Structural and Multidisciplinary Optimization (2023) 66:152
https://doi.org/10.1007/s00158-023-03583-z

EDUCATIONAL PAPER

A Python script for discontinuity layout optimization

Linwei He[1] · **Mattia Schiantella**[2] · **Matthew Gilbert**[1] · **Colin C. Smith**[1]

Received: 27 September 2022 / Revised: 24 March 2023 / Accepted: 1 May 2023 / Published online: 21 June 2023

Abstract
Discontinuity layout optimization (DLO) is a powerful numerical limit analysis technique that can be used to identify the collapse load and associated failure mechanism of a solid or structure. The method successfully automates the traditional 'upper bound' method of plasticity, with applications including metal extrusion problems, where die forces are sought, and geotechnical engineering problems, where the stability of foundations or retaining walls are to be established. Notably the basic DLO method uses the same underlying mathematical formulation as 'ground structure'-based truss layout (or 'topology') optimization and is demonstrated in this contribution via a Python script capable of solving plane strain limit analysis problems. Extensions to the basic method are presented to allow treatment of larger-scale problems incorporating cohesive-frictional materials, and with self-weight treated in a new and conceptually elegant way. Finally, various examples are presented to illustrate the capabilities of DLO, with displacement vectors shown to aid interpretation.

Keywords Discontinuity layout optimization · Limit analysis · Python · Education

1 Introduction

Michell (1904) famously demonstrated that optimal truss structures contain an infinite number of bars with infinitesimal areas, so-called Michell continua. It was later observed that Michell continua are remarkably similar to the so-called slip-line fields associated with plane plasticity problems; both involve special geometric forms known as 'Hencky-Prandtl nets'. Consequently, theories developed for plane plasticity problems were transferred to optimal trusses (e.g., Hemp 1958; Prager 1959).

However, despite the known similarities between optimal truss forms and the forms of plane plasticity failure mechanisms, the efficient numerical "ground structure"-based method of identifying optimal truss structures later developed by Dorn et al. (1964) was not applied to plastic analysis problems until comparatively recently, by Smith and Gilbert (2007), who developed the so-called discontinuity layout optimization (DLO) procedure. This is perhaps surprising, given that efficient numerical means of treating plastic limit analysis problems had been sought for many years. For example, the method of characteristics was proposed by Sokolovskii (1965), though this only provides incomplete lower-bound type solutions. Efforts have also been made to enhance limit equilibrium approaches, such as the method of slices for slope stability (e.g., Zhu et al. 2003), but these methods rely on a number of assumptions and the solutions obtained lack formal status. Over the past few decades, finite element limit analysis-based formulations have also been proposed by researchers (e.g., Lysmer 1970; Sloan 1988; Kobayashi 2005; Makrodimopoulos and Martin 2006). Finite element limit analysis formulations usually involve discretization of a body using deformable solid elements, though rigid elements can also be used in conjunction with interface elements placed between solid elements to permit jumps in the stress or strain rate fields (Alwis 2000). When using early finite element limit analysis methods, *a priori* knowledge of the failure mechanism was often required to obtain accurate solutions, with the user required to tailor the mesh discretization on a case-by-case basis. This was to allow singularities in the stress and/or displacement rate field to be treated in an accurate manner. While the use of adaptive mesh refinement overcomes this to an extent, this

Responsible Editor: Josephine Voigt Carstensen

✉ Linwei He
linwei.he@sheffield.ac.uk

[1] Department of Civil and Structural Engineering, The University of Sheffield, Mappin St, Sheffield S1 3JD, UK

[2] Department of Civil and Environmental Engineering, University of Perugia, Perugia, Italy

comes at the expense of some complexity and also questions over the remeshing criteria to use remain (e.g., Lyamin et al. 2005). Nevertheless, these methods have the significant advantage of being able to model the failure state without the need to resort to incremental solution schemes, required when using the discrete element method (Cundall and Strack 1979) or non-linear finite elements (De Borst et al. 2012).

On the other hand, by taking advantage of the analogy between slip-line fields and Michell continua, Smith and Gilbert (2007) demonstrated that the numerical layout optimization method developed for truss design problems could be modified so as to able to identify failure mechanisms in plane plasticity problems, with any singularities identified in an entirely natural manner. Also, as with truss layout optimization, use of an adaptive solution scheme (Gilbert and Tyas 2003) means that a very large set of potential discontinuities can be treated, such that highly accurate solutions (e.g., often with $\ll 1\%$ error) can be obtained. To date, several DLO formulations have been proposed, including those capable of identifying in-plane rotational mechanisms (Gilbert et al. 2010; Smith and Gilbert 2013), out-of-plane rotational mechanisms (Gilbert et al. 2014), and three-dimensional translational mechanisms (Hawksbee et al. 2013; Zhang 2017). DLO has to date been applied to numerous applications, ranging from tunnels subject to fire loading (Sun et al. 2019), metal cutting processes (Pritchard et al. 2019), multi-scale masonry analysis (Valentino et al. 2023), and the bearing capacity of volcanic pyroclasts (Galindo et al. 2021), with commercial DLO software tools used not only by industry, but also by researchers seeking to better understand new and longstanding problems alike, e.g., see Leshchinsky and Ambauen (2015), Xie and Leshchinsky (2015), Wang et al. (2017), Liang and Knappett (2017), Zhou et al. (2018), and Zheng et al. (2020). Finally the first textbook on DLO was recently published by Zhang et al. (2022).

Although it has been shown that the DLO procedure can be used to obtain accurate solutions at modest computational cost for many problems, the power of the method still appears under-appreciated by the community. This appears in part to be due to a lack of accessible educational resources for use by researchers and practitioners alike. Although a MATLAB DLO script was previously presented at a specialist conference (Gilbert et al. 2010a), this was limited in that it could only treat small-scale problems involving rectangular problem geometries. Also, self-weight was treated in a somewhat complex manner, not taking advantage of recent research by Smith and Gilbert (2022) that enables complex domain geometries to be handled elegantly. (The alternative simplified approach to treating self-weight for complex domain geometries proposed by Salinas and Zegard (2022) involves a number of approximations, while the use of a supplementary FEM analysis to estimate the effect of body forces at each discontinuity line, recently proposed by Zhang et al. (2022), comes at the expense of significant additional complexity.) Finally, the open-source language Python has been quickly gaining popularity in recent years, particularly in industry circles. These issues are all addressed in the present contribution.

The paper is organized as follows: in Sect. 2, the analogy between truss layout optimization and DLO is discussed and the basic DLO formulation is presented; in Sect. 3 the most important sections of the script are explained; in Sect. 4, the formulation is expanded in order to deal with different yield surfaces, dead loads, and problems involving a large number of nodes, taking advantage of an adaptive solution procedure; in Sect. 5 the expanded formulation is used to solve benchmark problems, showing the potential of the method; finally, in Sect. 6 conclusions are drawn.

2 Analogy with optimal truss layout optimization

2.1 Truss layout optimization

To demonstrate the analogy between truss layout optimization and discontinuity layout optimization, it is useful to first revisit the basic truss layout optimization formulation.

For a planar truss design problem involving n nodes and m potential truss bars connecting those nodes, the plastic minimum volume truss layout optimization (equilibrium) formulation for a single-load case problem can be written as follows:

$$\min_{\mathbf{q}} \quad V = \mathbf{c}^{\mathrm{T}}\mathbf{q}, \tag{1a}$$

$$\text{s.t.} \quad \mathbf{B}\mathbf{q} = \mathbf{f}, \tag{1b}$$

$$\mathbf{q} \geq \mathbf{0}, \tag{1c}$$

where V is the volume of the structure, $\mathbf{q}^{\mathrm{T}} = \{q_1^+, q_1^-, q_2^+, q_2^- \ldots q_m^+, q_m^-\}$ is a vector containing tensile and compressive forces, each non-negative: $\mathbf{c}^{\mathrm{T}} = \{l_1/\sigma_1, l_1/\sigma_1, l_2/\sigma_2, l_2/\sigma_2 \ldots l_m/\sigma_m\}$, where l_i and σ_i are respectively the length and yield stress of each bar i. $\mathbf{B}$ is a suitable $(2n \times 2m)$ equilibrium matrix and $\mathbf{f}^{\mathrm{T}} = \{f_1^x, f_1^y, f_2^x, f_2^y \ldots f_n^y\}$, where f_j^x and f_j^y are the x and y components of the external load applied to node j $(j = 1, 2 \ldots n)$. The presence of supports at nodes can be accounted for by omitting the relevant terms from $\mathbf{f}$, together with the corresponding rows from $\mathbf{B}$.

Figure 1a–d presents the steps involved in setting up and solving a simple truss layout optimization problem.

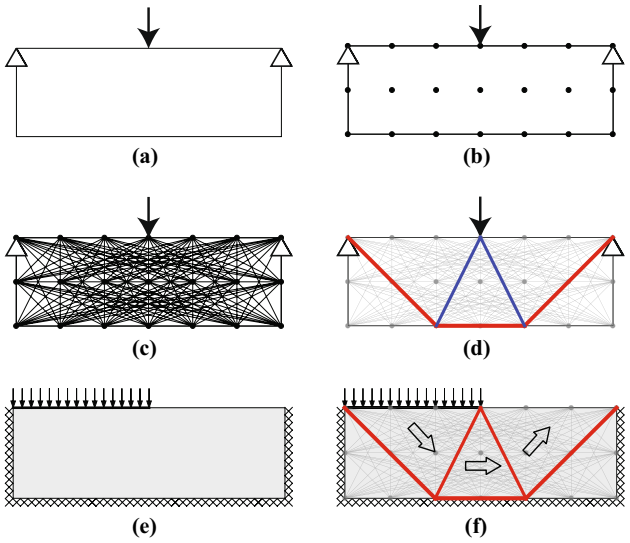

Fig. 1 Truss and discontinuity layout optimization problems: **a** truss design domain, loading and support conditions; **b** domain populated with nodes; **c** nodes interconnected with potential truss bars; **d** optimal truss layout (tensile bars shown in red; compressive bars shown in blue); **e** plastic analysis domain and loading conditions; **f** optimal layout of discontinuities at failure, also showing implied movements of the enclosed solid bodies

2.2 Discontinuity Layout Optimization (DLO)

The layout of discontinuities that form at failure in the case of a quasi-statically loaded perfectly cohesive body in plane strain has been demonstrated to be analogous to the layout of bars forming an optimal truss (Smith and Gilbert 2007). Thus, the 'kinematic' slip-line DLO formulation for a body discretized using m nodal connections (slip-line discontinuities) and n nodes can be written as follows:

$$\min_{\mathbf{d}} \quad E = \mathbf{g}^{\mathrm{T}}\mathbf{d}, \tag{2a}$$

$$\text{s.t.} \quad \mathbf{B}\mathbf{d} = \mathbf{u}, \tag{2b}$$

$$\mathbf{d} \geq \mathbf{0}, \tag{2c}$$

where E is the total internal energy dissipated due to shearing along the discontinuities, $\mathbf{d}^{\mathrm{T}} = \{s_1^+, s_1^-, s_2^+, s_2^- ... s_m^-\}$, where s_i^+, s_i^- are non-negative relative shear displacement jumps between blocks of material along discontinuity i ($i = 1...m$); $\mathbf{g}^{\mathrm{T}} = \{c_1 l_1, c_1 l_1, c_2 l_2, c_2 l_2 ... c_m l_m\}$, where l_i and c_i are respectively the length and associated cohesive shear strength of discontinuity i. $\mathbf{B}$ is a suitable $(2n \times 2\,m)$ compatibility matrix and $\mathbf{u}^{\mathrm{T}} = \{u_1^x, u_1^y, u_2^x, u_2^y ... u_n^y\}$, where u_j^x and u_j^y are the x and y components of the (virtual) displacement jumps imposed at node j ($j = 1...n$). Figure 1e–f presents a coarse nodal discretization DLO solution to the Prandtl punch problem, which results in the same optimal layout as for the truss problem described in Fig. 1a–d; the analogy between truss equilibrium at a node and the compatibility of displacements of bodies sliding relative to each other is illustrated in Fig. 2.

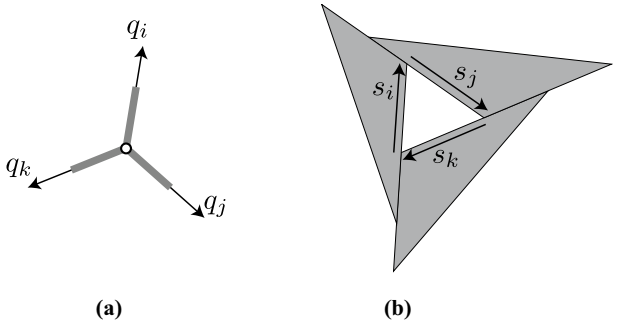

Fig. 2 Analogy between nodal equilibrium and compatibility conditions: **a** truss equilibrium enforced at a node; **b** discontinuity (slip-line) compatibility condition enforced at a node, shown here with infinitesimal displacements magnified for sake of clarity

The DLO formulation can, thus, be interpreted as finding the mechanism that dissipates the minimum internal energy for a given imposed displacement $\mathbf{u}$ on the system. Conservation of energy then means that the load(s) $\mathbf{f_u}$ associated with this displacement can be determined by the equation $\mathbf{f_u}^{\mathrm{T}}\mathbf{u} = \mathbf{g}^{\mathrm{T}}\mathbf{d}$, albeit the interpretation of $\mathbf{f_u}$ and $\mathbf{u}$ in this context is not particularly intuitive or convenient. It is, thus, helpful to reformulate formulation (2) to allow easier interpretation and more general usage, as will be described in the next section.

Note that, for convenience, the terms 'energy dissipation' and 'displacement' are herein used as shorthand for 'rate of energy dissipation' and 'displacement rate' (or 'velocity'), respectively.

2.3 General form of DLO

While formulation (2) demonstrates the analogy between the truss layout optimization and DLO problem formulations, it is useful to introduce a more general formulation for DLO, which for example also allows the potential for dilational displacements to occur along slip-line discontinuities (see Fig. 3):

$$\min_{\mathbf{d},\mathbf{p}} \quad \lambda \mathbf{f}_{\mathrm{L}}^{\mathrm{T}}\mathbf{d} = \mathbf{g}^{\mathrm{T}}\mathbf{p}, \tag{3a}$$

$$\text{s.t.} \quad \mathbf{B}\mathbf{d} = \mathbf{0}, \tag{3b}$$

$$\mathbf{N}\mathbf{p} - \mathbf{d} = \mathbf{0}, \tag{3c}$$

$$\mathbf{f}_{\mathrm{L}}^{\mathrm{T}}\mathbf{d} = 1, \tag{3d}$$

$$\mathbf{p} \geq \mathbf{0}, \tag{3e}$$

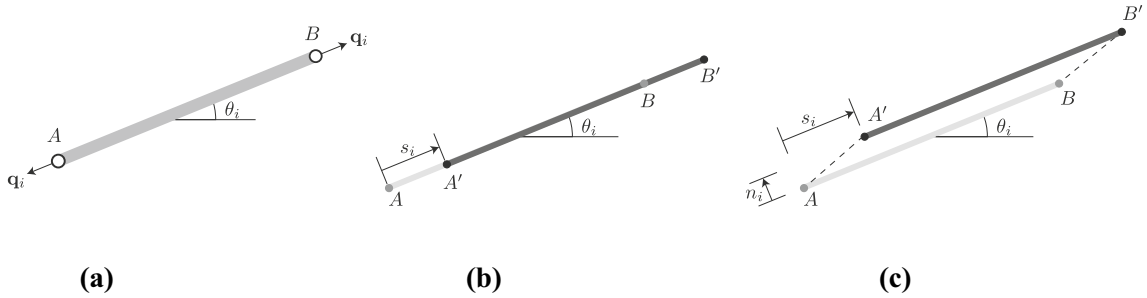

Fig. 3 Variables in truss and DLO problems: **a** force variable of a truss bar i; **b** shear displacement variable of a slip-line i moving from AB to $A'B'$; **c** shear and normal displacement variables. For **b** and **c** when using the sign convention adopted in this paper, the indicated relative displacement jump occurs moving across the discontinuity from below to above

where $\mathbf{f}_{\mathrm{L}} = \{S_{\mathrm{L}1}, N_{\mathrm{L}1}, S_{\mathrm{L}2}, N_{\mathrm{L}2}..N_{\mathrm{L}m}\}$ is a vector containing live loads acting on the discontinuities, $\mathbf{d}^{\mathrm{T}} = \{s_1, n_1, s_2, n_2...n_m\}$ contains relative shear and normal displacements along the discontinuities; λ is the load factor, such that $\lambda \mathbf{f}_{\mathrm{L}}^{\mathrm{T}} \mathbf{d}$ in Eq. (3a) is the work done by live loads. Also, $\mathbf{p}$ is a vector of non-negative plastic multipliers describing the plastic flow at discontinuities, such that the right-hand side of Eq. (3a) is the internal energy dissipation. Therefore, the objective function Eq. (3a) identifies the minimum live load required to cause plastic flow of the structure (i.e., collapse).

While theoretically live loads can be applied to any discontinuity, in general, they will only be applied to discontinuities lying on free boundaries, such that $S_{\mathrm{L}i}$ and $N_{\mathrm{L}i}$ will be zero for any non-free boundary i.

It is also important to note that the displacements involved are all *relative* and that in this paper, the following sign convention is adopted: shear displacements s are taken as positive clockwise (as shown in Fig. 2b) and for normal displacements n, dilational displacements are taken as positive. Thus, 'inward' displacement into a body at a boundary corresponds to dilation at that boundary (where the dilation is acting relative to a fixed 'external' domain). Correspondingly, a normal load at a boundary is considered positive if it is applied inwards with respect to the domain boundary, such that it does positive work. So, for example, if the same positive load is applied to the upper boundary or to the lower boundary, it is oriented downwards or upwards, respectively. Similarly a boundary shear load is considered positive if it acts in an anticlockwise direction around the boundary.

In constraint (3b), the compatibility matrix $\mathbf{B}_i$ of the ith discontinuity can be written as follows:

$$\mathbf{B}_i \mathbf{d}_i = \begin{bmatrix} \alpha_i & -\beta_i \\ \beta_i & \alpha_i \\ -\alpha_i & \beta_i \\ -\beta_i & -\alpha_i \end{bmatrix} \begin{bmatrix} s_i \\ n_i \end{bmatrix} = \mathbf{0}, \tag{4}$$

where α_i and β_i are suitable direction cosines for this boundary.

Constraint (3c) imposes a flow rule linking displacements s_i and n_i. For the example shown in Fig. 2b, only shear plastic flow is involved, such that the flow rule for the ith slip line would be written as follows:

$$\mathbf{N}_i \mathbf{p}_i - \mathbf{d}_i = \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} p_{1,i} \\ p_{2,i} \end{bmatrix} - \begin{bmatrix} s_i \\ n_i \end{bmatrix} = \mathbf{0}, \tag{5}$$

where the normal plastic flow is set to zero. Note that the flow rule constraint (5) is applied to all internal slip lines. For boundary slip lines, the flow rules need to be modified to satisfy the particular boundary conditions involved.

Note that the use of $p_{1,i} + p_{2,i}$ in the work equation (3a), with $p_{1,i}$, $p_{2,i} \geq 0$, ensures that work done is always positive, regardless of the direction of displacement s_i. Since Eq. (3a) is being minimized, the flow rule can be viewed as being equivalent $p_{1,i} + p_{2,i} = |s_i|$; this is illustrated in Table 1, which shows a range of possible $p_{1,i}$, $p_{2,i}$ values for cases where $s_i = 10$ or $s_i = -10$, indicating that the optimal (minimum) value will always occur when $p_{1,i} + p_{2,i} = |s_i|$, where one of the values of p will always be zero.

Live load is applied directly on boundary discontinuities. For sake of simplicity, here only a unit (inward, compressive) normal load is used:

Table 1 Examples of plastic multiplier values, $p_{1,i}$, $p_{2,i}$, showing that the optimal (minimum) work value coincides with $p_{1,i} + p_{2,i} = |s_i|$

$s_i = p_{1,i} - p_{2,i}$	$p_{1,i}$	$p_{2,i}$	Work $(p_{1,i} + p_{2,i}) c_i l_i$	Minimum?
10	10	0	$10 c_i l_i$	Yes
10	11	1	$12 c_i l_i$	No
10	15	5	$20 c_i l_i$	No
− 10	0	10	$10 c_i l_i$	Yes
− 10	1	11	$12 c_i l_i$	No
− 10	5	15	$20 c_i l_i$	No

$$\mathbf{f}_{\mathrm{L}i}^{\mathrm{T}} = \begin{cases} [0, l_i], & \text{for } i \in \mathbb{F}, \\ [0, 0], & \text{otherwise}, \end{cases} \tag{6}$$

where $\mathbb{F}$ is a set containing the loaded boundary discontinuities, and l_i is the length of the ith discontinuity. Note that this defines a 'flexible' load, i.e., the loaded boundary line is free to deform.

2.4 Boundary conditions

Boundary conditions affect the flow rule and work terms associated with the relevant discontinuities. For a fixed boundary, no additional conditions are required, since the relevant discontinuities have the same properties as internal ones. For discontinuities at free boundaries, the flow rules in Eq. (3c) do not need to be applied, since there is no requirement that the displacements (i.e., s_i and n_i) are coupled. In addition, since no internal energy is dissipated on free boundaries, plastic multiplier terms are not included in the work calculation, Eq. (3a). Appendix 1 shows all boundary conditions considered in this work.

In the examples considered in this paper, each boundary is visually represented as follows:

- **Free boundary**: line only
- **Fixed boundary**: cross hatch
- **Symmetry boundary**: dot-dash line
- **Loaded boundary**: directional load arrows

3 Python implementation of basic DLO formulation

The formulation described in Sect. 2.3 has been programmed in the Python script `dlo_basic`.

3.1 Program code excerpts

Python is an open-source high-level interpreted programming language that is becoming an increasingly popular tool when solving scientific and engineering problems.

The key parts of the formulation described in Sect. 2.3 are now associated with the corresponding program code. Specifically, function `DLO` performs the high-level steps required to solve a DLO problem.

First, a polygonal problem domain is created:

```
poly = Polygon(vt)
```

where here the geometrical library `shapely` is used to generate a polygon using its vertices `vt`.

Nodes and discontinuities are then created:

```
Nd = createNodes(poly)
Cn = createDiscontinuities(poly, Nd)
```

where `Nd` is a $(n \times 2)$ array of nodes, with rows defining the x and y coordinates of nodes. `Cn` is a $(m \times 3)$ array of discontinuities, with each row defining the indices of the connected nodes and the discontinuity length. Note that only nodes and discontinuities lying entirely inside the polygonal domain are created. Also, to remove redundant collinear discontinuities, overlapping connections are filtered out by imposing the following condition:

```
if gcd(int(dx), int(dy)) == 1:
```

where `gcd` is a function that finds the greatest common divisor of the x and y increments, `dx` and `dy`. Note that for sake of simplicity, only regular Cartesian nodal grids are considered here; alternative nodal grid and connection schemes can, however, be employed if required, e.g., see Zegard and Paulino (2014).

Boundary conditions are then defined:

```
bd = setCnBoundaryCondition(Nd,Cn,vt,edgebd)
```

which generates an array `bd` defining boundary conditions for all discontinuities.

A DLO problem is then set up and solved:

```
factor, d, p = solveLP(Nd, Cn, bd, mat)
```

and results are displayed graphically:

```
plot(vt, d, Cn, Nd, bd)
```

Further details of the key steps involved are now presented.

3.1.1 Setting up the DLO optimization problem

As successfully utilized by He et al. (2019), the convex optimization package `cvxpy` (Diamond and Boyd 2016) is here used to solve the minimization problem Eq. (3), processed via function `solveLP`.

Firstly, all coefficient vectors and matrices in problem (3) are obtained:

```
B = calcB(Nd, Cn)
N = calcN(bd)
g = calcg(Cn, mat['cohesive'])
fL = unitLoad(bd)
```

Boundary conditions are then considered:

```
activeN, activeG = boundaryConditions(bd)
```

which generate two vectors `activeN` and `activeG`, containing binary data used later to impose boundary conditions when working with the flow rule matrix `N` and the energy dissipation vector `g`.

Optimization variables are then created:

```
d = cvx.Variable(2 * m)
p = cvx.Variable(2 * m)
```

To improve readability, mathematical expressions are written in as natural a format as possible. Also note that matrix multiplications in `cvxpy` (from version 1.0) are defined using the symbol "@". The objective function Eq. (3a) is created using:

```
energy = (g * activeG).transpose() @ p
obj = cvx.Minimize(energy)
```

where the array `activeG` is used as a mask to set certain coefficients in `g` to 0, to fit the boundary conditions involved.

All constraints in problem (3) are contained in a list `cons`:

```
cons = []
cons.append(B @ d == 0)
cons.append(fL @ d == 1)
cons.append(p >= 0)
cons.append(cvx.multiply(N@p-d, activeN) == 0)
```

The optimization problem can now be created and solved:

```
prob = cvx.Problem(obj, cons)
factor = prob.solve(solver='ECOS')
```

where here the LP problem is initially solved via the free `ECOS` solver (Domahidi et al. 2013), which is installed with the `cvxpy` package. The optimization variables can then be obtained:

```
d = np.array(d.value).flatten()
p = np.array(p.value).flatten()
```

3.1.2 Compatibility constraints

To improve computer memory efficiency, the compatibility matrix in Eq. (4) is stored in a sparse matrix. Therefore, it is necessary only to store the values and locations of non-zeros (i.e., row and column identifiers) in matrix **B**. Since `numpy` can handle element-wise calculations in arrays, it is convenient to define local compatibility matrices for all discontinuities:

```
s, n = np.arange(0,2*m,2), np.arange(1,2*m,2)
alpha, beta = X / l, Y / l
values = [[alpha, -beta],
          [beta, alpha],
          [-alpha, beta],
          [-beta, -alpha]]
rows = [n1*2,
        n1*2+1,
        n2*2,
        n2*2+1]
columns = [s, n]
```

where here `s` and `n` are respectively index vectors of shear and normal displacements in **d**; see Eq. (4). Also `n1` and `n2` are indices of the first and second nodes connected by discontinuities.

The above script collects all non-zeros in the **B** matrix and their corresponding locations. To create the sparse matrix, the following function is called:

```
return toSparseMatrix(values, rows, columns,
    shape = (n*2, m*2))
```

which creates a $2n \times 2m$ sparse matrix using the non-zeros provided.

3.1.3 Flow rule

The flow rule matrix N in Eq. (5) is calculated as follows:

```
values = [[one , -one],
          [zero, zero]]
columns = [s, n]
rows = [np.arange(0,2*m,2),\
        np.arange(1,2*m, 2)]
```

3.1.4 Applied loading

In accordance with Eq. (6), loads can be applied to discontinuities lying along boundaries using the following code fragment:

```
m, l = len(bd), Cn[:,2]
loadedCn = np.where(bd == BD.Load)[0]
fL = np.zeros((m, 2))
fL[loadedCn, 1] = l[loadedCn]
```

Note that this applies a 'flexible' unit load (since no kinematic constraints have been created to link together the displacements of adjacent loaded segments).

3.2 Illustrative examples

The formulation described is now applied to simple literature problems. For all problems, a laptop PC equipped with an Intel I7-7700HQ CPU and running 64-bit Windows 10 was used. These problems are characterized by a pure cohesive model (i.e., one that can be described by the Tresca failure envelope, in which only shear plastic strains occur). Appendix 2 indicates the function name to be called to recreate the following and all other examples presented in the paper.

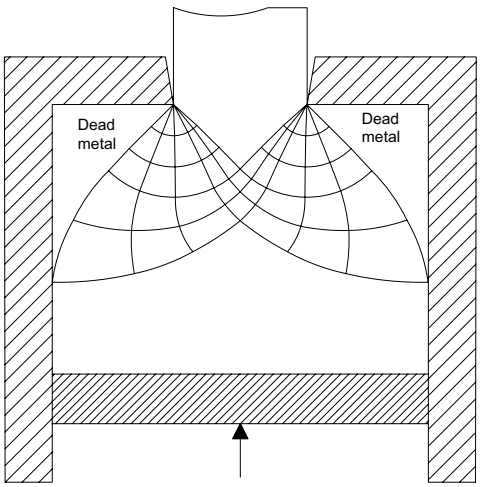

Fig. 4 Metal extrusion: form of slip-line failure mechanism proposed by Hill (1950)

3.2.1 Metal extrusion

The first example is a classical metal extrusion problem considered by Hill (1950), in which metal is pushed through a rectangular die by a ram, leading to 'steady motion' metal extrusion (i.e., a uniform displacement (rate) field exists at the bottom boundary) (Fig. 4).

Figure 5 presents results for three different domain heights. It is evident that a slip-line field similar to that obtained by Hill is only obtained when the loaded boundary is a sufficient distance from the opening. This is because the loading presented in Sect. 3.1.4 is 'flexible,' and does not ensure a uniform displacement field is present at the loaded boundary; this can be addressed by instead using a rigid load, as will be described in Sect. 4.2.

3.2.2 Prandtl punch

Figure 6a shows a variant of the well-known Prandtl punch problem (Hill 1950). Taking symmetry conditions into account, a rectangular domain with 10×5 nodal divisions is used here; see Fig. 6b. The computed load factor $\lambda = 5.222$, which is just 1.56% above the analytical solution of $2 + \pi$. An important benefit of the DLO method compared with comparable finite element analysis methods is that the singularity in the displacement field that occurs at the edge of the punch is identified automatically, without the need for e.g., tailored meshes or adaptive mesh refinement.

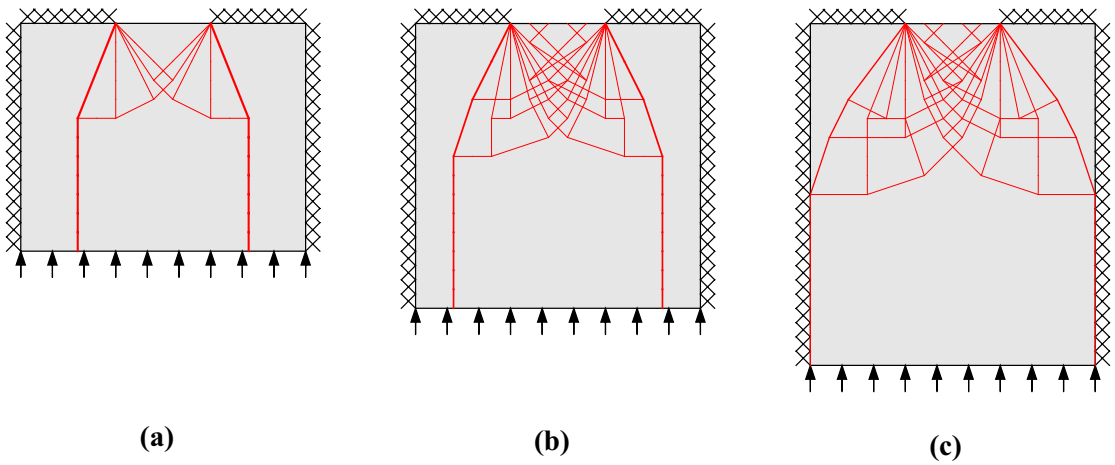

Fig. 5 Metal extrusion—DLO solutions for: **a** 15×12 nodal divisions, $\lambda = 4.237$; **b** 15×15 nodal divisions, $\lambda = 4.841$; **c** 15x18 nodal divisions, $\lambda = 5.320$ (with unit load and 1/3 of top edge open, taking cohesion $c = 1$)

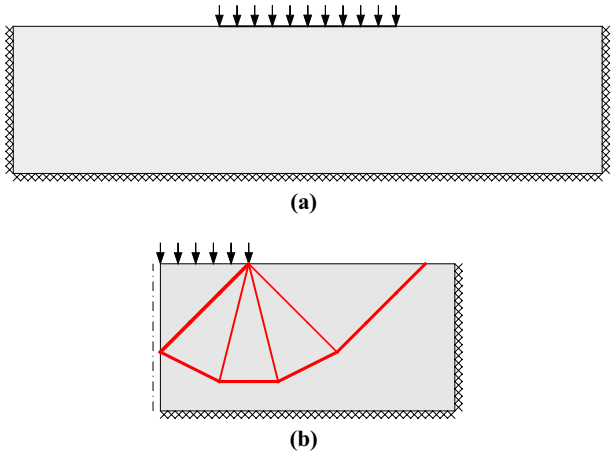

Fig. 6 Prandtl punch: **a** problem specification; **b** DLO solution for half domain discretized using 10×5 nodal divisions, $\lambda = 5.222$ (unit load applied across 3 nodal divisions of top edge, taking cohesion $c = 1$)

4 Extensions

A range of extensions to the basic DLO method are now considered, to increase its range of applicability.

4.1 Cohesive-frictional materials

By changing the flow rule matrix, it is possible to treat different convex yield surfaces. For example, it is straightforward to implement the Mohr–Coulomb model for handling cohesive-frictional materials.

To achieve this, no changes to the compatibility conditions imposed in Eq. (3b) are required, since normal displacements are already represented in the basic DLO formulation (e.g., to permit the presence of normal displacements at boundaries). However, to implement the Mohr–Coulomb model, the flow rule constraint for the ith slip line needs to be modified to read as follows:

$$\mathbf{N}_i\mathbf{p}_i - \mathbf{d}_i = \begin{bmatrix} 1 & -1 \\ \tan\phi & \tan\phi \end{bmatrix} \begin{bmatrix} p_i^1 \\ p_i^2 \end{bmatrix} - \begin{bmatrix} s_i \\ n_i \end{bmatrix} = \mathbf{0}, \tag{7}$$

where $\mathbf{N}_i$ is the local plastic flow matrix, $\mathbf{p}_i$ is a vector containing non-negative plastic multipliers, and ϕ is the angle of friction of the material.

4.2 Rigid loads

In contrast to the 'flexible' loads defined using Eq. (6), it is possible to specify loads that are rigid, i.e., such that the shape of a given loaded boundary line remains fixed, with discontinuities belonging to the boundary all displacing the same amount. This can be implemented by introducing additional equality constraints that link the displacement variables involved:

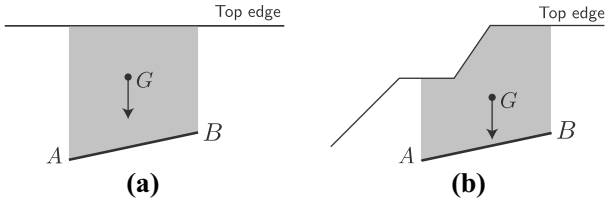

Fig. 7 Calculating the body force G associated with the shaded area for a discontinuity AB using the approach described in Smith and Gilbert (2007): **a** if the top edge is flat then it is straightforward to calculate the area 'above' the discontinuity, e.g., see Gilbert et al. (2010); **b** however, if a non-smooth top edge is present, then a more complex algorithm is required, as the profile of the top edge must be considered

```
linkN, linkS = processCnLinks(bd)
if len(linkN):
    nL = d[linkN * 2 + 1]
    cons.append(nL[1:] == nL[:-1])
if len(linkS):
    sL = d[linkS * 2]
    cons.append(sL[1:] == sL[:-1])
```

where `linkN` and `linkS` are arrays containing the indices of discontinuities to be linked, considering normal and shear displacements, respectively; also `sL` and `nL` are arrays of the corresponding displacement variables. Appendix 1 presents further details of how various types of load can be represented.

4.3 Treatment of body forces

4.3.1 Theory

In previous work (e.g Smith and Gilbert 2007), the work done by body forces was implemented by considering the work done moving a column of material that lies, e.g., vertically above a given slip-line discontinuity. For simple examples involving domains with a flat uppermost boundary (e.g., see Fig. 7a), it is relatively easy to calculate the gravity load imposed by materials lying above any discontinuity. However, for general cases, the calculations can become rather complex. For example, in Fig. 7b, since the uppermost edge is non-smooth, any algorithm developed to calculate the gravity load would need to first identify intersection points and vertices along this edge in order to obtain the polygonal areas above a given underlying discontinuity line. Due to the requirement to calculate intersection points, this process can also become computationally expensive when a large number of discontinuities are present in a given DLO problem. For this reason, handling distributed body forces was

identified as being somewhat challenging when using DLO by He and Gilbert (2016).

However, following recent work by Smith and Gilbert (2022), a much simpler and more elegant approach becomes possible. Here, this will be described from a conceptual standpoint.

Consider first a body containing only a non-dilational material within which a translational mechanism is formed (Fig. 8a). Due to conservation of volume, all normal displacements at the domain boundary must sum to zero. To an external observer, material that is displaced at one boundary discontinuity by a normal displacement must 'reappear' at one or more other boundary discontinuities as illustrated for example in Fig. 8b. Here the volume of material pushed downwards at the top of the slope must equal the volume of material pushed outwards on the slope face, i.e., $n_1 l_1 + n_5 l_5 = 0$. Given that adopted sign convention for n is dilation positive, the interpretation at boundaries is thus that n_1 is positive and n_5 is negative.

To compute the work done by body forces, it is therefore not necessary to track the movement of material throughout a body (since this is done implicitly by enforcing compatibility elsewhere in the DLO formulation), but to simply note the potential of material that vanishes (positive normal displacement), or appears (negative normal displacement) at a boundary, and to sum these to form the body force work term. Since shear displacements do not affect volume, these need not be considered. Additionally, since the calculation is in terms of a body force energy potential, the computation is independent of the direction of the normal displacement. Hence in Fig. 8b, under normal 1 g gravity, the potential energy change of this global movement of material is equal to $-\gamma(l_1 n_1 h_1 + l_5 n_5 h_5)$, where h is the height from an arbitrary datum to the centroid of the discontinuity that is undergoing normal displacement. Since displacement n is assumed to be small, the centroid can be assumed to remain at the mid-point of the discontinuity.

For a material that undergoes volume change on deformation, e.g., dilation, the argument can be extended to include volume generation (or loss) internal to the body, as illustrated in Fig. 8c, with the same principles applying as for the non-dilational material. Here the volume of material pushed downwards on the soil surface plus the volume of dilation on interfaces 2, 3 and 4, must equal the volume of material pushed upwards at the soil surface, i.e., $n_1 l_1 + n_2 l_2 + n_3 l_3 + n_4 l_4 + n_5 l_5 = 0$. Under normal 1 g gravity, the potential energy change of this global movement of material is thus equal to $-\gamma(l_1 n_1 h_1 + l_2 n_2 h_2 + l_3 n_3 h_3 + l_4 n_4 h_4 + l_5 n_5 h_5)$.

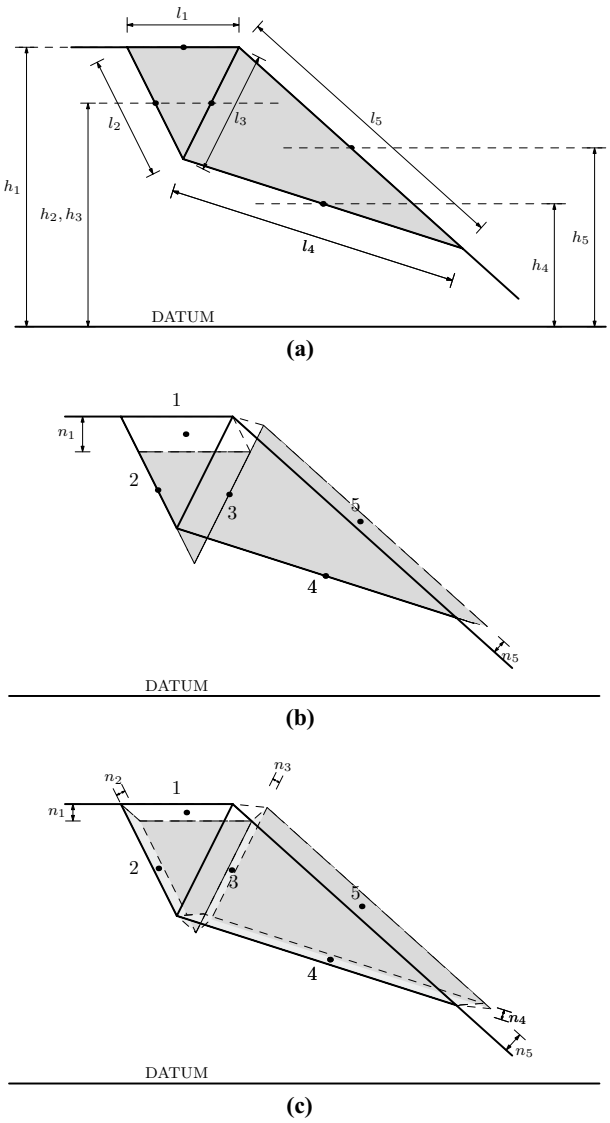

Fig. 8 Calculating body force potential energy changes using the current approach based on normal displacements: **a** example two wedge slope problem, showing dimensions; **b** displaced wedges for problem with non-dilational material with normal displacements at the boundaries only; **c** displaced wedges for problem with dilational material with internal and boundary normal displacements. Centroid of dilating volumes are marked with a dot. Normal displacements are exaggerated for illustrative purposes but are considered infinitesimal in the analysis (hence, for example, the centroid positions for interfaces 1 and 5 coincide with the soil surface)

4.3.2 General equations

Taking the origin as datum, for an individual discontinuity i, the loss of body force potential P_i due to a normal displacement n_i is given by:

$$\Delta P_i = -n_i l_i (k_h \gamma x_m + k_v \gamma y_m), \tag{8}$$

where (x_m, y_m) is the coordinate of the mid-point of the discontinuity, γ is the material unit weight; l_i is the discontinuity length; k_v and k_h are respectively the vertical and horizontal body force accelerations acting in the positive x and y directions (e.g., for normal gravity set $k_v = -1$). Smith and Gilbert (2022) provide an analytical proof of this formulation

in terms of the integral of a body force potential function within the dual equilibrium form of DLO, discussed later in this paper.

When combined with discrete loads applied to any discontinuity, this gives the general equation for loading (live or dead) on a discontinuity i as follows:

$$\mathbf{f}_i^{\mathrm{T}}\mathbf{d}_i = \left[\, S_i,\ \gamma l_i(-k_v \cdot y_m - k_h \cdot x_m) + N_i \,\right] \begin{bmatrix} s_i \\ n_i \end{bmatrix}, \tag{9}$$

where S_i and N_i are respectively shear and normal loads applied to the discontinuity. Body forces may be applied as either live or dead loads. In this paper, only gravity loads are considered as dead loads, and therefore Eq. (9) can be simplified to

$$\mathbf{f}_{\mathrm{D},i}^{\mathrm{T}}\mathbf{d}_i = \left[0,\ \gamma \cdot l_i \cdot y_{m,i}\right] \begin{bmatrix} s_i \\ n_i \end{bmatrix}, \tag{10}$$

which involves significantly simpler computations than those associated with the strip model shown in Fig. 7.

To accommodate such body forces it is necessary to extend Eq. (3a) to now include dead loads (D) as follows:

$$\min_{\mathbf{d},\mathbf{p}} \quad \lambda \mathbf{f}_{\mathrm{L}}^{\mathrm{T}}\mathbf{d} = \mathbf{g}^{\mathrm{T}}\mathbf{p} - \mathbf{f}_{\mathrm{D}}^{\mathrm{T}}\mathbf{d}. \tag{11}$$

4.4 Alternative LP solvers

By default, the LP problem (3) is solved using the open-source solver ECOS (Domahidi et al. 2013) that is distributed with `cvxpy`. However, `cvxpy` also supports many other, potentially more efficient, solvers—albeit these need to be installed separately by users. For example, to use the MOSEK solver (MOSEK 2019), the `solve` command in function `solveLP` is replaced with the following:

```
factor, d, p = prob.solve(solver = cvx.MOSEK,\
    mosek_params={"MSK_IPAR_INTPNT_BASIS":0})
```

where the MOSEK parameter `"MSK_IPAR_INTPNT_BASIS"` disables the unnecessary basis identification step to improve speed.

4.5 Adaptive solution procedure

Similar to the adaptive 'member adding' procedure proposed for numerical truss layout optimization problems (Gilbert and Tyas 2003; He et al. 2019), an adaptive solution scheme can also be employed when solving DLO problems, significantly improving computational efficiency. Figure 9 shows how a solution is obtained for a Prandtl punch problem when using the adaptive solution process. For sake of simplicity, in this work, all potential discontinuities are created (Fig. 9c), though only small subsets of these are actually used to solve problem (3), e.g., Fig. 9d–i. (Note however that, to improve the memory efficiency further, the step in Fig. 9c can alternatively be omitted, with only required subsets stored.)

In the adaptive solution procedure, the dual problem of Eq. (3) is examined. Using duality theory (e.g., see He et al. 2019), the dual problem of Eq. (3), extended in Eq. (11), can be derived as follows:

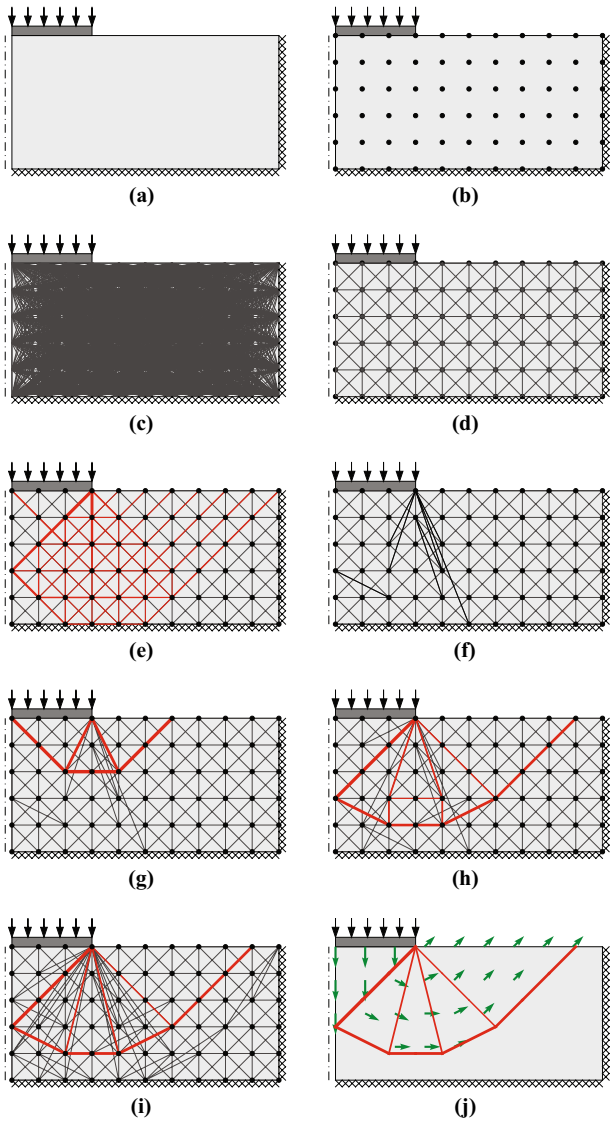

Fig. 9 Solving a DLO problem via the adaptive solution scheme: **a** problem specification; **b** node discretization (10×5 divisions); **c** create set of potential discontinuities by interconnecting all node pairs (total: 1361); **d** activate subset of potential discontinuities (215); **e** iteration 1, solve problem (3), $\lambda = 6.000$; **f** check dual violation using (13), and activate most violating discontinuities via (14); **g** iteration 2, $\lambda = 5.333$, with dual violation; **h** iteration 4, $\lambda = 5.259$, with dual violation; **i** final iteration, $\lambda = 5.222$, no dual violation (total activated discontinuities: 248); **j** discontinuities with non-zero energy dissipation highlighted and showing displacement vectors obtained using algorithm presented in Sect. 4.6 (taking cohesion $c = 1$)

$$\max_{\mathbf{t},\mathbf{q},\lambda} \lambda, \tag{12a}$$

$$\text{s.t.} \quad \mathbf{B}^{\mathrm{T}}\mathbf{t} - \mathbf{q} + \lambda \mathbf{f}_{\mathrm{L}} = -\mathbf{f}_{\mathrm{D}}, \tag{12b}$$

$$\mathbf{N}^{\mathrm{T}}\mathbf{q} \leq \mathbf{g}, \tag{12c}$$

where $\mathbf{t}$ is a vector containing nodal forces, $\mathbf{q} = [S_1, N_1, S_2, N_2, ..., S_m, N_m]^{\mathrm{T}}$ is a vector of discontinuity forces, and where S_i and N_i are shear and normal forces acting on discontinuity i of m discontinuities. Note that the inequality constraint Eq. (12c) now defines the Mohr–Coulomb yield-criterion, e.g., for discontinuity i:

$$\begin{bmatrix} 1 & \tan\phi \\ -1 & \tan\phi \end{bmatrix} \begin{bmatrix} S_i \\ N_i \end{bmatrix} \leq \begin{bmatrix} c_i l_i \\ c_i l_i \end{bmatrix}. \tag{13}$$

When primal problem (3) is solved via the primal-dual interior point method, the nodal forces $\mathbf{t}$ at every node are obtained from the compatibility constraints. Therefore, the discontinuity force vector $\mathbf{q}$ for all potential discontinuities (i.e., discontinuities that are not included in the primal problem) can be obtained from equality constraint Eq. (12b):

```
q = B.transpose().dot(t) + factor * fL + fD
```

Since the potential discontinuities are not included in the primal problem, yield condition Eq. (12c) is unlikely to be satisfied for all these discontinuities. However, the degree to which the yield condition is violated can be calculated from

```
y = N.transpose().dot(q) / g
```

where, for each discontinuity, two violation numbers, corresponding to the two inequality constraints in (13), are obtained. These are extracted using

```
vio1 = y[np.arange(0, m * 2, 2)]
vio2 = y[np.arange(1, m * 2, 2)]
```

The violation for all potential discontinuities can now be sorted, in descending order:

```
vio = np.vstack((vio1, vio2))
vio = np.amax(vio, axis=0)
```

The solution obtained in the next iteration is clearly likely to be improved if discontinuities where violation of the yield condition has been identified are added to the primal problem. However, in the interests of computational efficiency, only discontinuities where violation is greatest should be added initially. Taking m_{v} to be the number of potential (i.e., currently non-activated) discontinuities where violation has been found, and m_{p} as the total number of potential discontinuities, the following selection criteria are used to obtain the set of potential discontinuities to be added to the problem at the next iteration:

$$\Delta m = \begin{cases} \kappa_1 m_{\mathrm{v}}, & m_{\mathrm{v}} \geq \kappa_2 m_{\mathrm{p}} & \text{(14a)} \\ m_{\mathrm{v}}, & m_{\mathrm{v}} \leq \kappa_1 \kappa_2 m_{\mathrm{p}} & \text{(14b)} \\ \kappa_1 \kappa_2 m_{\mathrm{p}}, & \kappa_1 \kappa_2 m_{\mathrm{p}} < m_{\mathrm{v}} < \kappa_2 m_{\mathrm{p}}, & \text{(14c)} \end{cases}$$

where Δm is the number of discontinuities to be added; κ_1 and κ_2 are coefficients determining the percentage of new discontinuities to be added; in this work, both coefficients are taken as 0.05. In Eq. (14), if the number of violated discontinuities is relatively large (i.e., $m_{\mathrm{v}} \geq \kappa_2 m_{\mathrm{p}}$), only a small proportion of the violated discontinuities are added, Eq. (14a). This prevents the problem from growing very rapidly during early iterations of the adaptive solution process, when a large number of dual violations are expected. On the other hand, if the number of violated discontinuities is relatively small (i.e., $m_{\mathrm{v}} \leq \kappa_1 \kappa_2 m_{\mathrm{p}}$), all violated discontinuities are added, Eq. (14b), since adding these will only increase the size of the problem slightly. Alternatively, if the number of violated discontinuities is neither large nor small, a fixed proportion of currently non-activated discontinuities are added, Eq. (14c). Finally, if no violation is detected, considering all potential discontinuities (i.e., $\Delta m = m_{\mathrm{v}} = 0$), then no discontinuities need to be added to the problem and the adaptive procedure terminates.

Note that although the selection criteria given in Eq. (14) is based on heuristics, and different heuristic strategies are possible, the adaptive procedure will always obtain the same load factor as that obtained by solving the full problem, regardless of the specific heuristics used.

4.6 Graphical display of mechanism kinematics

To provide a visual indication of the movements associated with any given failure mechanism, the latter can be overlain with a grid of displacement vectors. This is here achieved by passing horizontal rays across the domain at regular y-intervals. For each ray, the absolute displacement is first set to zero, and then, as the ray passes across the domain from left to right, the absolute displacement is updated as each discontinuity is crossed (by adding the relative displacement occurring at that discontinuity), as shown in Fig. 10. It is convenient to then display vectors of absolute displacement at regular x- and y-intervals; this grid need not coincide with the grid of DLO nodes.

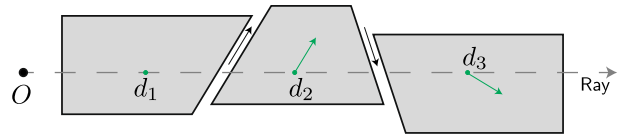

Fig. 10 Generating displacement field by collecting displacements d at intersections between discontinuities and a ray

5 Numerical examples

In this section, a range of examples that illustrate the extensions described in Sect. 4 are presented. All applied loads are 'rigid' unless stated otherwise.

5.1 Metal forming examples

5.1.1 Metal extrusion example revisited

Taking advantage of features introduced in Sect. 4, the metal extrusion example described in Sect. 3.2.1 is now revisited. First, the ram can now be modeled as a rigid load, ensuring 'steady motion' without the need to set the loaded boundary far from the opening. Also, to obtain more accurate solutions than shown in Fig. 5c, the number of nodal divisions considered can be increased by taking advantage of the adaptive solution scheme described in Sect. 4.5. Figure 11 shows results obtained using 30×30 and 60×60 nodal divisions while Table 2 provides a summary of the solutions obtained with varying numbers of nodal divisions.

Note that if the adaptive solution procedure is not employed, the open-source ECOS solver fails to obtain solutions when the number of nodal divisions is increased from 15×15 to 30×30, though solutions to this problem can be obtained using the more robust MOSEK solver. On the other hand, the ECOS solver is able to solve the problem with 60×60 nodal divisions when the adaptive solution scheme is used. Also, increasingly significant reductions in CPU time are observed as problem size increases, when using the MOSEK solver. It is important to note that, although for each problem the CPU cost may vary markedly depending on the solution strategy (i.e., solving

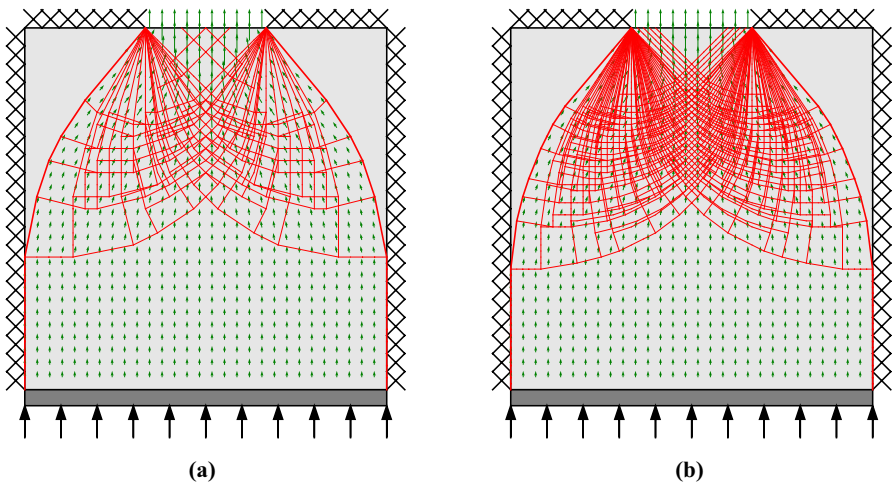

Fig. 11 Metal extrusion example revisited—DLO solutions for: **a** 30×30 nodal divisions, $\lambda = 4.880$; **b** 60×60 nodal divisions, $\lambda = 4.861$ (with unit load and 1/3 of top edge open, taking cohesion $c = 1$)

Table 2 Metal extrusion example revisited: results for various nodal divisions

Nodal divisions ($x \times y$)	Total number of discontinuities	Load factor λ	ECOS solver CPU time Full (s)[a]	ECOS solver CPU time Adaptive (s)[b]	MOSEK solver CPU time Full (s)[a]	MOSEK solver CPU time Adaptive (s)[b]	Speed up factor
15×15	20,074	4.920	3	1	1	1	1
30×30	280,916	4.880	[c]	9	21	9	2.3
45×45	1,362,082	4.868	[c]	52	114	24	4.8
60×60	4,209,056	4.861	[c]	457	514	98	5.2
75×75	10,145,578	4.858	[c]	[c]	2281	183	12.5

[a]Cumulative CPU time spent in function `solveLP`

[b]Cumulative CPU time spent in functions `solveLP` and `stopViolation`

[c]Maximum number of iterations reached in ECOS

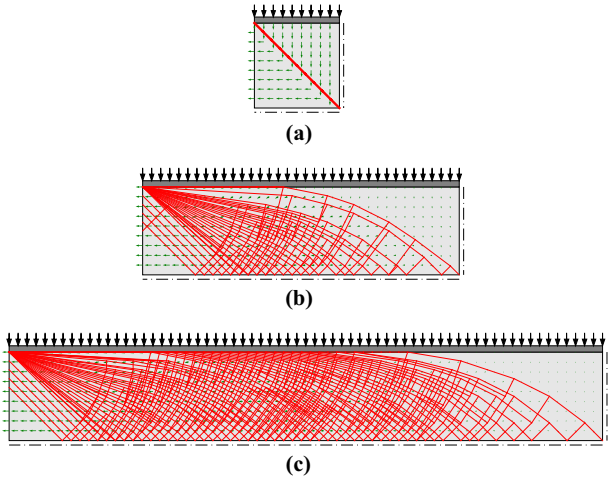

Fig. 12 Metal block compressed between rough platens—DLO solutions for: **a** 10×10 nodal divisions, aspect ratio = 1, $\lambda = 2.000$; **b** 36×10 nodal divisions, aspect ratio = 3.6, $\lambda = 3.325$; **c** 67×10 nodal divisions, aspect ratio = 6.7, $\lambda = 4.978$ (taking cohesion $c = 1$)

the full problem directly vs using the adaptive solution scheme) and solver (i.e., using ECOS vs using MOSEK), the reported load factors remain the same for any given nodal discretization.

5.1.2 Metal block compressed between rough platens

The second example is a rectangular metal block compressed vertically between two rough platens. The analytical solution to this problem has been presented by Chakrabarty (1991, 2006). As observed by Smith and Gilbert (2007), when considering half the width of the metal block, the optimal layout of discontinuities is identical to the optimal layout of truss bars in a minimum volume cantilever truss, e.g., see Johnson (1961) and Lewiński et al. (1994).

Here, three different aspect ratios are considered, with sample graphical output shown in Fig. 12, and with higher resolution numerical solutions provided in Table 3 to permit comparison with analytical solutions. This indicates that highly accurate solutions can be obtained using the DLO procedure described herein.

5.2 Geotechnical examples

Availability of the Mohr–Coulomb failure criterion and the capability to apply body forces makes DLO formulation Eq. (3) suitable for solving a range of geotechnical engineering problems, examples of which are now presented.

Table 3 Metal block compressed between rough platens: results for various block width: height aspect ratios (taking cohesion $c = 1$)

Block aspect ratio	Analytical λ	Numerical Nodal divisions ($x \times y$)	λ	Error%	CPU (s)[a]
1	2.000	25×25	2.000	0.00	2
3.644	3.334	91×25	3.335[b]	0.03	80
6.718	4.894	168×25	4.900[b]	0.11	116

[a]Adaptive solution procedure via the MOSEK solver

[b]Aspect ratios of 3.64 and 6.72 were used in the numerical studies

5.2.1 Bearing capacity problem with cohesive-frictional soil

In this example, the purely cohesive material present in the Prandtl punch problem considered in Sect. 3.2.2 is replaced with a cohesive-frictional material to represent a classical geotechnical bearing capacity problem. Here, the Mohr–Coulomb failure criterion is used, with the analytical solution for a weightless soil material obtainable from $\lambda_0 = c \cdot (N_q - 1)\cot\phi$, where c is the cohesion, ϕ is the angle of friction, and $N_q = \exp(\pi \tan\phi)\tan^2(45 + \phi/2)$ (see e.g., Smith 2005).

Taking advantage of symmetry, a half domain is here discretized using 48×16 nodal divisions, with a unit rigid load applied across 5 nodal divisions. Taking $c = 1$ and $\phi = 25°$, a λ value of 21.0124 can be computed using the DLO script presented herein, which differs from the exact analytical value of 20.721 by just 1.4%. The corresponding failure mechanism is presented in Fig. 13, which clearly shows how the introduction of friction affects the geometry of the failure mechanism.

5.2.2 Retaining wall

In the case of geotechnical engineering problems, self-weight is often important. Thus, the next example involves a smooth-faced rectangular retaining wall with self-weight taken into account. In this case, the 'passive' condition is considered, where the lateral resistance to movement of the wall into the retained soil body is to be determined. The soil cohesion and unit weight are taken as unity and the angle of friction of the soil $\phi = 20°$. The domain is discretized using 40×20 nodal divisions and a unit load is applied on the left boundary.

In this case the computed load factor $\lambda = 23.254$ is just 0.007% higher than the theoretical value of 23.252 (calculated from $\lambda_0 = 0.5K_p\gamma H + 2c\sqrt{K_p}$, where

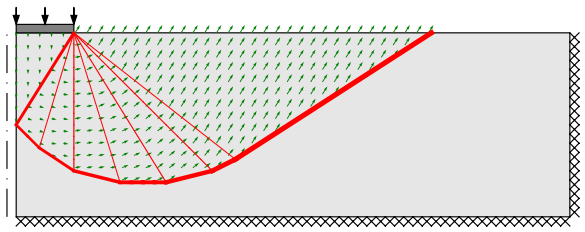

Fig. 13 Bearing capacity problem with cohesive-frictional soil: DLO solution for domain discretized with 48×16 nodal divisions, $\lambda = 21.024$ (with unit load applied over a length of 5 nodal divisions and taking $c = 1, \phi = 25°, \gamma = 0$)

$K_p = \tan^2(45 + \phi/2)$, and $H = 20$ is the wall height, after Rankine (1857) and Bell (1915)). The failure mechanism shown in Fig. 14 also matches perfectly the expected single 'wedge' failure mechanism for a smooth wall. It is straightforward to analyse the equivalent active condition, by reversing the direction of the live load, as defined in Eq. (6).

5.2.3 Terraced slope with crest surcharge

A key benefit of DLO and other generally applicable numerical methods over hand calculation methods is that they can be applied to problems with arbitrary geometry, where the mode of response is not known in advance. Thus, Fig. 15 depicts a more geometrically complex problem, a terraced slope with crest surcharge. The Mohr–Coulomb failure criterion is used, taking $c = 1$ and $\phi = 25°$, and the unit weight is taken as $\gamma = 1$. The non-convex domain is discretized with 20×12 nodal divisions; nodes lying outside the domain are removed in function `createNodes`, and discontinuities intersecting concave edges on the top surface are removed in function `createDiscontinuities`. Since the slope has a non-smooth top surface, calculating the effects of the self-weight using previously described approaches (e.g., Smith and Gilbert 2007; Gilbert et al. 2010) becomes quite involved; see also Fig. 7. However, using the new body force formulation encapsulated in Eq. (8), the load effects can be calculated very efficiently.

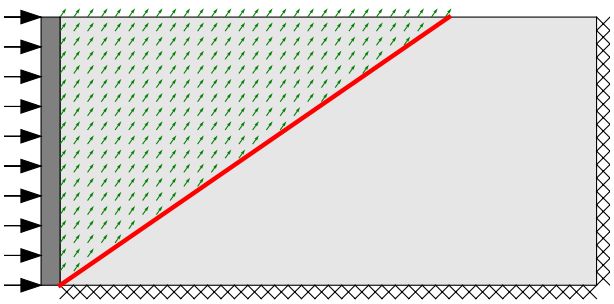

Fig. 14 Retaining wall: DLO solution for 40×20 nodal division case, $\lambda = 23.254$ (smooth wall subject to unit load and taking $c = 1, \phi = 20°, \gamma = 1$)

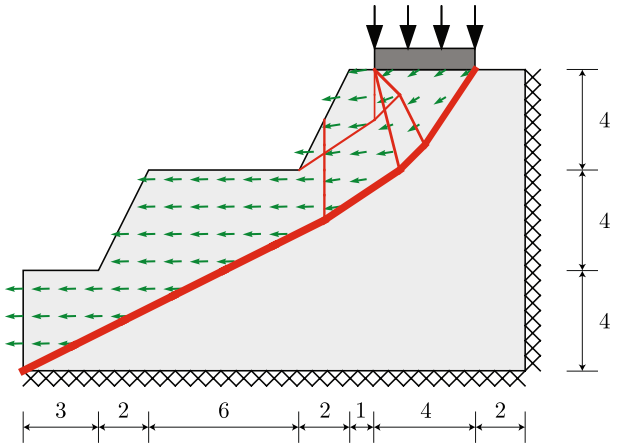

Fig. 15 Terraced slope with crest surcharge: DLO solution for 20×12 nodal division case, $\lambda = 9.170$ (using unit load and taking $c = 1, \phi = 25°, \gamma = 1$)

The computed DLO failure mechanism shown in Fig. 15 includes a piecewise linear-curved slip line, which converts the vertical displacement of the rigid load into horizontal movement of the underlying soil, as observed from the displacement vectors. The example, therefore, demonstrates the flexibility of the DLO method, and its ability to deal with complex geometries and boundary conditions that pose challenges when traditional hand analysis methods are used, given that it will often not be clear *a priori* whether failure will involve e.g., bearing capacity failure, slope failure, or a combined failure mechanism.

6 Concluding remarks

Discontinuity layout optimization (DLO) is a powerful numerical limit analysis technique that can be used to automatically identify the collapse load and associated failure mechanism of a solid or structure. Output is in the form of slip-line failure mechanisms that are familiar to engineers. DLO is a member of a wider family of layout optimization procedures that includes the well-known truss layout optimization method originated by Dorn et al. (1964). These methods rely on node and interconnecting lines, forming what is referred to as a 'ground structure' in the case of the truss layout optimization problem.

In this paper, a practical Python implementation of the DLO method is described. This allows the critical failure mechanism and associated load factor to be obtained using linear optimization for a wide range of translational plane

plasticity problems. Specifically, the presented script is capable of treating problems involving

1. Cohesive-frictional materials.
2. Free, rigid, and symmetry boundaries.
3. Flexible and rigid live loads.
4. Horizontal and vertical body forces, taking advantage of a new approach.
5. Domains with arbitrary boundary geometries.
6. Failure mechanisms that include singularities in the displacement field.

The paper also describes an adaptive solution procedure to enable treatment of large-scale problems, and a routine to plot absolute displacement vectors to help users interpret the graphical output obtained.

Finally, a series of examples encompassing metal forming and geotechnical engineering problems are presented. These demonstrate the simple Python code implementation described in this paper can be used to obtain close agreement with known solutions, considering both identified mechanism and computed failure load. The Python script can also be readily applied to a wide range of other problems, and while the focus has been on single domain problems in this paper, it is relatively straightforward to extend the approach to modeling multiple domains (e.g., see He and Gilbert 2016; Salinas and Zegard 2022), with each domain containing a material with different strength and density.

7 Supplementary material

Downloadable Python scripts (`DLO_basic.py`, `DLO.py` and `example.py`) are available from: https://doi.org/10.15131/shef.data.21175060.

Appendix 1: Boundary conditions

Symmetry boundary: in this case, the material adjacent to the boundary and its symmetrical counterpart must slide in the same direction. This is equivalent to a relative shear displacement that does not dissipate energy.

Loaded boundary (flexible): this boundary condition is equivalent to a free boundary, except for the live load term, Eq. (6).

Loaded boundary (rigid): all relative displacements associated with discontinuities belonging to a rigid load must be the same (i.e., all loads making up a line load remain in-line after deformations have taken place). This requires additional constraints to be applied:

$$\left.\begin{aligned} s_i &= s_j \\ n_i &= n_j \end{aligned}\right\} \forall i,j \in \mathbb{F}_{\text{rigid}}, \tag{14}$$

where $\mathbb{F}_{\text{rigid}}$ is a set containing all discontinuities belonging to the region of the boundary subjected to the rigid load. Also note that there is no energy dissipation or flow rule associated with relative displacements in this case.

Loaded boundary (rigid, laterally constrained): all normal displacements associated with loaded discontinuities must be the same, i.e.:

$$n_i = n_j, \quad \forall i,j \in \mathbb{F}_{\text{lateral}}, \tag{15}$$

where $\mathbb{F}_{\text{lateral}}$ is a set containing all discontinuities belonging to the region of the boundary subjected to the laterally constrained rigid load. Since the load is constrained laterally, there will be energy dissipation associated with shear displacements.

A summary of all boundary conditions considered in this work is shown in Table 4.

Appendix 2: Running the example problems

Two main scripts have been prepared and made available as supplementary material; the first, `DLO_basic.py`, covers the basic DLO formulation described in Sect. 3; the second, `DLO.py`, incorporates the extensions described in Sect. 4. For readers unfamiliar with DLO, it is strongly recommended that `DLO_basic.py` is referred to first as this is shorter and has been designed to be as accessible as possible. (Note that `DLO.py` also includes code optimizations to improve speed, e.g., loops have been replaced with vectorized calculations.)

Both the scripts depend on a number of packages that can be installed via the `conda` cross-platform command line package management tool distributed with Anaconda (which is freely available from https://www.anaconda.

Table 4 Boundary conditions: implementation details

Boundary type	Flow rule	Internal energy dissipation	Additional constraint
Free	None	No	No
Rigid/Internal	s_i and n_i	Yes	No
Symmetry	n_i	Yes[a]	No
Flexible load	None	No	No
Rigid load	None	No	Eq. (14)
Lateral load	s_i	Yes	Eq. (15)

[a]Energy only dissipated in the presence of limiting normal tension or compression at a symmetry boundary

Table 5 Python function calls for all example problems

Description	Figure	Python function
Extrusion (flexible load)	Fig. 5a	`Extrusion12()`
	Fig. 5b	`Extrusion15()`
	Fig. 5c	`Extrusion18()`
Prandtl punch	Fig. 6	`Prandtl()`
Extrusion (rigid load)	Fig. 11a	`Extrusion30()`
	Fig. 11b	`Extrusion60()`
Metal block (compressed)	Fig. 12a	`Metal10()`
	Fig. 12b	`Metal36()`
	Fig. 12c	`Metal67()`
Bearing capacity	Fig. 13	`Bearing()`
Retaining wall (passive failure)	Fig. 14	`Retaining()`
Slope (terraced with surcharge)	Fig. 15	`Slope()`

com/download and also includes a Python development environment):

```
conda install numpy
conda install scipy
conda install shapely
conda install -c conda-forge cvxpy
conda install matplotlib
```

To allow readers to run the example problems, an additional script, `example.py`, has been prepared. This includes functions to run each of the examples considered in this work (Table 5) and to solve all examples considered in the paper in sequence, readers can run the following command in a terminal:

```
python example.py
```

In addition, to obviate the need to install Python and associated packages, the scripts have also been made available via Google Colab (Bisong 2019), at: https://colab.research.google.com/drive/1qgXhlD2JSC_pKU_PF0p3kECFiM9gXTYd. Entering this url in a web browser automatically runs all examples – scroll to the bottom of the page to view the text and graphical output generated for all problems, which will take a few seconds to appear. This link will work for as long as Google Colab services remain publicly available.

Acknowledgements Linwei He, Matthew Gilbert, and Colin Smith acknowledge the financial support provided by the Engineering and Physical Research Council, under grant reference EP/T001305/1.

Declarations

Conflict of interest On behalf of all authors, the corresponding author confirms that there are no conflicts of interest.

Replication of Results Python scripts have been provided to enable all the example problems to be solved by readers.

References

Alwis W (2000) Discrete element slip model of plasticity. Eng Struct 22(11):1494–1504. https://doi.org/10.1016/S0141-0296(99)00094-2

Bell AL (1915) The lateral pressure and resistance of clay and the supporting power of clay foundations. Min Proc Inst Civil Eng 199:233–272

Bisong E (2019) Google Colaboratory, Apress, Berkeley, pp 59–64. https://doi.org/10.1007/978-1-4842-4470-8_7

Chakrabarty J (1991) The analytical solution of some boundary value problems in plane plastic strain. Int J Mech Sci 33(2):89–99. https://doi.org/10.1016/0020-7403(91)90059-C

Chakrabarty J (2006) Theory of plasticity, 3rd edn. Butterworth-Heinemann, Oxford, pp 685–690

Cundall PA, Strack OD (1979) A discrete numerical model for granular assemblies. Géotechnique 29(1):47–65

De Borst R, Crisfield MA, Remmers JJ, Verhoosel CV (2012) Nonlinear finite element analysis of solids and structures. Wiley, Hoboken

Diamond S, Boyd S (2016) CVXPY: a Python-embedded modeling language for convex optimization. J Mach Learn Res 17(83):1–5

Domahidi A, Chu E, Boyd S (2013) ECOS: an SOCP solver for embedded systems. In: 2013 European Control Conference (ECC), IEEE, pp 3071–3076

Dorn WS, Gomery RE, Greenberg H (1964) Automatic design of optimal structures. J de Mecanique 3:25–52

Galindo R, Millán MÁ, Hernández-Gutiérrez LE, Olalla Marañón C, Patiño H (2021) Bearing capacity of volcanic pyroclasts using the discontinuity layout optimization method. Sustainability 13(24):13,733

Gilbert M, Tyas A (2003) Layout optimization of large-scale pin-jointed frames. Eng Comput 20(8):1044–1064

Gilbert M, Smith CC, Haslam IW, Pritchard TJ (2010a) Application of discontinuity layout optimization to geotechnical limit analysis problems. Proc 7th Eur Conf Num Meths Geo Eng. Trondheim, Norway, pp 169–174

Gilbert M, Smith CC, Pritchard TJ (2010b) Masonry arch analysis using discontinuity layout optimisation. Proc Inst Civil Eng 163(3):155–166

Gilbert M, He L, Smith CC, Le CV (2014) Automatic yield-line analysis of slabs using discontinuity layout optimization. Proc R Soc A 470(2168):20140071

Hawksbee S, Smith C, Gilbert M (2013) Application of discontinuity layout optimization to three-dimensional plasticity problems. Proc R Soc A 469(2155):20130,009

He L, Gilbert M (2016) Automatic rationalization of yield-line patterns identified using discontinuity layout optimization. Int J Solids Struct 84:27–39. https://doi.org/10.1016/j.ijsolstr.2015.12.014

He L, Gilbert M, Song X (2019) A Python script for adaptive layout optimization of trusses. Struct Multidisc Optim 60(2):835–847

Hemp W (1958) Report 115: Theory of structural design. College of Aeronautics, Cranfield

Hill R (1950) The mathematical theory of plasticity. Oxford University Press, Oxford

Johnson W (1961) An analogy between upper-bound solutions for plane-strain metal working and minimum-weight two-dimensional frames. Int J Mech Sci 3(4):239–246

KobayashI S (2005) Hybrid type rigid plastic finite element analysis for bearing capacity characteristics of surface uniform loading. Soils Found 45(2):17–27

Leshchinsky B, Ambauen S (2015) Limit equilibrium and limit analysis: comparison of benchmark slope stability problems. J Geotech Geoenviron Eng 141(10):04015043. https://doi.org/10.1061/(ASCE)GT.1943-5606.0001347

Lewiński T, Zhou M, Rozvany G (1994) Extended exact solutions for least-weight truss layouts-Part I: Cantilever with a horizontal axis of symmetry. Int J Mech Sci 36(5):375–398

Liang T, Knappett J (2017) Newmark sliding block model for predicting the seismic performance of vegetated slopes. Soil Dyn Earthq Eng 101:27–40. https://doi.org/10.1016/j.soildyn.2017.07.010

Lyamin AV, Sloan SW, Krabbenhøft K, Hjiaj M (2005) Lower bound limit analysis with adaptive remeshing. Int J Numer Methods Eng 63(14):1961–1974

Lysmer J (1970) Limit analysis of plane problems in soil mechanics. J Soil Mech Found Div 96(4):1311–1334

Makrodimopoulos A, Martin C (2006) Lower bound limit analysis of cohesive-frictional materials using second-order cone programming. Int J Numer Methods Eng 66(4):604–634

Michell AGM (1904) LVIII. The limits of economy of material in frame-structures, The London, Edinburgh, and Dublin. Philos Mag J Sci 8(47):589–597. https://doi.org/10.1080/14786440409463229

MOSEK ApS (2019) MOSEK Optimizer API for C 9.1.13. https://docs.mosek.com/9.1/capi/index.html

Prager W (1959) On a problem of optimal design. Pergamon, Oxford, pp 125–132

Pritchard T, Smith C, Ghadbeigi H, Galindo-Fernandez M, Gilbert M, Ayvar-Soberinas S (2019) Modelling orthogonal and oblique cutting via discontinuity layout optimization. Procedia CIRP 82:37–42. 17th CIRP Conference on Modelling of Machining Operations (17th CIRP CMMO). https://doi.org/10.1016/j.procir.2019.04.009

Rankine WJM (1857) On the stability of loose earth. Philos Trans R Soc Lond II 147:9–27

Salinas D, Zegard T (2022) Discontinuity layout optimization using unstructured meshes and material layering in 2D. Struct Multidisc Optim 65(5):1–16

Sloan SW (1988) Lower bound limit analysis using finite elements and linear programming. Int J Numer Anal Methods Geomech 12(1):61–77

Smith CC (2005) Complete limiting stress solutions for the bearing capacity of strip footings on a Mohr-Coulomb soil. Géotechnique 55(8):607–612

Smith C, Gilbert M (2007) Application of discontinuity layout optimization to plane plasticity problems. Proc R Soc A 463:2461–2484. https://doi.org/10.1098/rspa.2006.1788

Smith CC, Gilbert M (2013) Identification of rotational failure mechanisms in cohesive media using discontinuity layout optimisation. Géotechnique 63(14):1194–1208

Smith CC, Gilbert M (2022) The stress function basis of the upper bound theorem of plasticity. Int J Solids Struct 244(111):565

Sokolovskii VV (1965) Statics of granular media. Pergamon Press, Oxford

Sun Z, Zhang Y, Yuan Y, Mang HA (2019) Stability analysis of a fire-loaded shallow tunnel by means of a thermo-hydro-chemo-mechanical model and discontinuity layout optimization. Int J Numer Anal Methods Geomech 43(16):2551–2564. https://doi.org/10.1002/nag.2991

Valentino J, Gilbert M, Gueguin M, Smith CC (2023) Limit analysis of masonry walls using discontinuity layout optimization and homogenization. Int J Numer Methods Eng 124(2):358–381. https://doi.org/10.1002/nme.7124

Wang L, Leshchinsky B, Evans TM, Xie Y (2017) Active and passive arching stresses in $c' - \phi'$ soils: a sensitivity study using computational limit analysis. Comput Geotech 84:47–57. https://doi.org/10.1016/j.compgeo.2016.11.016

Xie Y, Leshchinsky B (2015) MSE walls as bridge abutments: optimal reinforcement density. Geotext Geomembr 43(2):128–138. https://doi.org/10.1016/j.geotexmem.2015.01.002

Zegard T, Paulino GH (2014) GRAND–ground structure based topology optimization for arbitrary 2D domains using MATLAB. Struct Multidisc Optim 50(5):861–882. https://doi.org/10.1007/s00158-014-1085-z

Zhang Y (2017) Multi-slicing strategy for the three-dimensional discontinuity layout optimization (3D DLO). Int J Numer Anal Methods Geomech 41(4):488–507

Zhang Y, Sun Z, Wang X (2022a) Discontinuity layout optimization: theory, methods and applications. China Communication Press Co Ltd, Beijing (in Chinese)

Zhang Y, Wang X, Wang X, Mang HA (2022b) Virtual displacement based discontinuity layout optimization. Int J Numer Methods Eng 123(22):5682–5694

Zheng G, Yu X, Zhou H, Wang S, Zhao J, He X, Yang X (2020) Stability analysis of stone column-supported and geosynthetic-reinforced embankments on soft ground. Geotext Geomembr 48(3):349–356. https://doi.org/10.1016/j.geotexmem.2019.12.006

Zhou H, Zheng G, Yin X, Jia R, Yang X (2018) The bearing capacity and failure mechanism of a vertically loaded strip footing placed on the top of slopes. Comput Geotech 94:12–21. https://doi.org/10.1016/j.compgeo.2017.08.009

Zhu DY, Lee CF, Jiang HD (2003) Generalised framework of limit equilibrium methods for slope stability analysis. Géotechnique 53(4):377–395. https://doi.org/10.1680/geot.2003.53.4.377