary over each segment of the inner and outer boundaries of the cylinder, even though the geometry is assumed to be piecewise linear. The linear programming formulation developed by Sloan [2] does not incorporate this important modification, and assumes that the orientation of the surface tractions is constant over each boundary segment. The advantage of this refinement for problems with curved boundaries is particularly evident in the results for the coarse mesh, where the error in the non-linear programming lower bound is roughly five times smaller than the error in the linear programming lower bound.

5.3. *Retaining wall in frictional soil*

The lateral earth pressure imposed on a sloping retaining wall by a frictional soil is a classical stability problem in soil mechanics and was first considered by Coulomb [15]. Much later, Sokolovskii [16] employed the slip-line method to obtain accurate approximate solutions, while Chen [17] derived rigorous upper bounds using limit analysis. The geometry and stress boundary conditions assumed for the various analyses, together with a diagram of one of the lower bound meshes, are shown in Figure 13. In all cases the surface of the backfill is taken as horizontal, while the wall is assumed to be rigid, perfectly smooth and inclined at an angle $\alpha = 70^\circ$ to the horizontal. Following conventional practice, it is convenient to define the thrust



Meshes:

Quantity	Coarse	Medium	Fine
nodes	208	800	3136
triangular elements	64	256	1024
extension elements	4	8	16
discontinuities	91	375	1519

Soil properties:

$c'=0,\ \phi'=30^\circ,\ \gamma=1$

Figure 13. Lower bound mesh for retaining wall problem.

Table IV. Active and passive horizontal earth pressure coefficients for various methods ($c'=0,\ \phi'=30^\circ,\ \gamma=1,\ a=70^\circ$).

Method	K_{ah}	K_{ph}
Lower bound	0.47	2.13
Coulomb	0.47	2.14
Sokolovskii	0.49	2.03
Chen	0.47	2.13

acting on the wall, P, in terms of an earth pressure coefficient, K, and the unit weight, γ, according to

$$P=\tfrac{1}{2}\gamma H^2 K \tag{46}$$

To distinguish between the active and passive loading cases in (46), the subscripts a and p are used for both P and K. The additional subscripts h and v denote the horizontal and vertical components of these quantities.

Table IV compares the K_{ah} and K_{ph} values obtained by the new lower bound formulation with the limit equilibrium solutions of Coulomb [15], the slip-line solutions of Sokolovskii [16], and upper bound limit analysis solutions of Chen [17]. The lower bound results were obtained using the finest mesh with 3136 nodes. It is worth remarking here that Sokolovskii's solutions are neither upper bounds nor lower bounds. The results shown in Table IV suggest that the lower bound estimates of the active and passive thrusts are identical to Chen's upper bound solutions.

The results presented in Table V compare the accuracy and performance of Sloan's active set linear programming formulation [2, 13] with the new non-linear programming formulation for passive loading of the wall and a variety of meshes. Both methods give estimates that are close to Coulomb's exact values, with differences of just -0.7 and -0.28 per cent, respectively, for the analyses with the finest mesh. The non-linear programming scheme, however, is over 40 times faster for these runs and its iteration count again grows slowly as the mesh is refined. In this example, the linear programming formulation is more efficient than its non-linear counterpart for the coarsest mesh.

The classical problem depicted in Figure 13 has also been used to investigate the efficiency of various deflection strategies for maintaining feasibility in the non-linear programming solver. These strategies have been discussed in detail in Section 4.2 and have

Table V. Lower bounds for passive horizontal earth pressure coefficient ($c' = 0,\ \phi' = 30^\circ,\ \gamma = 1,\ \alpha = 70^\circ$).

	Active set linear programming, NSID = 24				Non-linear programming $\varepsilon = 0.01$				Ratio of LP/NLP values		
Mesh	K_{ph}	Diff.* (%)	CPU (s)	Iter.	K_{ph}	Diff.* (%)	CPU (s)	Iter.	K_{ph}	CPU	Iter.
Coarse	2.103	−1.7	1.68	128	2.112	−1.3	2.55	40	0.996	0.66	3.2
Medium	2.122	−0.84	126	2421	2.130	−0.47	15.3	50	0.996	8.24	48
Fine	2.125	−0.70	4651	12 613	2.134	−0.28	106	58	0.996	44	217

*With respect to Coulomb [15] solution of $K_{ph} = 2.14$.

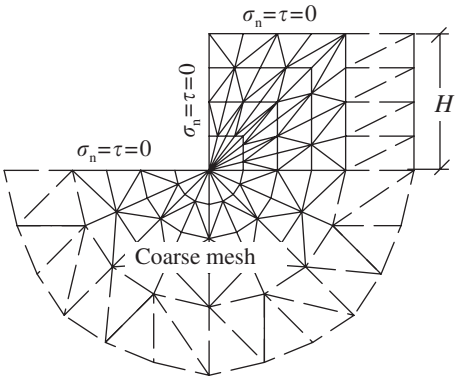
Table VI. Effect of different deflection strategies on performance of lower bound scheme ($c' = 0, \phi' = 30^\circ, \gamma = 1, \varepsilon = \varepsilon_c = \varepsilon_\lambda = \varepsilon_f = 0.01$).

	Equal deflection angle [5]			Scaling and relaxation [8]			Proportional deflection angle		
Mesh	K_{ph}	CPU (s)	Iter.	K_{ph}	CPU (s)	Iter.	K_{ph}	CPU (s)	Iter.
Coarse	1.00*	12.8	200† (168‡)	2.108	4.04	62 (50‡)	2.112	2.55	40 (21‡)
Medium	1.00*	60.7	200† (128‡)	2.128	24.2	75 (56‡)	2.130	15.3	50 (26‡)
Fine	1.00*	342	200† (97‡)	2.132	304	180 (68‡)	2.134	106	58 (30‡)

*Convergence not achieved, algorithm idles in vicinity of initial point.
†Maximum number of iterations exceeded.
‡Phase one iterations to obtain initial feasible point.

a marked effect on the speed of the overall solution process. Results for the equal deflection scheme of Zouain *et al*. [5], the scale and relaxation scheme of Borges *et al.* [8], and the new scheme, which uses a deflection angle that is proportional to the local curvature of the yield surface, are shown in Table VI.

For passive loading of a purely frictional soil by a smooth wall, a number of nodes were found to be unstressed and cluster at the apex of the rounded Mohr–Coulomb yield surface. Under these conditions, the advantage of using a deflection angle that is proportional to the local curvature of the yield function is very pronounced. Indeed, for the finest mesh, this method is almost three times faster than the scaling and relaxation strategy of Borges *et al.* [8]. The latter scheme suffers from the shortcoming that it is rather sensitive to the location of the origin used for scaling. In the original algorithm described by Borges *et al.* [8], no allowance was made for material weight and the stress state $\boldsymbol{\sigma} = \mathbf{0}$ was adopted as the initial point and the scaling origin. For problems with soil weight, this stress state is no longer feasible and cannot be used in such a way. This implies that a phase one algorithm must be invoked to find the initial feasible point. However, even if we can start the phase one procedure at a point which is well inside the feasible region, so that the scaling and relaxation schemes perform effectively, the starting point for the phase two (main) optimization problem will often be positioned close to the surface bounded by the inequality constraints. As highlighted



Meshes

Quantity	Coarse	Medium	Fine
nodes	396	882	3492
triangular elements	112	264	1104
extension elements	16	24	48
discontinuities	183	419	1703

Soil properties

$c_u=1$
$\phi_u=0°$

Figure 14. Lower bound mesh for vertical cut problem.

in Figure 10, this is not a good place for the scaling origin as it results in very small steps in the iteration process.

The results given in Table VI confirm the benefits of making the deflection angle proportional to the local curvature of the yield surface. This procedure is most effective in dealing with the problems caused by cone-shaped yield surfaces, zero cohesion, zero stress boundary conditions, and non-zero soil weight. It also accommodates extension elements, where at least one of the nodal stress states is always close to the apex of the yield cone.

5.4. Critical height of an unsupported vertical cut

All the problems considered so far have been concerned with the optimization of surface tractions applied along a specified part of the domain boundary. We now analyse the case of a vertical cut in a purely cohesive soil, where the vertical body force (unit weight) γ is optimized for a given cohesion c_u and cut height H. Since the stability of this problem is governed by the stability number $N_s = H\gamma/c_u$, the analysis may also be interpreted as finding the maximum height of the cut for a given unit weight and undrained cohesion. One of the three meshes used to analyse the cut is shown in Figure 14, together with the stress boundary conditions. Although this is an important practical problem for construction in undrained clays, its exact solution remains unknown. To date, the best upper and lower bounds on the stability number for the vertical cut are, respectively, $N_s^+ = 3.785864$ and $N_s^- = 3.76037$, and were obtained by Pastor *et al.* [18] using a simplex linear programming formulation with the optimization subroutine library (IBM 1992). As these authors reported their CPU times for the same machine that we used, it is interesting to compare the performance of their linear programming formulation against the performance of our new non-linear programming formulation.

According to Pastor *et al.* [18], their simplex linear programming formulation required about 3 weeks ($\approx$30 000 min) of CPU time to solve for a mesh with 3232 triangular elements. In comparison, our non-linear programming scheme required only 4 min of CPU time for a mesh with 2880 elements (not shown here) to give a slightly better lower bound of $N_s^- = 3.763$. An additional run, with a mesh of 6400 elements arranged in the same pattern as Figure 14, furnished the even better result of $N_s^- = 3.772$ at a CPU cost of 17 min.

Table VII. Results for vertical cut in purely cohesive soil ($c_u = 1, \phi_u = 0°$).

	Active set linear programming, NSID = 24				Non-linear programming $\varepsilon = 0.01$				Ratio of LP/NLP values		
Mesh	N_s	Error* (%)	CPU (s)	Iter.	N_s	Error* (%)	CPU (s)	Iter.	N_s	CPU	Iter.
Coarse	3.52	−7.0	14.5	487	3.54	−6.5	2.30	18	0.99	6.3	27.1
Medium	3.62	−4.4	122	1506	3.64	−3.9	6.42	20	0.99	19	75.3
Fine	3.71	−2.0	4699	10 513	3.73	−1.5	51.0	25	0.99	92	421

*With respect to the Pastor *et al.* [18] upper bound of $N_s^+ = 3.785864$.

This solution would now appear to be the best known lower bound for the vertical cut problem, and highlights the outstanding speed and accuracy advantages to be gained from the non-linear programming formulation.

The results for the coarse, medium and fine meshes, shown in Table VII, again compare the performance of Sloan's active set linear programming method [2, 13] with the new non-linear programming method. As in all previous examples, the iteration counts for the linear programming formulation grow rapidly as the mesh is refined, while the iteration counts for the non-linear programming method are essentially constant. For the finest mesh, these two solution schemes give lower bounds of $N_s^- = 3.71$ and 3.73, respectively, but the non-linear programming method is over 90 times faster. The best lower bound of $N_s^- = 3.73$ differs by just 1.5 per cent from the Pastor *et al.* [18] upper bound of $N_s^+ = 3.785864$, and uses less than a minute of CPU time. In this example, the slight discrepancy between the linear programming and non-linear programming solutions is caused by the yield surface linearization used in the former.

5.5. *Circular footing on weightless cohesive-frictional soil*

The bearing capacity q of a smooth, rigid circular footing resting on a semi-infinite half-space of Mohr–Coulomb soil has been derived by Cox *et al.* [19]. This problem, defined in Figure 15, is difficult to solve because the stress distribution across the footing is not known *a priori* and there is a stress singularity at its edge. Moreover, the computed stress field must be extended throughout the unbounded domain to guarantee that the solution is a rigorous lower bound.

Cox *et al.* [19] derived their solution using the slip-line method and invoked the Haar–Von Karman hypothesis [20] to resolve the intermediate principal stress. This slip-line solution is an upper bound on the actual collapse pressure, as a kinematically admissible velocity field can be associated with the partial stress field in a bounded region beneath the footing. Cox *et al.* [19] have also shown that their partial stress field can be extended throughout the rest of the body without violating the yield condition. Provided the Haar-Von Karman hypothesis is valid both inside and outside the plastic region, this means that their slip-line solution is also a lower bound for the average bearing capacity pressure and is, therefore, the exact solution.

The results presented in Table VIII compare the average footing pressure q and the maximum footing pressure $q_{\max}$ obtained from the lower bound method with those computed by Cox *et al.* [19]. The greatest error in these two quantities is around 5 and 12 per cent,

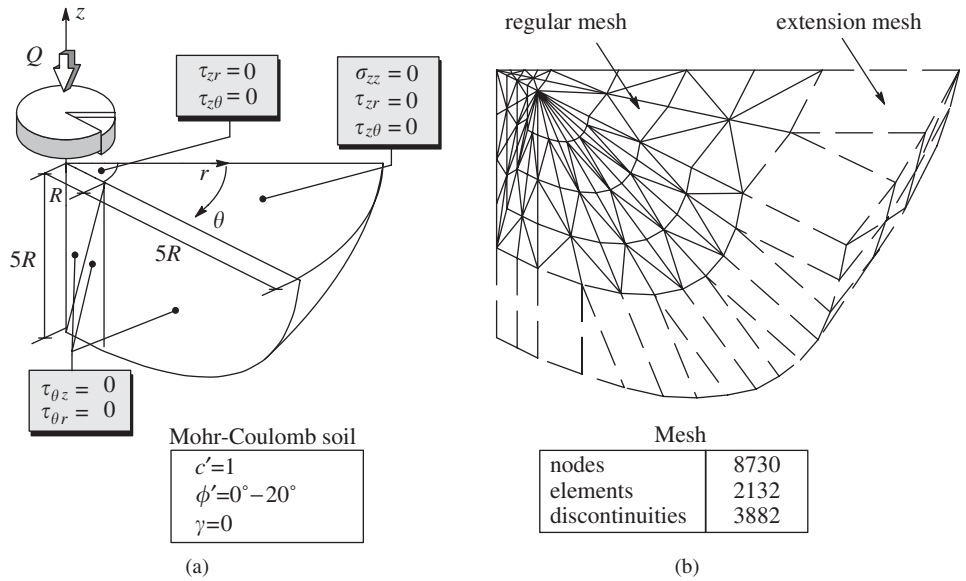


Figure 15. Smooth rigid circular footing on a cohesive-frictional soil: (a) general view with applied boundary conditions; and (b) generated mesh.

Table VIII. Lower bounds for bearing capacity of smooth circular footing on weightless cohesive-frictional soil ($c'=1,\ \phi'=0°-20°, \gamma=0$).

	Cox *et al.* [19]		Lower bound				Difference (%)	
$\phi'(°)$	q/c'	q_{max}/c'	q/c'	q_{max}/c'	Iter.	CPU (s)	q/c'	q_{max}/c'
0	5.69	7.1	5.54	6.55	40	2930	−2.6	−7.7
5	7.44	9.7	7.20	9.04	42	3061	−3.2	−6.8
10	9.98	14.0	9.61	12.90	31	2280	−3.7	−7.9
15	13.90	22.0	13.30	19.46	49	3522	−4.3	−12.0
20	20.10	34.0	19.06	29.87	45	3270	−5.2	−12.1

respectively, and occurs for a friction angle of $\phi'=20°$. Although acceptable, these errors are somewhat greater than expected. The discrepancy observed between the two solutions could be attributable to the use of an insufficiently fine discretization, though a detailed trial-and-error process was employed to try and limit this effect. Another possible minor source of error could be due to the approximations that were used to round the yield surface corners in the deviatoric plane [9].

The mesh used to generate the results in Table VIII is the largest so far and has 8730 nodes, 2132 elements, and 3882 discontinuities. With this grid, and the tolerances set to $\varepsilon_c=\varepsilon_\lambda=\varepsilon_f=10^{-2}, \delta_\lambda^0=5\times10^{-3}$ and $\delta_f^0=10^{-3}$, the lower bound scheme needed an average of 41 iterations and 3013 s of CPU time. Despite this problem being larger than those considered previously, the round-off error in the solution was estimated to be less than 10^{-4}.

The normal pressure distributions over the circular footing, for the cases of $\phi'=0°$ and 20°, are plotted in Figure 16. Both of the lower bound curves mimic the slip-line distributions,

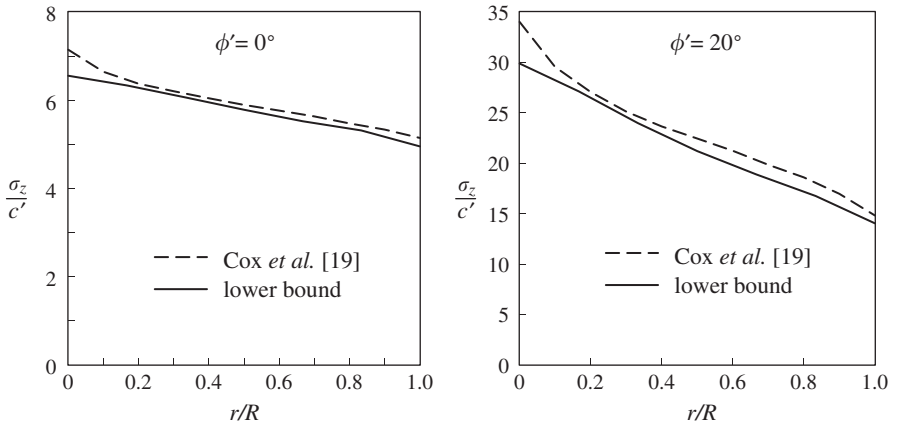


Figure 16. Normal pressure distribution across smooth rigid circular footing on weightless cohesive-frictional soil.

except for small values of r/R, and always lie below them. The maximum difference between the two solutions occurs at the centre of the footing and is equal to 7.7 per cent for $\phi' = 0°$ and 12.1 per cent for $\phi' = 20°$.

6. CONCLUSIONS

A new algorithm for performing lower bound limit analysis in two and three-dimensions has been presented. Numerical results show that the new approach provides accurate solutions for a broad range of stability problems and is vastly superior to a commonly used linear programming formulation, especially for large-scale applications.

APPENDIX A: 2D LOWER BOUND DISCRETIZATION PROCEDURE

A.1. Correspondence between D-dimensional and traditional 2D notation

Co-ordinates: global—$x_1 \to x$, $x_2 \to y$; local—$x_1' \to x'$, $x_2' \to y'$
Stresses: global—$\sigma_{11} \to \sigma_x$, $\sigma_{22} \to \sigma_y$, $\sigma_{12} \to \tau_{xy}$; local—$\sigma_{11}' \to \sigma_n$, $\sigma_{12}' \to \tau$
Body forces: optimized—$h_1 \to h_x$, $h_2 \to h_y$; prescribed—$g_1 \to g_x$, $g_2 \to g_y$
Dimensions: element volume $V \to$ element area A;
element side area $\Delta \to$ edge length L

A.2. Linear finite elements

The stresses vary throughout an element according to

$$\sigma_x = \sum_{l=1}^{3} N_l \sigma_x^l, \quad \sigma_y = \sum_{l=1}^{3} N_l \sigma_y^l, \quad \tau_{xy} = \sum_{l=1}^{3} N_l \tau_{xy}^l, \quad \text{here } N_l = \frac{a_l + b_l x + c_l y}{2A} \tag{A1}$$

and

$$a_1 = \det\begin{vmatrix} x_2 & y_2 \\ x_3 & y_3 \end{vmatrix}, \quad b_1 = -\det\begin{vmatrix} 1 & y_2 \\ 1 & y_3 \end{vmatrix}, \quad c_1 = \det\begin{vmatrix} 1 & x_2 \\ 1 & x_3 \end{vmatrix}; \quad 2A = \det\begin{vmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{vmatrix}$$

The coefficients for the other nodes are defined by cyclic interchange of the subscripts in the order $1, 2, 3$.

A.3. Element equilibrium

$$\frac{\partial \sigma_x}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + h_x = -g_x, \quad \frac{\partial \sigma_y}{\partial y} + \frac{\partial \tau_{xy}}{\partial x} + h_y = -g_y \tag{A2}$$

Substituting (A1) into (A2) leads to

$$\mathbf{A}^e_{\text{equil}} \mathbf{x}^e = \mathbf{b}^e_{\text{equil}} \tag{A3}$$

where

$$\mathbf{A}^e_{\text{equil}} = \frac{1}{2A^e}\begin{bmatrix} b_1 & 0 & c_1 & b_2 & 0 & c_2 & b_3 & 0 & c_3 \\ 0 & c_1 & b_1 & 0 & c_2 & b_2 & 0 & c_3 & b_3 \end{bmatrix}$$

$$\mathbf{x}^e = \{\sigma_x^1 \ \sigma_y^1 \ \tau_{xy}^1 \ \sigma_x^2 \ \sigma_y^2 \ \tau_{xy}^2 \ \sigma_x^3 \ \sigma_y^3 \ \tau_{xy}^3 \ h_x \ h_y\}^{\mathrm{T}}, \quad \mathbf{b}^e_{\text{equil}} = \{-g_x \ -g_y\}^{\mathrm{T}}$$

A.4. Discontinuity equilibrium

$$\begin{Bmatrix} x' \\ y' \end{Bmatrix} = \begin{bmatrix} \beta_{11} & \beta_{12} \\ \beta_{21} & \beta_{22} \end{bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix}$$

then

$$\begin{Bmatrix} \sigma_n \\ \tau \end{Bmatrix} = \left\{ \{\beta_{11} \ \beta_{12}\} \begin{bmatrix} \sigma_x & \tau_{xy} \\ \tau_{xy} & \sigma_y \end{bmatrix} \begin{bmatrix} \beta_{11} & \beta_{12} \\ \beta_{21} & \beta_{22} \end{bmatrix} \right\}^{\mathrm{T}}$$

or

$$\begin{Bmatrix} \sigma_n \\ \tau \end{Bmatrix} = \begin{bmatrix} \beta_{11}\beta_{11} & \beta_{12}\beta_{21} & \beta_{11}\beta_{21} + \beta_{12}\beta_{11} \\ \beta_{11}\beta_{12} & \beta_{12}\beta_{22} & \beta_{11}\beta_{22} + \beta_{12}\beta_{12} \end{bmatrix} \{\sigma_x \ \sigma_y \ \tau_{xy}\}^{\mathrm{T}}$$

and the discontinuity constraints (9) take the form

$$\mathbf{A}^d_{\text{equil}} \boldsymbol{\sigma}^d = \mathbf{b}^d_{\text{equil}} \tag{A4}$$

where

$$\mathbf{A}^d_{\text{equil}} = \begin{bmatrix} \mathbf{B} & -\mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B} & -\mathbf{B} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \beta_{11}\beta_{11} & \beta_{12}\beta_{21} & \beta_{11}\beta_{21} + \beta_{12}\beta_{11} \\ \beta_{11}\beta_{12} & \beta_{12}\beta_{22} & \beta_{11}\beta_{22} + \beta_{12}\beta_{12} \end{bmatrix}$$

$$\boldsymbol{\sigma}^d = \{\sigma_x^{1e_a} \ \sigma_y^{1e_a} \ \tau_{xy}^{1e_a} \ \sigma_x^{2e_b} \ \sigma_y^{2e_b} \ \tau_{xy}^{2e_b} \ \sigma_x^{3e_a} \ \sigma_y^{3e_a} \ \tau_{xy}^{3e_a} \ \sigma_x^{4e_b} \ \sigma_y^{4e_b} \ \tau_{xy}^{4e_b}\}^{\mathrm{T}}$$

$$\mathbf{b}^d_{\text{equil}} = \{0 \ 0 \ 0 \ 0\}^{\mathrm{T}}$$

A.5. Boundary conditions

Let the prescribed normal and shear stresses be $\sigma_n^1 = t_1^1$, $\tau^1 = t_2^1$ and $\sigma_n^2 = t_1^2$, $\tau^2 = t_2^2$ at nodes 1 and 2 of a boundary segment. In matrix form, these constraints read as

$$\mathbf{A}^b_{\text{bound}} \boldsymbol{\sigma}^b = \mathbf{b}^b_{\text{bound}} \tag{A5}$$

where

$$\mathbf{A}^b_{\text{bound}} = \begin{bmatrix} \mathbf{B}^1 & \mathbf{0} \\ \mathbf{0} & \mathbf{B}^2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} \beta_{11}\beta_{11} & \beta_{12}\beta_{21} & \beta_{11}\beta_{21} + \beta_{12}\beta_{11} \\ \beta_{11}\beta_{12} & \beta_{12}\beta_{22} & \beta_{11}\beta_{22} + \beta_{12}\beta_{12} \end{bmatrix}$$

$$\boldsymbol{\sigma}^b = \{\sigma_x^1 \;\; \sigma_y^1 \;\; \tau_{xy}^1 \;\; \sigma_x^2 \;\; \sigma_y^2 \;\; \tau_{xy}^2\}^{\mathrm{T}}, \quad \mathbf{b}^b_{\text{bound}} = \{t_1^1 \;\; t_2^1 \;\; t_1^2 \;\; t_2^2\}^{\mathrm{T}}$$

A.6. Objective function and loading constraints

Consider surface loading on a side defined by nodes 1 and 2 which is to be optimized with respect to a local co-ordinate system. Then we have

$$Q^s = \frac{L}{2}\left(\text{sign}(q_1^1)\sigma_n^1 + \text{sign}(q_2^1)\tau^1 + \text{sign}(q_1^2)\sigma_n^2 + \text{sign}(q_2^2)\tau^2\right)$$

or

$$Q^s = \mathbf{c}^{s^{\mathrm{T}}} \boldsymbol{\sigma}^s \tag{A6}$$

where

$$\mathbf{c}^s = \{\text{sign}(q_1^1)\mathbf{B}_1^1 \;\; \text{sign}(q_2^1)\mathbf{B}_2^1 + \text{sign}(q_1^2)\mathbf{B}_1^2 \;\; \text{sign}(q_2^2)\mathbf{B}_2^2\}^{\mathrm{T}}$$

$$\boldsymbol{\sigma}^s = \{\sigma_x^1 \;\; \sigma_y^1 \;\; \tau_{xy}^1 \;\; \sigma_x^2 \;\; \sigma_y^2 \;\; \tau_{xy}^2\}^{\mathrm{T}}$$

and $\mathbf{B}_i^l$ is the ith row of transformation matrix $\mathbf{B}^l$ for node l.

Load orientation constraints, e.g. for node 1, will give relations of the form

$$\frac{\sigma_n^1}{q_1^1} = \frac{\tau^1}{q_2^1}$$

or in matrix form

$$\mathbf{A}^o_{\text{orient}} \boldsymbol{\sigma}^o = \mathbf{b}^o_{\text{orient}} \tag{A7}$$

where

$$\mathbf{A}^o_{\text{orient}} = \frac{\mathbf{B}_1^1}{q_1^1} - \frac{\mathbf{B}_2^1}{q_2^1}, \quad \boldsymbol{\sigma}^o = \{\sigma_x^1 \;\; \sigma_y^1 \;\; \tau_{xy}^1\}^{\mathrm{T}}, \quad \mathbf{b}^o_{\text{orient}} = \{0\}$$

Load profile constraints give relations between the magnitudes of the load components at nodes 1 and 2

$$\frac{\sigma_n^1}{q_1^1} = \frac{\sigma_n^2}{q_1^2} \quad \text{and} \quad \frac{\tau^1}{q_2^1} = \frac{\tau^2}{q_2^2}$$

or in matrix form

$$\mathbf{A}^p_{\text{profile}}\boldsymbol{\sigma}^p = \mathbf{b}^p_{\text{profile}} \tag{A8}$$

where

$$\mathbf{A}^p_{\text{profile}} = \begin{bmatrix} \dfrac{\mathbf{B}^1_1}{q^1_1} & -\dfrac{\mathbf{B}^2_1}{q^2_1} \\ \dfrac{\mathbf{B}^1_2}{q^1_2} & -\dfrac{\mathbf{B}^2_2}{q^2_2} \end{bmatrix}, \quad \boldsymbol{\sigma}^p = \{\sigma^1_x \;\; \sigma^1_y \;\; \tau^1_{xy} \;\; \sigma^2_x \;\; \sigma^2_y \;\; \tau^2_{xy}\}^{\mathrm{T}}, \;\; \mathbf{b}^p_{\text{profile}} = \{0 \;\; 0\}^{\mathrm{T}}$$

Constraints (A7) and (A8) can be assembled to form the final set of loading constraints

$$\mathbf{A}^l_{\text{load}}\boldsymbol{\sigma}^l = \mathbf{b}^l_{\text{load}} \tag{A9}$$

A.7. Yield condition

A smooth hyperbolic approximation of the Mohr–Coulomb yield criterion in the 2D case has the form

$$f = \sigma_m \sin\phi + \sqrt{\bar{\sigma}^2 + a^2 \sin^2\phi} - c\cos\phi \tag{A10}$$

where

$$\sigma_m = \tfrac{1}{2}(\sigma_x + \sigma_y), \quad \bar{\sigma} = \sqrt{\tfrac{1}{2}(s_x^2 + s_y^2) + \tau_{xy}^2}, \quad s_x = \sigma_x - \sigma_m, \;\; s_y = \sigma_y - \sigma_m$$

The gradient ∇f can be expressed in the computationally convenient form

$$\nabla f = \frac{\partial f}{\partial \boldsymbol{\sigma}} = C^1_1 \frac{\partial \sigma_m}{\partial \boldsymbol{\sigma}} + C^1_2 \frac{\partial \bar{\sigma}}{\partial \boldsymbol{\sigma}} \tag{A11}$$

where

$$C^1_1 = \sin\phi, \quad C^1_2 = \alpha = \frac{\bar{\sigma}}{\sqrt{\bar{\sigma}^2 + a^2\sin^2\phi}}, \quad \frac{\partial \sigma_m}{\partial \boldsymbol{\sigma}} = \frac{1}{2}\begin{Bmatrix} 1 \\ 1 \\ 0 \end{Bmatrix}, \quad \frac{\partial \bar{\sigma}}{\partial \boldsymbol{\sigma}} = \frac{1}{2\bar{\sigma}}\begin{Bmatrix} s_x \\ s_y \\ 2\tau_{xy} \end{Bmatrix} = \frac{1}{2\bar{\sigma}}\mathbf{s}$$

The gradient derivatives $\nabla^2 f$ are computed in a similar way according to

$$\nabla^2 f = \frac{\partial^2 f}{\partial \boldsymbol{\sigma}^2} = \frac{\partial C^1_2}{\partial \boldsymbol{\sigma}} \frac{\partial \bar{\sigma}}{\partial \boldsymbol{\sigma}} + C^1_2 \frac{\partial^2 \bar{\sigma}}{\partial \boldsymbol{\sigma}^2} = C^2_1 \frac{\partial \bar{\sigma}}{\partial \boldsymbol{\sigma}} \otimes \frac{\partial \bar{\sigma}}{\partial \boldsymbol{\sigma}} + C^2_2 \frac{\partial \mathbf{s}}{\partial \boldsymbol{\sigma}} \tag{A12}$$

where

$$C^2_1 = \frac{1-\alpha}{\sqrt{\bar{\sigma}^2 + a^2\sin^2\phi}} - \frac{1}{\bar{\sigma}}, \quad C^2_2 = \frac{1}{2\bar{\sigma}}, \quad \frac{\partial \mathbf{s}}{\partial \boldsymbol{\sigma}} = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} & 0 \\ -\frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

A.8. Assembly of constraint equations

Using Equations (A3)–(A5) and (A9) the various equality constraints may be assembled to give the overall equality constraint matrix according to

$$\mathbf{A} = \sum_{e=1}^{E} \mathbf{A}^e_{\text{equil}} + \sum_{d=1}^{Ds} \mathbf{A}^d_{\text{equil}} + \sum_{b=1}^{Bn} \mathbf{A}^b_{\text{bound}} + \sum_{l=1}^{Ld} \mathbf{A}^l_{\text{load}} \tag{A13}$$

where the coefficients are inserted into the appropriate rows and columns and E is the total number of elements, Ds is the total number of discontinuities, Bn is the total number of boundary edges with prescribed tractions, and Ld is the total number of loading constraints.

Similarly, the vector $\mathbf{b}$ and objective function coefficients $\mathbf{c}$ are assembled according to

$$\mathbf{b} = \sum_{e=1}^{E} \mathbf{b}^d_{\text{equil}} + \sum_{d=1}^{Ds} \mathbf{b}^e_{\text{equil}} + \sum_{b=1}^{Bn} \mathbf{b}^b_{\text{bound}} + \sum_{l=1}^{Ld} \mathbf{b}^l_{\text{load}} \tag{A14}$$

$$\mathbf{c} = \sum_{s=1}^{S} \mathbf{c}^s \tag{A15}$$

where S is the total number of boundary edges over which the surface forces are to be optimized.

APPENDIX B: SOLUTION ALGORITHMS

B.1. Algorithm 1 (Two stage quasi-Newton algorithm)

1. (*Initialization*)
   - 1.1. Read $\beta, \varepsilon_c, \varepsilon_\lambda, \varepsilon_f, \delta_f^0, \delta_\lambda^0, \varkappa_{\min}$
   - 1.2. Set $\mathbf{x} = \mathbf{x}^0$ and $\mathbf{d} = \mathbf{d}^0$
   - 1.3. Find $s = \min_{j \in J} s_j | f_j(\mathbf{x} + s_j \mathbf{d}) = \delta_f^0 f_j(\mathbf{x})$
   - 1.4. Set $\mathbf{x} = \mathbf{x} + s\mathbf{d}$
   - 1.5. For all $j \in J$ set $\bar{\lambda}_j = 1$
2. (*Increment estimate*)
   - 2.1. For all $j \in J$ do steps 2.1.1–2.1.5
     - 2.1.1. Compute $\nabla f_j(\mathbf{x})$ and $\mathbf{Z}_j = \nabla^2 f_j(\mathbf{x})$
     - 2.1.2. Compute $\mathbf{H}_j^{-1} = [\bar{\lambda}_j(\varepsilon_j \mathbf{I}_j + \mathbf{Z}_j)]^{-1}$
     - 2.1.3. Compute $\mathbf{Z}_j = R_j \mathbf{Z}_j$
     - 2.1.4. Compute $f_j(\mathbf{x})/\bar{\lambda}_j$ and mount in $\mathbf{F}$
     - 2.1.5. Mount $\mathbf{H}_j^{-1}$ in $\mathbf{H}^{-1}$ and $\nabla f_j(\mathbf{x})$ in $\mathbf{G}$
   - 2.2. Compute the matrices

$$\mathbf{Q} = \mathbf{H}^{-1}\mathbf{G}^{\mathrm{T}}$$
$$\mathbf{W} = \mathbf{G}\mathbf{Q} - \mathbf{F}$$
$$\mathbf{D} = \mathbf{H}^{-1} - \mathbf{Q}\mathbf{W}^{-1}\mathbf{Q}^{\mathrm{T}}$$
$$\mathbf{K} = \mathbf{A}\mathbf{D}\mathbf{A}^{\mathrm{T}}$$

2.3. Decompose $\mathbf{K}$
2.4. If $\mathbf{K}$ becomes singular go to step 3.4
2.5. Solve $\mathbf{K}\boldsymbol{\mu}_0 = \mathbf{ADc}$
2.6. Compute $\boldsymbol{\lambda}_0 = \mathbf{W}^{-1}\mathbf{Q}^{\mathrm{T}}(\mathbf{c} - \mathbf{A}^{\mathrm{T}}\boldsymbol{\mu}_0)$
2.7. Compute $\mathbf{d}_0 = \mathbf{D}(\mathbf{c} - \mathbf{A}^{\mathrm{T}}\boldsymbol{\mu}_0)$

3. (*Check convergence*)

3.1. Compute $\lambda_{\max} = \max_{j \in J} \lambda_{0j} \,|\, \lambda_{0j} > 0$
3.2. If $\|\mathbf{c} - \mathbf{A}^{\mathrm{T}}\boldsymbol{\mu}_0 - \mathbf{G}^{\mathrm{T}}\boldsymbol{\lambda}_0\| > \varepsilon_c \|\mathbf{c}\|$ go to step 4
3.3. For all $j \in J$ do steps 3.3.1–3.3.2

3.3.1. If $\lambda_{0j} < -\varepsilon_\lambda \lambda_{\max}$ go to step 4
3.3.2. If $\lambda_{0j} > \varepsilon_\lambda \lambda_{\max}$ and $f_j(\mathbf{x}) < -\varepsilon_j$ go to step 4

3.4. Exit with optimal solution $\mathbf{x}^*$

4. (*Deflection*)

4.1. For all $j \in J$ compute $\varkappa_j^0 = \|\mathbf{Z}_j \mathbf{d}_{0j}\| / (\|\nabla f_j\| \, \|\mathbf{d}_{0j}\|)$
4.2. Compute $\varkappa_{\max} = \max_{j \in J} \varkappa_j^0$
4.3. For all $j \in J$ do steps 4.3.1–4.3.2

4.3.1. Compute $\varkappa_j = \max[\varkappa_j^0 / \varkappa_{\max}, \varkappa_{\min}]$
4.3.2. Compute $u_j = \|\nabla f_j\| \, \|\mathbf{d}_{0j}\| \varkappa_j$

4.4. Compute $\theta_d = \min[(1 - \beta)\mathbf{c}^{\mathrm{T}}\mathbf{d}_0 / |\boldsymbol{\lambda}_0^{\mathrm{T}}\mathbf{u}|, 1]$
4.5. Solve $\mathbf{K}\boldsymbol{\mu} = \mathbf{ADc} - \theta_d \mathbf{AQW}^{-1}\mathbf{u}$
4.6. Set $\mathbf{d} = \mathbf{D}(\mathbf{c} - \mathbf{A}^{\mathrm{T}}\boldsymbol{\mu}) - \theta_d \mathbf{QW}^{-1}\mathbf{u}$

5. (*Line search*)

5.1. Set $\delta_f = \min[\delta_f^0, |\mathbf{c}^{\mathrm{T}}\mathbf{d}_0| / |\mathbf{c}^{\mathrm{T}}\mathbf{x}|]$
5.2. Find $s = \min_{j \in J} s_j \,|\, f_j(\mathbf{x} + s_j \mathbf{d}) = \delta_f f_j(\mathbf{x})$

6. (*Updating*)

6.1. Set $\mathbf{x} = \mathbf{x} + s\mathbf{d}$
6.2. Set $\delta_\lambda = \min[\delta_\lambda^0, \|\mathbf{d}\| / \|\mathbf{x}\|]$
6.3. For all $j \in J$ set $\bar{\lambda}_j = \max[\lambda_{0j}, \delta_\lambda \lambda_{\max}]$
6.4. Go to step 2

B.2. Algorithm 2 (*Two phase algorithm for lower bound limit analysis*)

1. (*Handle equalities*)

1.1. Factorize $\mathbf{AA}^{\mathrm{T}}$
1.2. If detected, delete redundant rows from matrix $\mathbf{A}$ and refactorize $\mathbf{AA}^{\mathrm{T}}$
1.3. Solve $\mathbf{AA}^{\mathrm{T}}\boldsymbol{\mu} = -\mathbf{b}$
1.4. Compute $\mathbf{x}^0 = -\mathbf{A}^{\mathrm{T}}\boldsymbol{\mu}$
1.5. Solve $\mathbf{AA}^{\mathrm{T}}\boldsymbol{\mu} = \mathbf{Ac}$
1.6. Compute $\mathbf{d}^0 = \mathbf{c} - \mathbf{A}^{\mathrm{T}}\boldsymbol{\mu}$

2. (*Handle inequalities*)

   2.1. Compute $\alpha^0 = \max_{j \in J} f_j(\mathbf{x}^0)$

   2.2. If $\alpha^0 \leqslant 0$ go to step 7

3. (*Form phase one problem*)

   3.1. Expand $\mathbf{x}$ to $\mathbf{x}_\alpha^{\mathrm{T}} = \{\mathbf{x}^{\mathrm{T}}, \alpha_1, \ldots, \alpha_{r+1}\}$
   3.2. Expand $\mathbf{x}^0$ to $\mathbf{x}_\alpha^{0\mathrm{T}} = \{\mathbf{x}^{0\mathrm{T}}, \alpha_1^0, \ldots, \alpha_{r+1}^0\}$
   3.3. Add equalities $\alpha_{j+1} - \alpha_j = 0,\ j \in J$
   3.4. Modify inequalities according to $f_j(\mathbf{x}_\alpha) - \alpha_j \leqslant 0,\ j \in J$
   3.5. Add inequality $-\alpha_{r+1} \leqslant 0$
   3.6. Set $d_{\alpha i}^0 = 0$ for $i = 1, \ldots, n$
   3.7. Set $d_{\alpha i}^0 = -1$ for $i = n+1,\ n+r+1$
   3.8. Form objective function vector as

   $$c_{\alpha i} = 0 \text{ for } i = 1,\ n+r \text{ and } c_{\alpha_{n+r+1}} = 1$$

4. (*Solve phase one problem*)

   4.1. Solve phase one problem using Algorithm 1 with modified constraints and $\mathbf{x}_\alpha^0,\ \mathbf{d}_\alpha^0,\ \mathbf{c}_\alpha$ as initial data
   4.2. Exit with optimal solution $\mathbf{x}_\alpha^*$

5. (*Check for feasible solution*)

   5.1. If $x_{\alpha_{n+r+1}}^* \neq 0$ print message that phase one problem has no feasible solution and stop

6. (*Extract initial data for phase two*)

   6.1. Restore all constraints to original form
   6.2. Set $x_i^0 = x_{\alpha i}^*$ for $i = 1, \ldots, n$

7. (*Solve phase two problem*)

   7.1. Solve phase two problem using Algorithm 1 with original constraints and $\mathbf{x}^0,\ \mathbf{d}^0,\ \mathbf{c}$ as initial data
   7.2. Exit with optimal solution $\mathbf{x}^*$

REFERENCES

1. Bottero A, Negre R, Pastor J, Turgeman S. Finite element method and limit analysis theory for soil mechanics problems. *Computational Methods in Applied Mechanics and Engineering* 1980; **22**:131–149.
2. Sloan SW. Lower bound limit analysis using finite elements and linear programming. *International Journal for Numerical and Analytical Methods in Geomechanics* 1988; **12**:61–77.
3. Pastor J. Analyse limite: détermination de solutions statiques complètes—Application au talus vertical. *Journal de Mécanique Appliquée*, now *European Journal of Mechanics, A/Solids* 1978; **2**(2):176–196.
4. Abbo AJ, Sloan SW. A smooth hyperbolic approximation to the Mohr–Coulomb yield criterion. *Computers and Structures* 1995; **54**:427–441.
5. Zouain N, Herskovits J, Borges LA, Feijóo RA. An iterative algorithm for limit analysis with nonlinear yield functions. *International Journal of Solids and Structures* 1993; **30**(10):1397–1417.
6. Herskovits J. A two-stage feasible directions algorithm for nonlinearly constrained optimization. *Mathematical Programming* 1986; **36**:19–38.
7. Herskovits J. Feasible direction interior-point technique for nonlinear optimization, *Journal of Optimization Theory and Applications* 1998; **99**(1):121–146.

8. Borges LA, Zouain N, Huespe AE. A nonlinear optimization procedure for limit analysis. *European Journal of Mechanics*, *A/Solids* 1996; **15**(3):487–512.
9. Sloan SW, Booker JR. Removal of singularities in Tresca and Mohr–Coulomb yield criteria. *Communications in Applied Numerical Methods* 1986; **2**:173–179.
10. Lyamin AV. Three-dimensional lower bound limit analysis using nonlinear programming. *PhD Thesis*, Department of Civil, Surveying and Environmental Engineering, University of Newcastle, Australia, 1999.
11. Duff IS, Reid JK. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Transactions on Mathematical Software* 1983; **9**(3):302–325.
12. Prandtl L. Über die Härte plastischer Körper. *Nachrichten von der Gesellschaft der Wissenschaften zu Gottingen Mathematisch-physikalische Klasse* 1920; **12**:74–85.
13. Sloan SW. A steepest edge active set algorithm for solving sparse linear programming problems. *International Journal for Numerical Methods in Engineering* 1988; **26**:2671–2685.
14. Yu HS. Expansion of a thick cylinder of soils. *Computers and Geotechnics* 1992; **14**:21–41.
15. Coulomb CA. Essai sur une application des règles de maximis et minimis à quelques problèmes de statique relatifs à 1'architecture. *Mémoires de Mathématique et de Physique présentés à l'Académie Royale des Sciences*, Paris 1773; **7**:343–382.
16. Sokolovskii VV. *Statics of Granular Media*. Pergamon Press: New York, 1965.
17. Chen WF. *Limit Analysis and Soil Plasticity*. Elsevier Scientific Publishing Company: Amsterdam, 1975.
18. Pastor J, Thai TH, Francescato P. New bounds for the height limit of a vertical slope. *International Journal for Numerical and Analytical Methods in Geomechanics* 2000; **24**:165–182.
19. Cox AD, Eason G, Hopkins HG. Axially symmetric plastic deformation in soils. *Philosophical Transactions of Royal Society of London*, *Series* A 1961; **254**(1036):1–45.
20. Haar A, Von Karman T. Zur Theorie der Spannungszustände in plastichen und sandartigen Medien. *Nachrichten von der Gesellschaft der Wissenschaften zu Gottingen Mathematisch-physikalische Klasse* 1909; **1909**:204–218.

