Revista de la Facultad de Ciencias Universidad Nacional de Colombia, Sede Medellín
V 5 N 2 Julio-Diciembre de 2016 • ISSN-e 2357-5749 • Artículo Investigación • Páginas 88 a 104
DOI: https://doi.org/10.15446/rev.fac.cienc.v5n2.59914

PROGRAMA EN CÓDIGO ABIERTO PARA EL ANÁLISIS BIDIMENSIONAL DE ESTABILIDAD DE TALUDES POR EL MÉTODO DE EQUILIBRIO LÍMITE[a]

OPEN CODE PROGRAM FOR THE BIDIMENSIONAL ANALYSIS OF SLOPES' STABILITY BY THE EQUILIBRIUM METHOD

LUDGER O. SUÁREZ-BURGOA[b], EXNEYDER A. MONTOYA ARAQUE[c]

Recibido 05-09-2016, aceptado 20-12-2016, versión final 28-12-2016.
Artículo Investigación

RESUMEN: El artículo realiza una recopilación sobre los desarrollos pasados en la programación computacional referente al análisis de estabilidad de taludes aplicada a la geotecnia. Luego, el artículo presenta el programa desarrollado por los autores, que posibilita resolver problemas de estabilidad de taludes por el método de equilibrio límite mediante las soluciones de Fellenius y de Bishop usando PYTHON3®. El programa consiste de 22 funciones procedimentales independientes, que como conjunto forma un sistema anidado que posibilita correr el programa completo; todo esto ayudado por una interfaz gráfica de usuario que se corre desde la terminal. Los resultados se presentan de forma gráfica. Finalmente, el artículo coloca algunos ejemplos y validaciones. Se puede concluir que el programa es en principio académico porque no presenta la solución general; pero debido a que es un código abierto se puede usar en futuras implementaciones inclusive las generales.

PALABRAS CLAVE: código fuente libre, análisis de estabilidad de taludes, método de equilibrio límite, método de Bishop.

ABSTRACT: This article makes a recopilation about the past developments in computer programming in regard to the analysis of slope stability applied to geotechnics. After that, the article presents the computer program developed by the authors which allows to solve slope stability problems under the method of limit equilibrium with the solutions propossed by Fellenius and Bishop using PYTHON3®. The program consist of 22 procedural independent functions, which as a set form a nested system which allow to run the complete analysis; all of these helped by a graphical user interfase run from the terminal. Results are presented graphically. Finally, this article presents some examples and validations. We can conclude that the program is in principle for academic uses because it does not solve the genral solution; but because it is an open source

[a]Suárez-Burgoa, L. O. & Montoya Araque, E. A. (2016). Programa en código abierto para el análisis bidimensional de estabilidad de taludes por el método de equilibrio. *Revista de la Facultad de Ciencias*, 5 (**2**), 88–104. DOI: https://doi.org/10.15446/rev.fac.cienc.v5n2.59914

[b]Departamento de Ingeniería Civil, Facultad de Minas, Universidad Nacional de Colombia, losuarezb@unal.edu.co.
[c]Departamento de Geociencias, Facultad de Minas, Universidad Nacional de Colombia, eamontoyaa@unal.edu.co.

code it can be used for future implementations inclusive the general ones.

KEYWORDS: free source code, slope stability analysis, limit equilibrium method, Bishop method.

1. INTRODUCCIÓN

La forma de resolver los problemas de análisis bidimensional (2D) de estabilidad de taludes (AET) por el método de equilibrio límite (MEL) en sus versiones originales (*i.e.* hoy en día versiones clásicas) se mantienen prácticamente inalterada; por tanto, es posible afirmar que lo único que ha cambiado es el modo cómo se resuelve el problema a través de la creación de un código computacional, y cómo aquel código interactúa con el usuario.

La primera implementación computacional del método de equilibrio para la estabilidad de taludes se puede determinar con el desarrollo obtenido por Bishop & Morgenstern (1960).

Pese a lo anterior, hoy en día —después de 50 años del creado el primer código para este fin— la comunidad de usuarios de esta herramienta tienen una libertad limitada de emplear algún código abierto que realice tal tarea, bien sea porque no existe como código abierto, porque éste si existe pero presenta problemas de incompatibilidad con los diversos sistemas operativos o porque no está acompañado de un mecanismo que facilite la interacción con el usuario que no simpatiza de ejecutar directamente desde la consola. Esto obliga en gran parte a buscar apoyo en códigos cerrados y/o comerciales; inclusive recurriendo a la piratería de éstos.

La lista de programas de computación de código cerrado para el análisis de taludes bidimensional por el método de equilibrio límite (2D-AET-MEL) es amplia, en comparación de la carencia de programas bajo la modalidad de código abierto. Esta situación es desventajosa debido a que impide la transferencia de conocimiento y fomenta la dependencia tecnológica con los desarrolladores de software privado y privativo.

En las siguientes secciones se presenta el *programa* denominado pyCSS®(del Inglés *circular slope stability pyprogram*). Éste fue desarrollado a partir de 22 funciones del lenguaje PYTHON3® en su versión 3.x, más dos módulos del mismo lenguaje con los que se ejecuta vía archivo de lotes desde la consola, o vía interfaz gráfica. También, se hace un listado de las capacidades y limitaciones del mismo para finalmente presentar los resultados del uso de éste con algunos ejemplos que validan su funcionalidad.

2. ANÁLISIS DE ESTABILIDAD DE TALUDES POR COMPUTADOR

La conceptualización de modelar un deslizamiento a través de la dinámica de un cuerpo rígido cilíndrico sobre una base que encaja el mismo se formuló a principios del siglo XX (Petterson 1916 citado por Steward, et al. (2011), Rendulic 1935 y Fellenius 1939 citado por Creager, et al. (1939)); sin embargo, los cálculos eran tediosos y largos porque se los hacía de forma manual. Pese a esta limitación, se reflejó gran aceptación en la ingeniería, tal como se puede apreciar en los diversos artículos presentados en el Segundo Congreso Internacional de Grandes Presas en 1936 (Ehrenberg, 1936; Fellenius, 1936; Frontard, 1936; Jáky, 1936; May & Brahtz, 1936).

La tarea para el cálculo manual fue acelerada con la elaboración de ábacos de cálculo presentados por Terzaghi (1936) y Taylor (1937) (consulte también a Baker (2003) y Steward, et al. (2011)), y posteriormente mejorados con la introducción del concepto de *coeficientes de estabilidad* y los ábacos presentados por Bishop & Morgenstern (1960) y Hoek & Bray (1977).

Sólo a finales de los años sesenta del siglo pasado se logró crear un programa computacional de análisis de estabilidad de taludes para los métodos más comunes de aquel entonces, que se resumían en el método de Bishop (1955) y el método de Janbu (1954). En aquel tiempo, estos programas fueron considerados como los más complicados que se hayan escrito para un computador para tal fin (Bromhead, 1978; Morgenstern & Price, 1965; Morgenstern & Price, 1967).

Con el desarrollo de la informática en esa misma época, el 2D-AET-MEL fue reformulado para que éste se use en forma automatizada, y sea aplicable a taludes con cualquier forma de superficie de deslizamiento. A razón de esta nueva formulación se presentaron casi en forma simultánea dos métodos que hoy en día se los conoce como el método de Bishop y el método de Janbu, y ambos bajo el principio del *método de las dovelas*.

La cronología de los desarrollos tempranos entre los métodos de Bishop y Janbu es confusa, y siempre fue motivo de discusión porque no se tiene absoluta certeza de quién fue el primero en presentar el método de dovelas; pero pareciera que fue del siguiente modo (Simons, et al., 2001).

1. El artículo de Bishop fue presentado en una conferencia un año antes de que apareciera impreso. Investigaciones separadas permitieron a Janbu y Kenney llegar a un mismo resultado (Kenney trabajando bajo la dirección de Bishop en la Escuela Imperial de Londres).

2. Janbu publicó primero, en 1955, pero en una forma incorrecta, y la tesis de Kenney apareció un año después.

3. Posteriormente, Janbu republicó la forma correcta de las ecuaciones, pero en una forma relativamente inaccesible para los lectores de habla inglesa.

4. En ese lapso, Bishop convenció a Price, que fue uno de los autores del primer programa computacional de análisis de estabilidad (Little & Price, 1958), de probar las ecuaciones del programa de Kenney para deslizamientos no circulares.

Fue a partir de la investigación de Little & Price (1958) donde se encontró que las ecuaciones básicas conducían a otros problemas numéricos cuando eran evaluadas para una alta precisión, debido a que no se obtenía una suficiente precisión como para definir la estabilidad de un deslizamiento [*i.e.* con alrededor de 5 % como el mejor valor]: él y Morgenstern (Morgenstern & Price, 1965; Morgenstern & Price, 1967) desarrollaron entonces un método más sofisticado (también en el Colegio Imperial de Londres) que hoy en día se lo conoce como el método de Morgenstern & Price. Esta vez, los autores estaban seguros de haber superado la complejidad en los cálculos y estaban listos para difundir el uso del método, y más aún debido a la mayor disponibilidad de las computadoras en el medio (Simons, et al., 2001).

Janbu desarrolló su método después, y publicó su procedimiento generalizado de dovelas en 1973, y un número más de métodos también aparecieron publicados a lo largo del fin de la década de los 60 y 70 del siglo XX.

A partir de estos primeros logros se tienen hasta la actualidad un innumerable número de desarrollos orientados a resolver el problema por medio de computadoras, que fueron publicados en estos años en revistas indexadas. Con base a una muestra de 115 artículos publicados en revistas indexadas de alto impacto referente a temas de AET-LEM entre los años 1965 hasta 2010, se observó que el número de publicaciones fue creciendo en cantidad cada 5 años desde 1965 hasta un máximo en el periodo de 1980 a 1985; y a partir de éste, la cantidad de publicaciones fue decreciendo paulatinamente hasta el año 2000. Sin embargo, entre los años 2000 a 2005 se observó nuevamente un incremento en la cantidad de artículos publicados, esta vez tratando el tema del AET-LEM para tres dimensiones (3D), el cual siguió una tendencia en descenso en el segundo lustro (*i.e.* entre el 2005 y 2010).

3. PROGRAMA DESARROLLADO

El método de equilibrio límite (MEL) implica realizar un análisis de estabilidad del macizo que tendrá que deslizarse a lo largo de una superficie de falla sin considerar las deformaciones y las distorsiones que ello implica. Esta gran limitación era tolerada cuando se aplicaba el método en su inicio (*i.e.* hasta los años 90 del siglo pasado); sin embargo, con la aplicación de los métodos de análisis de estabilidad a través del análisis esfuerzo–deformación de la masa del macizo por algún

método numérico (*e.g.* el método de elementos finitos) el MEL se convirtió sólo en una herramienta académica; aseveración que aún es poco aceptada por ingenieros geotecnistas tradicionalistas y empresas consultoras en geotecnia con alta carga analítica, poco tiempo y escaso personal especialista.

El desarrollo de ecuaciones mostrado en la anterior sección se ha introducido dentro de un código en el lenguaje de programación PYTHON3®, que en conjunto definen un programa que se ha denominado pyCSS®, con el cual se puede realizar el 2D-AET-MEL.

Para lograr el presente desarrollo computacional se le dió gran importancia en crear un *marco geométrico* lo suficientemente solvente para poder luego generalizar el método de las dovelas. A este marco se lo entiende como aquellas funciones que permiten definir la geometría de la superficie del talud, el contorno donde abarca el geomaterial, la geometría del nivel freático, la geometría de la superficie de falla y la forma cómo ésta se divide en dovelas en función de la geometría de la superficie del talud y superficie del nivel freático. Luego de tener definido el número de dovelas, se logra agrupar toda la información en cada una de ellas, con el fin de ser usada durante la solución para obtener el factor de seguridad contra el deslizamiento.

El paradigma de programación usado para el marco geométrico del presente desarrollo fue el de *programación estructurada*, donde se deben usar únicamente tres estructuras: secuencia, selección e iteración; considerando innecesario el uso de la instrucción de transferencia incondicional (*i.e.* básicamente el comando `goto`).

En total se desarrollaron 22 funciones haciendo un total de 148,1 kB de espacio, más un archivo de lotes (*i.e. script*) para hacer más fácil la relación del código con el usuario y un archivo que convoca a una sencilla interfaz gráfica que hace aún más intuitiva la ejecución del programa para algunos usuarios.

De las 22 funciones, se gastaron 19 (*i.e.* 103,4 kB) solo para desarrollar el marco geométrico del método, una sola función (*i.e.* 4,1 kB) de visualización y dos funciones (*i.e.* 40,6 bytes) que permiten la convergencia de la solución del problema (Tabla 1).
En lo que respecta el grupo de funciones, las más importantes resultan ser `automaticslipcircles.py` y `onlyonecircle.py`, que obtienen la solución para hallar la superficie crítica y para evaluar una única superficie definida; sin embargo, resulta necesario que el contexto del problema sea definido en todas las funciones del marco geométrico; por tanto resulta también importante tener un solvente marco geométrico.

A continuación se mencionan las principales variables de entrada que se usan para definir el marco introductorio, geométrico y geomecánico para la ejecución del programa; sin embargo, una des-

Tabla 1: Funciones destinadas para realizar distintas tareas.

Tarea Global	Nombres de las Funciones
Marco Geométrico	azimuthangle.py, circleby2ptsradius.py, create2dsegmentstructure.py, defineslipcircle.py, defineswatertable.py, divideslipintoslices.m extractplinefrom2pts.py, interatefbishopsimpsat.py, interateffelleniussat.py, materialboundary.py, obtainmaxdepthdist.py polyarea.py, reportslicestructurevalues.py, sliparcdiscretization.py, tangentlineatcirclept.py, terrainsurface.py, uniquewithtolerance.py, unitvector.py, vertprojection2pline.py,
Visualización	plotslice.py
Solución	automaticslipcircles.py y onlyonecircle.py

cripción más detallada de estas variables y otras más se encuentra en el manual del usuario del programa Suárez-Burgoa & Montoya Araque (2016).

Previo a la introducción de variables que definen la geometría y otros aspectos para la ejecución del programa, se debe definir la información básica de éste; para ello es necesario indicar:

- Nombre del proyecto `projectName`;
- Autor del proyecto `projectAuthor`;
- Fecha de ejecución del análisis `projectDate`.

Para definir el marco geométrico del problema, es necesario:

- Definir la geometría de la superficie del talud y profundidad del macizo con la variables:
  - Altura vertical del talud `slopeHeight`.
  - Pendiente del talud `slopeDip` dado por un vector 1×2, donde el primer término es una distancia horizontal y el segundo una distancia vertical.
  - Distancia horizontal de la corona del talud hacia el límite izquierdo del macizo `crownDist`.
  - Distancia horizontal de la pata del talud hacia el límite derecho del macizo `toeDist`.
- Definir la superficie del nivel freático a partir de la profundidad, respecto la línea horizontal que define la corona del talud `wtDepthAtCrown` y especificar si se va a considerar el talud parcialmente sumergido o si será coincidente con el talud `toeUnderWatertable`.
- Definir las siguientes variables cuando se evalúe una única superficie de falla circular:
  - Punto de la superficie de falla que corta la parte baja del talud `hztDistPointAtToeFromCrown`.
  - Punto de la superficie de falla que corta la parte alta del talud `hztDistPointAtCrownFromCrown`.

  - Radio del círculo de la superficie de falla `slipRadius`.

- Especificar las siguientes variables cuando se evalúen múltiples superficies de falla circular con el fin de encontrar la superficie crítica:

  - Número de superficies que serán evaluadas `numCircles`.
  - Cantidad de incrementos en un radio inicial `radiusIncrement`.
  - Número de veces que un radio aumenta la cantidad anterior `numberIncrements`.
  - Máximo valor de f_s para mostrar en el diagrama de contornos.

- Número de dovelas a usar `numSlices=nDivs`.

Para definir las propiedades físicas del problema, se definen:

- Pesos unitarios del agua `waterUnitWeight` y del suelo (*i.e.* suelo seco) `materialUnitWeight`.
- Propiedades efectivas saturadas o secas (dependiendo del problema) del material:

  - Ángulo de fricción interna efectiva del material `frictionAngleGrad`.
  - Cohesión drenada efectiva del material `cohesion`.

Finalmente, para encontrar la convergencia del cálculo del método de Bishop simplificado se tiene que dar un valor del factor de seguridad inicial (*i.e.* un valor semilla `seedSafetyFactor`), el cual está programado para tomar aquel que se obtenga en primera instancia con el método de Fellenius; esto es fácilmente modificable para considerar otro valor inicial.

Las limitaciones de código, así inicialmente concebido, se listan a continuación:

- Es para taludes con un solo tipo de geometría estándar.
- Es para macizo de un solo material con parámetros últimos efectivos de resistencia seca o saturada. según el criterio de Mohr–Coulomb.
- El nivel freático es horizontal y no considera presiones intersticiales negativas;
- No tiene la opción de aplicación de fuerzas estabilizantes o desestabilizantes externas ni internas.
- Analiza problemas de taludes que miran hacia la derecha y de ocurrir un eventual deslizamiento el sentido del movimiento en masa tendría esta misma dirección (limitante sólo de comodidad visual).
- Obtiene la solución de Fellenius y de Bishop únicamente.

El código del archivo por lotes `finalModule.py` se muestra en el Apéndice 7, éste resulta ser una de las alternativas para que el programa interactúe con el usuario para hacer correr un ejemplo. Ese archivo define y resuelve un problema cuyo resultado se muestra en la Figura 1. De manera similar se pudo proceder para evaluar múltiples superficies y encontrar aquella que sea crítica.

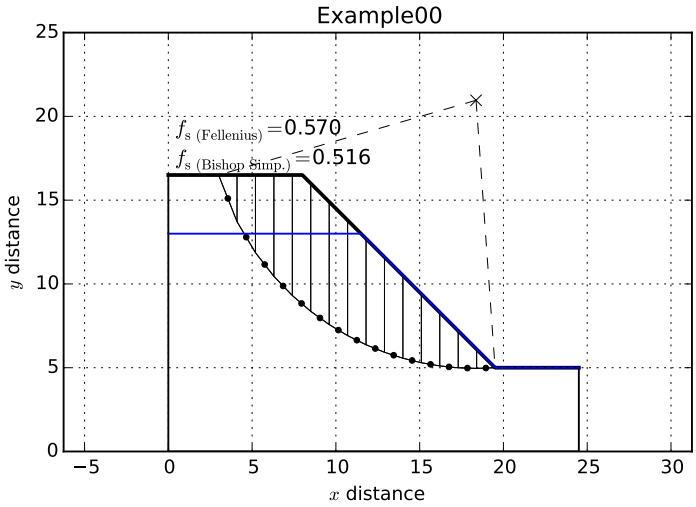


Figura 1: Problema ejemplo para encontrar el factor de seguridad contra la estabilidad con el código elaborado. Fuente: elaboración propia.

La otra alternativa para ejecutar cualquiera de las dos modalidades mencionadas es vía interfaz gráfica que resulta ser más intuitiva para muchos usuarios; ésta se muestra en la Figura 2.

Un exhaustivo estudio podrá desarrollarse con el acceso del código que se puede obtener según se explica en el Apéndice 7.

4. ALGORITMO PARA LA BÚSQUEDA DE LA SUPERFICIE CRÍTICA

El programa pyCSS® incluye una función que posibilita determinar la superficie de falla crítica (*i.e.* aquella cuyo valor de factor de seguridad es el más bajo para un talud definido), ésta lleva el nombre de `automaticslipcircles.py`. Los pasos a seguir para lograr el cálculo son los siguientes.

1. Definir la superficie del talud y el nivel freático si es que se desea (Figura 3(a)).

2. Seleccionar un par de puntos aleatorios sobre la superficie del talud (Figura 3(b)).

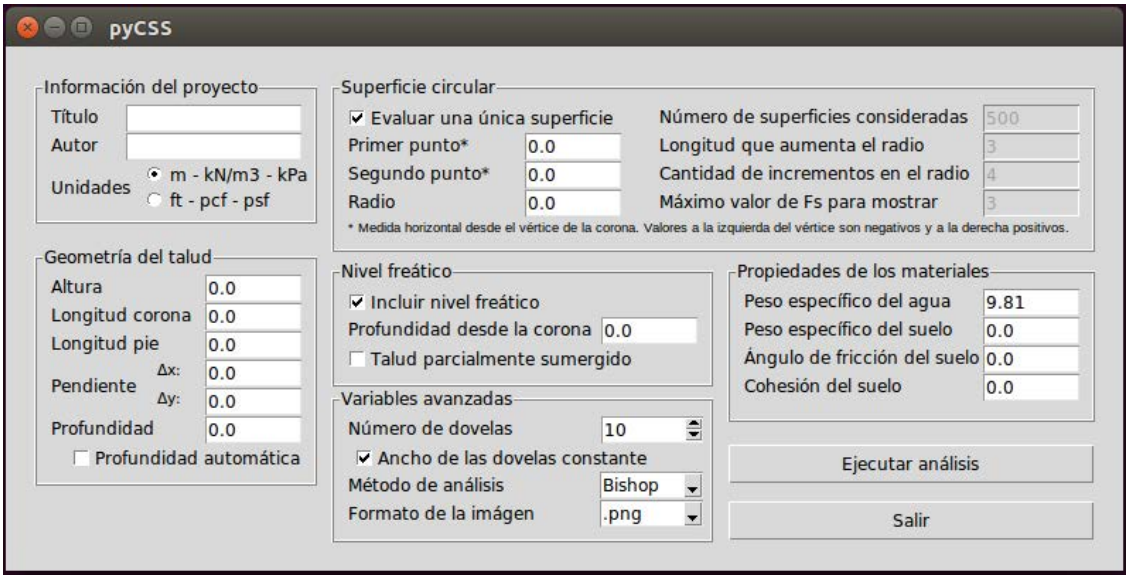


Figura 2: Vista de la interfaz gráfica del programa pyCSS® ejecutada en UBUNTU®.Fuente: Elaboración propia.

3. Encontrar el círculo con el radio mínimo posible que pase por el par de puntos aleatorios y evaluar esta superficie (Figura 3(c)).

4. Aumentar el radio inicial una magnitud l introducida por el usuario para el mismo par de puntos aleatorios.

5. Verificar que la superficie circular no corte la cara y la pata del talud simultáneamente y corregirla sea el caso para proceder a evaluar la nueva superficie como se observa en la Figura 3(d). La corrección consiste en descartar la porción de la superficie circular que no se encuentra dentro de la masa de suelo, conservando el radio intacto.

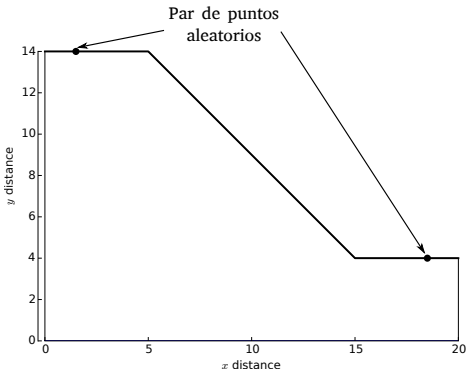
6. Repetir los dos ítem anteriores n veces, donde n también lo introduce el usuario hasta lograr un esquema como el que se muestra en la Figura 3(e).

7. Definir un nuevo par de puntos aleatorios sobre la superficie y repetir todo el ciclo desde el numeral 2. Esta cantidad de puntos aleatorios es función de la cantidad de superficies que se desean evaluar, lo cual es introducido por el usuario.

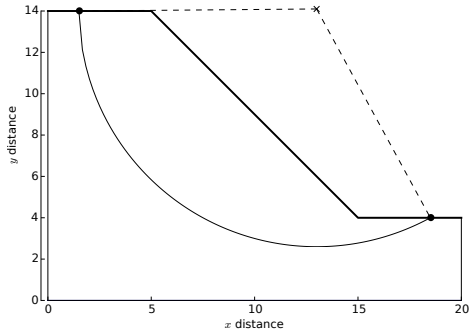
Del procedimiento iterativo anterior se obtiene una nube de puntos asociados a todos los centros de las superficies circulares evaluadas, donde cada uno también está ligado a un valor de factor de seguridad. De este modo, teniendo esta distribución espacial bidimensional de puntos, es posible interpolar los valores de factor de seguridad para visualizar la distribución de los mismos.

Con la visualización de la interpolación mencionada se podrá distinguir la región donde se encontraría el mínimo valor, aunque internamente la función se encarga de buscarlo y mostrárselo al

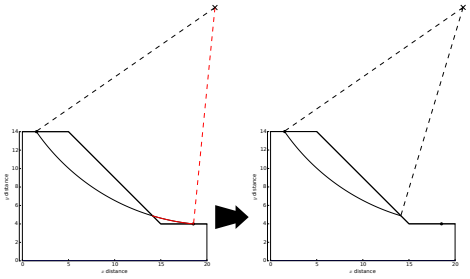
(a) Definición de la superficie del talud.



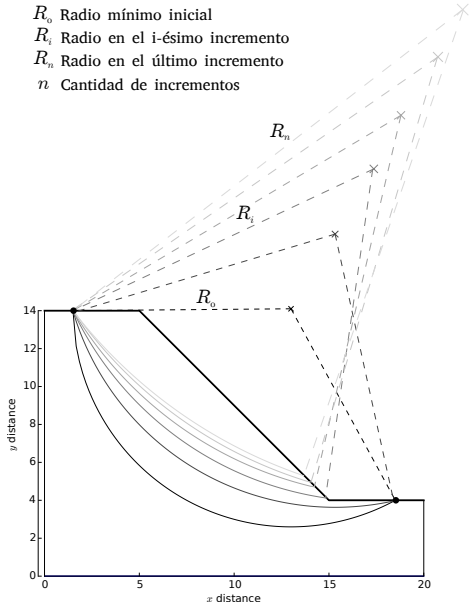
(b) Par de puntos aleatorios



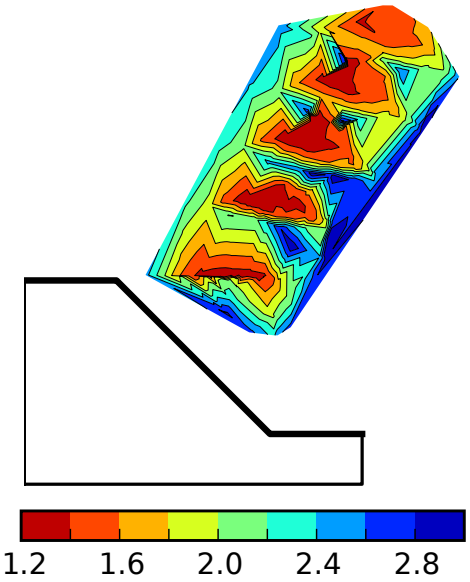
(c) Circulo inicial.



(d) Corrección de una superficie circular errónea



(e) Desarrollo de múltiples círculos que pasan por un par de puntos fijos sobre la superficie del talud.



(f) Distribución de los valores de factores de seguridad después de la evaluación de un talud

Figura 3: Algoritmo de la función que encuentra la superficie crítica. Fuente: Elaboración propia.

usuario. Cabe mencionar, que hay la posibilidad de encontrar una distribución con varios mínimos locales (como se menciona en varios autores en el artículo de Cheng & Lau (2008)), sin embargo, el mínimo global será el que defina la superficie de falla crítica.

5. EJEMPLOS PARA VALIDACIÓN

Para la validación del programa pyCSS® se procedió a comparar los resultados obtenidos de los valores de factor de seguridad respecto a cinco elementos ya validados, estos incluyen dos ejemplos extraídos de revistas internacionales, un manual directriz norteamericano y el paralelo respecto a dos ejemplos cualesquiera ejecutados en *software* comercial ampliamente aceptado y usado en la industria y la academia, tal y como sigue.

1. Ejemplo 2 (figura 6) del artículo de Chang (1992).

2. Cálculo manual del ejemplo de la figura 4-3 del manual de U. S. Army Corps of Engineers (2003), siguiendo los lineamientos planteados por ellos mismos.

3. Ejemplo 1 (figura 2, sección 4.1) del artículo de Zhao, et al. (2014).

4. Igual que el anterior, adicionando nivel freático, evaluado en el software comercial SLIDE®.

5. Igual que el anterior, considerando el talud parcialmente sumergido.

Las comparaciones se resumen en la Tabla 2.

Tabla 2: Comparación de los resultados obtenidos entre pyCSS® y las otras fuentes.

Val.	pyCSS		Otra fuente			Error absoluto e_{abs}	
	Fellenius	Bishop Simp.	Fellenius	Bishop Simp.	Referencia	Fellenius	Bishop Simp.
01	1.944	2.096	1.928	2.080	Chang (1992)	1.6 %	1.6 %
02	2.126	2.250	2.133	2.302	U. S. Army Corps of Engineers (2003)	0.7 %	5.2 %
03	0.967	0.992	0.967	0.992	Zhao, et al. (2014)	0.0 %	0.0 %
04	0.750	0.736	0.749	0.736	SLIDE®	0.1 %	0.0 %
05	0.962	1.004	0.938	0.966	SLIDE®	2.4 %	3.8 %

Se puede observar que las diferencias de valores están por el orden del 5 % o menos, lo que demuestra que el código está desempeñando correctamente los cálculos.

6. DISCUSIÓN

Los métodos manuales se aplicaron mucho antes de la llegada de las computadoras, las hojas de cálculo y los programas de computación de la actualidad.

Pese a que los métodos clásicos fueron automatizados por procesos computacionales —como el presente caso— o se han concebido nuevos métodos, todavía es responsabilidad del usuario (*i.e.* estudiante y/o ingeniero de diseño) el asegurar que las propiedades que definen el modelo (*i.e.* las entradas) y las predicciones de la estabilidad (*i.e.* la salida, reflejada en el valor del factor de seguridad contra el deslizamiento) pasen por una verificación. Inclusive, es un deber explícito del usuario de verificar a mano las salidas que dan los programas de computación; es decir, en una inmediata y transparente manera sin la presencia de la *caja negra* del *misterioso programa*.

Bajo un concepto de caja negra, no es posible revisar los pasos que llevaron a la obtención de un determinado resultado. En el caso de un programa con código abierto —debido a que es libre— el concepto de la *caja negra* no aplica; por tanto, este código puede ser empleado como *herramienta manual* de verificación, cuando haya pasado por una serie de pruebas y depuración (*i.e.* proceso de *debugging*).

El presente programa posibilita potenciales para que la comunidad interesada desarrolle una herramienta con la solución general abierta (expuesta) para el AET-MEL.

7. CONCLUSIONES

Se logró el primer paso de crear un código computacional simple con fines académicos, pero con gran potencialidad de que éste sea el marco de partida para que el mismo crezca con la ayuda de la comunidad científica adscrita a la filosofía planteada de software libre.

De este modo se puede esperar que este programa se convierta en un programa de mayores alcances para el aprovechamiento profesional de la región (*i.e.* Sudamérica). Este aspecto será un estímulo permanente para la apropiación de las nuevas tecnologías y la innovación (da Rosa & Heinz, 2007).

Los siguientes pasos a seguir con la comunidad voluntaria para el desarrollo del presente código son: crear el entorno geométrico más generalizado y bajo el paradigma de *programación orientada a objetos*; generalizar el método para cualquier geometría de talud, nivel freático, capas de materiales y superficie de deslizamiento; generalizar el cálculo del factor de seguridad para los demás métodos del 2D-MEL que existen en la literatura; crear un entorno gráfico mucho más amigable para el

usuario y que sea multiplataforma (i.e. que pueda ser compilado en cualquier sistema operativo).

APÉNDICE A

A. LISTADO DEL PROGRAMA DE LOTES

Este programa por lotes es un ejemplo del empleo de las funciones desarrolladas en este trabajo. El usuario puede copiar el siguiente código en cualquier editor de texto y ejecutarlo desde PYTHON3®, bien sea directamente desde la consola, desde un entorno gráfico creado para PYTHON3®, o introducir los valores a la interfaz gráfica de pyCSS®. En cualquiera de los tres casos, el manual del usuario Suárez-Burgoa & Montoya Araque (2016) , instruye paso a paso el procedimiento para lograr una ejecución satisfactoria del programa y/o sus funciones.

```
''' Description.
This is a minimal module in order to perform a circular arc slope stability
analysis for a trial example.
'''

#-----------------------------------------------------------------------------#
### Add functions directory ###
import sys
sys.path += ['../functions']

#-----------------------------------------------------------------------------#
## Modules/Functions import
import numpy as np
import time

from onlyonecircle import onlyonecircle

#-----------------------------------------------------------------------------#
### Poject data ###
projectName = 'Example00'
projectAuthor = 'Ludger O. Suarez-Burgoa and Exneyder A. Montoya Araque'
projectDate = time.strftime("%d/%m/%y")

#-----------------------------------------------------------------------------#
### Define inputs ###
# The slope geometry #
slopeHeight = [11.5, 'm']
slopeDip = np.array([1, 1])
crownDist = [8, 'm']
toeDist = [5, 'm']
wantAutomaticToeDepth = False
if wantAutomaticToeDepth == True:
toeDepth = ['automatic toe Depth']
else:
toeDepth = [5, 'm']
# The slip arc-circle #
wantEvaluateOnlyOneSurface = False
```

```
if wantEvaluateOnlyOneSurface == True:
hztDistPointAtCrownFromCrown = [-5, 'm']
hztDistPointAtToeFromCrown = [11.5, 'm']
slipRadius = [16, 'm']
else:
numCircles = 1000
radiusIncrement = [5, 'm']
numberIncrements = 5
maxFsValueCont = 3
# Water table depth #
wantWatertable = True
if wantWatertable == True:
wtDepthAtCrown = [3.5, 'm']
else:
wtDepthAtCrown = ['No watertable']
toeUnderWatertable = False
# Materials properties #
waterUnitWeight = [9.8, 'kN/m3']
materialUnitWeight = [17, 'kN/m3']
frictionAngleGrad = [25, 'degrees']
cohesion = [4.5, 'kPa']

### Advanced inputs ###
# Want divide the slip surface in constant width slices? #
wantConstSliceWidthTrue = True
# Number of discretizations of slip surface. #
numSlices = 15
# Number of discretizations of circular arcs. #
nDivs = numSlices
# Select the method to calcualte the safety factor ['Flns', 'Bshp' or 'Allm'] #
methodString = 'Allm'
# Select the output format image ['.eps', '.jpeg', '.jpg', '.pdf', '.pgf', \ #
# '.png', '.ps', '.raw', '.rgba', '.svg', '.svgz', '.tif', '.tiff']. #
outputFormatImg = '.pdf'

#-------------------------------------------------------------------------------#
# Operations for only one slip surface #
if wantEvaluateOnlyOneSurface == True:
msg = onlyonecircle(projectName, projectAuthor, projectDate, slopeHeight, \
slopeDip, crownDist, toeDist, wantAutomaticToeDepth, toeDepth, \
hztDistPointAtCrownFromCrown, hztDistPointAtToeFromCrown, slipRadius, \
wantWatertable, wtDepthAtCrown, toeUnderWatertable, waterUnitWeight, \
materialUnitWeight, frictionAngleGrad, cohesion, wantConstSliceWidthTrue, \
numSlices, nDivs, methodString, outputFormatImg)

#-------------------------------------------------------------------------------#
# Operations for multiple slip surface #
else:
automaticslipcircles(projectName, projectAuthor, projectDate, slopeHeight, \
slopeDip, crownDist, toeDist, wantAutomaticToeDepth, toeDepth, numCircles, \
radiusIncrement, numberIncrements, maxFsValueCont,  wantWatertable, \
wtDepthAtCrown, toeUnderWatertable, waterUnitWeight, materialUnitWeight, \
frictionAngleGrad, cohesion, wantConstSliceWidthTrue, numSlices, nDivs, \
methodString, outputFormatImg)
```

B. ALOJAMIENTO Y DESARROLLO DEL CÓDIGO

El desarrollo y administración del presente código está alojado en la plataforma para el hospedaje de códigos, que permite la colaboración y control de versiones, denominada *GitHub* en el enlace https://github.com/eamontoyaa/pyCSS. Allí podrá obtener un archivo comprimido `.zip` que contiene el programa pyCSS® y el respectivo manual del usuario en formato `.pdf`. A través de este sitio se hace las respectivas descargas, aportes y peticiones de participación en el proyecto.

C. LICENCIA DEL PROGRAMA

Los autores son miembros del *Semillero de Geología Matemática y Computacional* parte del Grupo de Investigación en Geotecnia de la Facultad de Minas de la Universidad Nacional de Colombia en Medellín.

D. DESCARGO DE RESPONSABILIDADES

El presente código computacional se distribuye con la esperanza de que sea útil, pero sin ninguna garantía; sin la garantía implícita en su comercialización o idoneidad para un propósito particular. Consulte la https://opensource.org/licenses/BSD-2-Clause para mayores detalles.

Referencias

Baker, R. (2003). A second look at taylor's stability chart. *Journal of Geotechnical and Geoenvironmental Engineering*, 129(**12**), 1102–1108.

Bishop, A. (1955). The use of the slip circle in the stability analysis of slopes. *Géotechnique*, 5, 7–17.

Bishop, A. & Morgenstern, N. (1960). Stability coefficients for earth slopes. *Géotechnique*, 10, 129–150.

Bromhead, E. (1978). Large landslides in London clay at Herne B, Kent. *Quarterly Journal of Engineering Geology*, 11, 291–304.

Chang, C. (1992). Discrete elements method for slope stability analysis. *Journal of Geotechnical Engineering*, 118(**12**), 1889–1905.

Cheng, Y. M. & Lau, C. K. (2008). Slope stability analysis and stabilization: new methods and insight (1 ed.). New York: Routledge.

Creager, W., Barnes, D., Philippe, R. & Plummer, F. (1939). Selected bibliography on soil mechanics. Manuals of Engineering Practice, 18 18, American Society of Civil Engineers, New York. Prepared by the committee of the soil mechanics and foundation division on soil mechanics bibliography.

da Rosa, F. & Heinz, F. (2007). Guía práctica sobre software libre: su selección y aplicación local en América Latina y el Caribe. Technical report, Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura, Montevideo.

Ehrenberg, J. (1936). Calculation of the stability of dams. In *Transactions of the 2nd Congress on Large Dams*, 4, 331–386.

Fellenius, W. (1936). Calculation of the stability of earth dams. In *ICOLD (Ed.), Transactions of the 2nd Congress on Large Dams*, 4, 445–459.

Frontard, J. (1936). Calculations on the stability of earth dams. In *ICOLD (Ed.), Transactions of the 2nd Congress on Large Dams*, 4, 288–293.

Hoek, E. & Bray, J. (1977). Rock slope engineering (2 ed.). Institution of Mining and Metallurgy.

Jáky, J. (1936). The stability of earth sloped. In *Proceedings of the International Conference on Soil Mechanics*, 2, 125–129.

Janbu, N. (1954). Stability analysis of slopes with dimensionless parameters. *Soil Mechanics* Series 46, University of Harvard, Cambridge.

Little, A. & Price, V. (1958). The use of an electronic computer for slope stablity analysis. *Geotechnique*, 8(**3**), 113–120.

May, D. & Brahtz, J. (1936). Proposed methods fo calculating the stability of earth dams. *In ICOLD (Ed.), Transactions of the 2nd Congress on Large Dams*, 4, 539–576.

Morgenstern, N. & Price, V. (1965). The analysis of the stability of general slip surfaces. *Géotechnique*, 15, 79–93.

Morgenstern, N. & Price, V. (1967). A numerical method for solving the equations of stability of general slip surfaces. *Computer Journal*, 9(**4**), 388–393.

Simons, N., Menzies, B. & Matthews, M. (2001). A short course in soil and rock slope engineering (1 ed.). London: Thomas Telford.

Steward, T., Sivakugan, N., Shukla, S. & Das, B. (2011). Taylor's slope stability charts revisited. *International Journal of Geomechanics*, 11(**4**), 348–352.

Suárez-Burgoa, L. O. & Montoya Araque, E. A. (2016). Circular slope stability pyprogram (pyCSS). Programa de código abierto para el análisis de estabilidad de taludes en 2D, métodos de Fellenius y de Bishop. Manual del usuario. Universidad Nacional de Colombia.

Taylor, D. (1937). Stability of earth slopes. *Journal of the Boston Society of Civil Engineers*, 24(**3**), 337–386. Reprinted in Contributions to Soil Mechanics: 1925–1940, Boston Society of Civil Engineers.

Terzaghi, K. (1936). Critical height and factor of safety of slopes against sliding. In *Proceedings of the International Conference on Soil Mechanics* 1, 156–161.

U. S. Army Corps of Engineers (2003). Engineearing and Design. Slope Stability. Department of the Army.

Zhao, Y., Tong, Z. Y. & Lü, Q. (2014). Slope stability analysis using slice-wise factor of safety. *Mathematical problems in engineering*, 6, 2014.